Tagungsband

HDI 2021

Hochschuldidaktik Informatik

15. – 16. September 2021
in Dortmund

STIFTERVERBAND
Bildung. Wissenschaft. Innovation.

Hochschul Didaktik Informatik GI

FernUniversität in Hagen

Jörg Desel, Simone Opel,
Juliane Siegeris (Hrsg.)

9. Fachtagung Hochschuldidaktik Informatik (HDI) 2021

15. – 16. September 2021
in Dortmund

Vorabdruck der Konferenzbeiträge

Vorwort

Die „Fachtagung für Hochschuldidaktik der Informatik" fand von 2006 bis 2018 im Zweijahresabstand statt. Der Startpunkt war allerdings deutlich früher, nämlich 1998. Erst ab der vierten Ausgabe im Jahre 2010 wurde der aktuelle Titel „Hochschuldidaktik der Informatik" mit Kürzel HDI verwendet. Im vergangenen Jahr 2020 hätte die HDI also ein neuntes Mal stattfinden sollen, sie wurde ein Opfer der Corona-Pandemie. Auch jetzt, ein Jahr später, beschäftigt uns das Virus, aber es ist endlich wieder möglich, wissenschaftliche Fachgespräche in Präsenz durchzuführen. Gerade von Informatikern (jeglichen Geschlechts) wird man erwarten, dass sie technisch auch Online-Konferenzen beherrschen, und tatsächlich läuft die parallel stattfindende Tagung DELFI der GI-Fachgruppe Bildungstechnologien rein virtuell. Die HDI hat aber eine andere Tradition. Sie ist geprägt durch den Austausch und die Diskussion, und dies gelingt auch unter Informatikern in einem Raum besser als über elektronische Netze.

Die HDI 2021 hat einen hervorgehobenen Themenbereich: Diversität. Nun ist Diversität der Studierenden selbst höchst divers und hat sehr viele Ausprägungen. Es gibt Unterschiede im Geschlecht, Bildungshintergrund, Herkunft, Alter, körperlichen Beeinträchtigungen, Motivation und persönliche Zielen, Lernstrategien, usw. usw. Viele aus den Einreichungen ausgewählte Arbeiten widmen sich Fragen der Diversität, und alle drei eingeladenen Fachexpertinnen und -experten berichten über ihre Sicht auf Diversität in der Informatiklehre.

Der Wissenschaftsrat hat vor knapp einem Jahr die Empfehlungen „Perspektiven der Informatik in Deutschland" herausgegeben. In diesem lesenswerten Papier stellt er heraus, dass Hochschullehre der Informatik nicht nur die klassischen Studiengänge Informatik betrifft, sondern viele verschiedene informatiknahe Studiengänge. Darunter ist Data Science vielleicht der aktuell erfolgreichste Newcomer. Die Klientel der Studiengänge unterscheidet sich, die Klientel der anbietenden Hochschultypen ebenso, und innerhalb dieser Gruppen gibt es eine hohe Heterogenität. Besonders besorgt die hohe Abbruchquote den Wissenschaftsrat. Wörtlich schreibt er: „Gemäß den wenigen vorliegenden Studien gelten als wichtigste Gründe für die hohen Abbruchquoten allerdings ungenügende Studienvorbereitung und mangelhaftes Bewältigen der Studieneinstiegsphase, auch verbunden mit falschen Erwartungen. Dementsprechend wird das Studium offenbar meist bereits in den ersten Semestern abgebrochen." Dieser Negativbefund gibt uns Lehrenden zu denken; wer sind die Menschen, die ein Studium in unserem Bereich beginnen, was können sie, was erwarten sie, wie können sie am besten motiviert werden und Kompetenzen erwerben? Und insbesondere stellt sich die Frage, wie die Verschiedenheit dieser Menschen in der Studienorganisation und in der Hochschuldidaktik berücksichtigt werden kann.

Zu diesem und zu anderen aktuellen Themen der Informatikhochschuldidaktik wurden 21 Beiträge aus 35 Einreichungen durch den Programmausschuss ausgewählt, wobei jedes eingereichte Papier wenigstens von drei Gutachtern bewertet wurde. Inhaltlich umfassen die akzeptierten Arbeiten für die Fachcommunity interessante Praxisberichte bis hin zu wissenschaftlichen Studien. Wie bei der HDI üblich, gibt es zur Tagung nur diese Pre-Proceedings. Ausgewählte Beiträge werden anschließend, ggfs. ergänzt um die Diskussionsergebnisse, in einem Sonderband der Fachzeitschrift Commentarii informaticae didacticae erscheinen.

Wir wünschen allen Leserinnen viele Anregungen beim Lesen dieses Tagungsbands und allen Tagungsteilnehmern eine fruchtvolle Diskussion!

Hagen und Berlin, September 2021

Jörg Desel, Simone Opel und Juliane Siegeris

Tagungsleitung

Conference Chair:	Jörg Desel, FernUniversität in Hagen
Organisation Chair:	Simone Opel, FernUniversität in Hagen
PC Chairs:	Jörg Desel, FernUniversität in Hagen Juliane Siegeris, HTW Berlin

Programmkomitee

- Nadine Bergner, TU Dresden
- Torsten Brinda, Universität Duisburg-Essen
- Jörg Desel, FernUniversität in Hagen (Vorsitz)
- Michael Goedicke, Universität Duisburg-Essen
- Detlef Krömker, Goethe-Universität Frankfurt
- Agathe Merceron, Beuth-Hochschule Berlin
- Johannes Magenheim, Universität Paderborn
- Simone Opel, FernUniversität in Hagen
- Barbara Paech, Universität Heidelberg
- Niels Pinkwart, Humboldt-Universität Berlin
- Axel Schmolitzky, HAW Hamburg
- Ulrik Schroeder, RWTH Aachen
- Andreas Schwill, Universität Potsdam
- Niels Seidel, FernUniversität in Hagen
- Juliane Siegeris, HTW Berlin (Vorsitz)
- Jan Vahrenhold, Westfälische Wilhelms-Universität Münster
- Karin Vosseberg, Hochschule Bremerhaven
- Karsten Weicker, HTWK Leipzig

Inhalt

Wie kann IT-Gestaltung diversity-gerechter werden? Grundlagen, Voraussetzungen und Herausforderungen für Informatik-Forschung und -Lehre

Claude Draude [1]

Informatik verändert nicht nur die Arbeitswelt, sondern zunehmend alle Lebensbereiche. Produkte, Dienste, Logistik und Infrastruktur, sowie Prozesse der Wissens-, Informations- und Kommunikationsorganisation werden maßgeblich durch IT-Systeme bestimmt. Die weltgestaltende Kraft der Informatik und die gegenseitige Durchdringung von technischer und sozialer Welt bestreitet heute niemand mehr. Hierbei wird die Informatik häufig als Innovationstreiber angesehen, aber sie steht auch zunehmend unter Druck, ihre Forschung und Entwicklungen zu überdenken und auf gesellschaftliche Herausforderungen angemessen zu reagieren [Ji17], [Sc16].

Zunehmend problematisiert wurden in den letzten Jahren IT-Entwicklungen, die diskriminierend wirken und soziale Ungleichheiten verstärken, wie z.B. rassistische oder sexistische Autovervollständigungen oder Fotoklassifizierungen bei Suchmaschinen bzw. bei der Gesichtserkennung [BG18], [No18]; inkorrekte Übersetzungssoftware [Sc11-18] oder Verzerrungen bei algorithmenbasierten Entscheidungshilfen, wie der Einschätzung von Rückfallquoten von Verurteilten [An16] oder der Vergabe von Weiterbildungsmaßnahmen auf dem Arbeitsmarkt [Fa20]. IT-Systeme sollten jedoch eine möglichst breite gesellschaftliche Teilhabe ermöglichen, demokratischen Werten und Normen gerecht werden und sozial und ökologisch nachhaltig wirken [WB19]. Vielfaltsaspekten und sozialen Ungleichheiten kommen hierbei eine besondere Bedeutung zu. Soziale Aspekte und gesellschaftliche Auswirkungen bei der IT-Gestaltung angemessen zu berücksichtigen, stellt die Forschung und Lehre in der Informatik allerdings vor verschiedene Herausforderungen. Für ein konsequentes Zusammendenken von sozialen und technischen Prozessen bildet die Informatik an Hochschulen derzeit größtenteils nicht aus. Zur Thematik gibt es langjährige Diskussionen, Ansätze und Best Practices [Go11], [Gr19].

Die Geschlechter- und Diversitäts-Forschung liefert seit Jahrzehnten wichtige Impulse für eine inklusivere, vielfältigere Informatik. Nach wie vor aktuell sind Diskussionen um Teilhabeaspekte, die Untersuchung der Fachkultur und das Bild der Informatik [Be20]. Wenn es um Verfahren diversity-gerechterer IT-Gestaltung und ihre Integration und Vermittlung in der Informatik geht, stellen sich darüber hinaus Fragen der Anwendbarkeit und Operationalisierbarkeit. Das „Gender Extended Research and Development Model" (GERD) stellt setzt an dieser Stelle an und ermöglicht es soziale Aspekte, insbesondere Gender- und Diversitäts-Aspekte, mit typischen Prozesszyklen der Informatik-Forschung und -Entwicklung zu verschränken [Ge21].

1 Universität Kassel, Fachbereich Elektrotechnik / Informatik, Pfannkuchstr. 1, Kassel, 34121, claude.draude@uni-kassel.de, https://orcid.org/0000-0001-8467-195X

Bibliografie

[An16] Angwin, J. et al.: There's software used across the country to predict future criminals. And it's biased against blacks. ProPublica, 23/05/2016, www.propublica.org/article/machine-bias-risk-assessments-in-criminalsentencing, accessed: 29/07/2021.

[Be20] Bereswill, M. et al.: Ungleiche Präferenzen? Zum Zusammenhang von Studienfachwahl und Geschlecht aus sozialisations- und geschlechtertheoretischer Perspektive am Beispiel des Studienfachs Informatik. ZSE Zeitschrift für Soziologie der Erziehung und Sozialisation 3/40, pp. 231–253, 2020.

[BG18] Buolamwini, J.; Gebru, T.: Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In (Friedler, S.A.; Wilson, C. eds.): Proceedings of the 1st Conference on Fairness, Accountability and Transparency. PMLR 81, New York, NY, USA, pp. 77–91, 2018.

[Fa20] Fanta, A.: Datenschutzbehörde stoppt Jobcenter-Algorithmus, 21/08/2020, https://netzpolitik.org/2020/oesterreich-ams-datenschutzbehoerde-stoppt-jobcenteralgorithmus/, accessed: 29/07/2021.

[Ge21] GERD – Gender Extended Research and Development Model, www.gerdmodel.com, accessed: 25/07/2021.

[Go11] Goldweber, M. et al.: Enhancing the social issues components in our computing curriculum. ACM Inroads 1/2, pp. 64–82, 2011.

[Gr19] Grosz, B. J. et al.: Embedded EthiCS. Communications of the ACM 8/62, pp. 54–61, 2019.

[Ji17] Jirotka, M. et al.: Responsible Research and Innovation in the Digital Age. Commun. ACM 5/60, pp. 62–68, 2017.

[No18] Noble, S. U.: Algorithms of oppression: How search engines reinforce racism. NYU Press. New York, 2018.

[Sc16] Schwab, K.: The Fourth Industrial Revolution: what it means, how to respond. World Economic Forum, 14/01/2016, www.weforum.org/agenda/2016/01/the-fourthindustrial-revolution-what-it-means-and-how-to-respond/, accessed: 29/07/2021.

[Sc11-18] Schiebinger, L. et al. (eds.): Gendered Innovations in Science, Health & Medicine, Engineering and Environment - Case study Machine Translation Analyzing data, 2011-2018, http://genderedinnovations.stanford.edu/case-studies/nlp.html#tabs-2, accessed:29/07/2021.

[WB19] WBGU, Politikpapier: Transformation unserer Welt im digitalen Zeitalter, 2019, www.wbgu.de/de/, accessed: 30/07/2021.

Leveraging Data Science and Social-Impact Analysis to Broaden Participation in Introductory Computer Science Courses

Kathi Fisler[1]

Abstract:
Recent trends – such as rising demand for computing courses, the emergence of Data Science as a critical skill, attention to the lack of diversity in computing workforces, and growing concerns about the social impacts of algorithmic decision-making systems – call on educators to revisit how we teach introductory computing and informatics courses. The speaker is three years into an experiment with redesigning the introductory computing course to combine data science, basic data structures, and social impacts into an introductory course meant for students across the university. The course has proven successful, attracting a diverse student population across each of gender, race, and academic interests. The talk will describe the course design, its research-based foundations, and lessons learned about addressing these trends through revitalized introductory courses.

Keywords:
Keynote; Data Science; Introductory Computing; Social Impact

1 Brown University, Providence, Rhode Island, USA; kathryn_fisler@brown.edu

Informatik und Barrierefreiheit

Gerhard Weber[1]

Abstract:
Barrierefreiheit fördert die Inklusion und ist damit eine der gesellschaftlichen Auswirkungen die von Informatikern und Informatikerinnen gestaltet wird. In der Praxis begegnet man der Anforderung „aber bitte barrierefrei" im Rahmen einer Webentwicklung und wenn es gut geht, ist die Prüfung auf Barrierefreiheit durch einen externen Dienstleister erfolgreich. Da nur etwa 5Der Beitrag greift drei Aspekte der Informatikausbildung auf. Es werden einige der notwendigen Kompetenzen vorgestellt, die in der Entwicklung von barrierefreien Benutzeroberflächen relevant sind. Am Beispiel von Mobiltelefonen wird gezeigt, dass Apps deutlich weniger Menschen ausschließen, wenn diese keine Webtechnologien verwenden und dafür die integrierten assistiven Technologen unterstützten. Barrierefreiheit greift nicht nur in grafische Benutzeroberflächen ein, sondern auch ins Betriebssystem, berührt Datensicherheit, die Softwaretechnik, das Requirements Engineering, KI und viele weitere Fachgebiete der Informatik. Einige Beispiele für diese Bezüge sollen verdeutlichen, dass die Informatikausbildung noch verbessert werden kann, so dass Software die Prüfung auf Barrierefreiheit leichter besteht. Der dritte Aspekt beleuchtet die Lehre selbst. E-Learning ist aktuell in breitem Einsatz, aber Lehrende erzeugen dabei Barrieren z.B. durch fehlende Untertitel in Videokonferenzen oder durch Einsatz digitaler Pinnwände. Der Beitrag gibt einen kurzen Überblick zur Gestaltung barrierefreier Lehrmaterialien und zum barrierefreien Prüfen.

Keywords:
Keynote; Mensch-Computer-Interaktion; Barrierefreiheit; Assistive Systeme

1 Technische Universität Dresden, Professur für Mensch-Computer Interaktion, Nöthnitzer Straße 46, 01187 Dresden; gerhard.weber@tu-dresden.de

“Es ist kompliziert, alles inklusive Privatleben unter einen Hut zu bekommen”

Eine Studie zu Nutzen und Schaden von Arbeitsverhältnissen für das Informatikstudium

Timon Schell[1], Andreas Schwill[2]

Abstract:
Eine übliche Erzählung verknüpft lange Studienzeiten und hohe Abbrecherquoten im Informatikstudium zum einen mit der sehr gut bezahlten Nebentätigkeit von Studierenden in der Informatikbranche, die deutlich studienzeitverlängernd sei; zum anderen werde wegen des hohen Bedarfs an Informatikern ein formeller Studienabschluss von den Studierenden häufig als entbehrlich betrachtet und eine Karriere in der Informatikbranche ohne abgeschlossenes Studium begonnen. In dieser Studie, durchgeführt an der Universität Potsdam, untersuchen wir, wie viele Informatikstudierende neben dem Studium innerhalb und außerhalb der Informatikbranche arbeiten, welche Erwartungen sie neben der Bezahlung damit verbinden und wie sich die Tätigkeit auf ihr Studium und ihre spätere berufliche Perspektive auswirkt. Aus aktuellem Anlass interessieren uns auch die Auswirkungen der Covid-19-Pandemie auf die Arbeitstätigkeiten der Informatikstudierenden.

Keywords:
Informatikstudium, Studienabbrecher, Studentenjobs, Studiendauer.

1 Einleitung

Nur rund 40,3% aller Studierenden schlossen 2016 das Bachelorstudium in der Regelstudienzeit ab [SB18], der Median der Studiendauer von Bachelor-Erstabsolventen in Deutschland im Prüfungsjahr 2016 betrug 7,0 Semester [SB18]. In der Informatik wichen die Verhältnisse noch einmal deutlich nach unten ab: Die Studiendauer eines Bachelor-Studiums Informatik betrug 2016 im Mittel an deutschen Universitäten 8,2 Semester, und nur 33% schafften es in der Regelstudienzeit. Im Masterstudium Informatik gelang das sogar nur noch 20,2% [SB18]. 44% aller Informatikstudierenden brechen das Studium vor dem Abschluss ab [HRS20].

Eine übliches Erklärungsmuster gegenüber Universitätsleitungen, Ministerien oder Außenstehenden für die beiden eher unvorteilhaften Zahlen bezieht studentische Arbeitstätigkeit und großartige Berufsaussichten ein:

- die verlängerte Studiendauer von Informatikstudierenden hänge mit der sehr gut bezahlten Nebentätigkeit von Studierenden in der Informatikbranche zusammen, die vom Studium ablenke und den Studienfortschritt bremse;

- die hohe Abbrecherquote sei auch eine Folge des hohen Bedarfs an Informatikern, so dass Studierende oft schon vor dem Abschluss attraktive Jobangebote in der Informatikbranche erhalten, einen formellen Studienabschluss dann als entbehrlich empfinden und die Hochschule vorzeitig verlassen.

Zumindest die erste Hypothese ist stark zu bezweifeln. Nach Workload-Analysen von R. Schulmeister im Informatikstudiengang an der Universität Paderborn [Sc15] verrichtete die Mehrheit der Studierenden keine bezahlte Arbeit, nur wenige (22%) übten gelegentlich oder einmalig einen Job für wenige Stunden aus, und nur 14% der Probanden haben über acht Stunden pro Woche gearbeitet. Zudem stellte Metzger [Me13] in einer allgemeineren fachübergreifenden Studie fest, dass nicht jeder, der viel jobbt, wenig Zeit in sein Studium investiert (die Korrelation zwischen Workload und der für Erwerbstätigkeit aufgewendeten Zeit war mit

1 Universität Potsdam, Institut für Informatik, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, timon.schell@uni-potsdam.de,

2 Universität Potsdam, Institut für Informatik, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, schwill@cs.uni-potsdam.de

-0,14 gar negativ), und auch nicht jeder, der viel Zeit für Studium und Erwerbstätigkeit aufbringt, fühlt sich automatisch hoch belastet. Vielmehr scheinen die Einstellung zum Studium bzw. zum Job sowie die Motivation das subjektive Belastungsempfinden zu prägen.

In dieser Arbeit gehen wir diesen und weiteren Fragestellungen nach. Anhand einer Befragung erhoben wir die Studienplanungen arbeitender und nichtarbeitender Studierender im Bachelor- und Masterstudium Informatik, die Arbeitsschwerpunkte innerhalb und außerhalb der Informatikbranche, die allgemeine Zufriedenheit mit der Tätigkeit sowie den erwarteten Beitrag der Jobs für Studium und späteren Beruf. Inwieweit die hohe Abbrecherquote auf attraktive Jobangebote vor Studienabschluss zurückzuführen ist, konnte in dieser Studie jedoch nicht geklärt werden, da die Universität Potsdam keine Nachverfolgung von Abbrechern betreibt.

Aus aktuellem Anlass waren wir auch interessiert zu erfahren, welchen Einfluss die Corona-Pandemie auf Jobs von Informatikstudierenden ausgeübt hat. Hier wurde in 2020 vielerorts u.a. berichtet, dass 40% der Studierenden ihren Job verloren haben [Ra20] und dass sich die Zahl der Stellenangebote für Studierende im September 2020 gegenüber dem Vorjahr halbiert hat, während die Nachfragen um etwa 50% gestiegen sind [He20, Wo20].

Im Sommersemester 2011 führten wir eine ähnliche Befragung am Institut für Informatik der Universität Potsdam mit 119 Teilnehmern durch [IfI11]. Wegen der damals noch verbreiteten Studiengänge Diplom und Magister sind nicht alle Ergebnisse vergleichbar. An den Stellen, an denen es geeignet und möglich erscheint, setzen wir die damaligen Ergebnisse zu den heutigen in Beziehung.

2 Forschungsfragen und Methodik

Ausgehend von diesen Vorüberlegungen formulierten wir folgende Forschungsfragen:

1. Welche Studiendauer planen Studierende im Bachelor- und Masterstudium Informatik und mit welchen Begründungen planen sie über die Regelstudienzeit hinausgehende Studiendauern? – Mit dieser Frage sollte geklärt werden, inwieweit die über die Regelstudienzeit hinausgehende Studienzeit auf persönlichen Entscheidungen der Studierenden beruht, die bereits zu Beginn oder im Laufe des Studiums getroffen werden. Solch eine Entscheidung sollte respektiert werden und kann nicht der Studiengangsorganisation angelastet werden. Zudem geben die Ergebnisse Hinweise darauf, ob Studierende mit oder ohne Nebentätigkeit unterschiedliche Studienplanungen verfolgen.

2. In welchen Branchen arbeiten Studierende? – Hier war für uns nur relevant, ob es sich um Tätigkeiten innerhalb oder außerhalb der Informatikbranche handelt.

3. Fühlen Studierende ihr Studium und ihre späteren Berufsaussichten durch das Arbeitsverhältnis eher befördert oder behindert? – Wenn – insbesondere durch Arbeit in der Informatikbranche – für das Studium selbst oder die spätere Berufstätigkeit Vorteile erzielt oder erwartet werden, spricht eigentlich nichts gegen eine studentische Nebentätigkeit, selbst wenn sich dadurch das Studium verlängert.

4. Wie zufrieden sind Studierende mit ihrem Arbeitsverhältnis? – Wir fragten nach der Entlohnung, sicher ein wesentliches Merkmal eines befriedigenden Studentenjobs, sowie nach der allgemeinen Zufriedenheit, die wir nicht in weitere Merkmale aufschlüsselten.

5. Welchen Einfluss hat die Corona-Pandemie auf studentische Arbeitsverhältnisse? – Angesichts der schon in Abschnitt 1 genannten Auswirkungen auf die Stellenangebote und -nachfragen von Studentenjobs und die Arbeitsverhältnisse selbst waren wir hier an den Wirkungen auf Informatikstudierende interessiert.

Die Umfrage wurde vom 8.12.2020 bis zum 15.1.2021 über das Befragungstool der Universität Potsdam unter survey.uni-potsdam.de zur Verfügung gestellt. Die Einladung zur Umfrage erfolgte über den Mailverteiler des Instituts für Informatik der Universität Potsdam und erging an die insgesamt 769 Studierenden aller deutschsprachigen Studiengänge des Instituts, das sind der Bachelorstudiengang Informatik/Computational Science, der Masterstudiengang Computational Science, die Lehramtsstudiengänge Informatik im Bachelor und Master sowie auslaufende Informatikstudiengänge.

3 Auswertung und Ergebnisse

Wir erhielten N=98 Rückläufe (12,7%), davon 73% von männlichen, 24% von weiblichen Studierenden. 70 Studierende, davon 19% weiblich, befanden sich in einem Bachelorstudiengang, 28 Studierende, davon 39% weiblich, in einem Masterstudiengang. Die Verteilung der Teilnehmer über die Studiengänge zeigt Tab. 1.

Tab. 1: Verteilung der Teilnehmer über die Studiengänge (N=98)

Studiengänge	absolut	anteilig
Bachelor Informatik/Computational Science	56	57%
Master Informatik/Computational Science	24	24%
Bachelor Lehramt Informatik	14	14%
Master Lehramt Informatik	4	4%
Andere	0	0%

Zwei Befragte mit der Geschlechtsangabe „divers" bzw. „andere" werden in den folgenden Auswertungen dann nicht weiter berücksichtigt, wenn geschlechtsspezifische Aussagen getroffen werden.

31 der 70 Bachelorstudierenden (44,3%) befanden sich im ersten Semester, 50 (71,4%) in den ersten drei Semestern, von den 28 Masterstudierenden sind vier (14,3%) im ersten Semester und neun (32,1%) in den ersten drei Semestern. Das höchste angegebene Fachsemester stammte aus dem Bachelor und lag bei 15. Im Mittel befanden sich die Studierenden im Bachelor im Semester 3,3 und im Master im Semester 5,6.

Tab. 2: Geplante Semester bis zum Bachelor- (N=70) bzw. Masterabschluss (N=87) (Angaben im Master bereinigt um Fälle mit Angabe "0" und "30" Semester)

Abschluss geplant im ...	Bachelor absolut	Bachelor in %	Master absolut	Master in %
3. Semester	0	0%	3	3%
4. Semester	1	1%	43	49%
5. Semester	0	0%	11	13%
6. Semester	25	36%	17	20%
7. Semester	13	19%	1	1%
8. Semester	18	26%	8	9%
9. Semester	3	4%	2	2%
≥ 10. Semester	10	14%	2	2%
Mittelwert	7,6	--	5,1	--

3.1 Studienplanung

Bachelor

Etwa ein Drittel der Studierenden (36%) plant, das Bachelorstudium in Regelstudienzeit von sechs Semestern abzuschließen. Im Mittel planen die 70 Bachelorstudierenden 7,6 Semester für das Bachelorstudium ein, also 1,6 Semester oberhalb der Regelstudienzeit (Tab. 2). Dies entspricht recht genau der mittleren Studiendauer von 7,4 Semestern im Bachelorstudium Informatik bundesweit [ISA19].

Interessant ist hier ein Vergleich der Planungen arbeitender und nicht-arbeitender Studierender: Erstere planen im Mittel 8,3 Semester, letztere 7,2 Semester bis zum Abschluss des Bachelorstudiums ein.

Master

Fast die Hälfte der Studierenden (49%) plant, das Masterstudium in Regelstudienzeit von vier Semestern zu beenden. Drei Studierende beabsichtigen sogar, das Studium vorzeitig abzuschließen. Im Mittel werden 5,1 Semester für das Masterstudium eingeplant, also 1,1 Semester über der Regelstudienzeit (Tab. 2). Zehn Teilnehmer gaben an, Null geplante Mastersemester zu benötigen, was wir als Absicht interpretieren, nach dem Bachelorstudium nicht weiter studieren zu wollen. In einem Fall wurde die Eingabe von 30 Semestern als Fehleingabe gewertet und aus der Auswertung gestrichen.

Auch im Masterstudium weichen die Planungen arbeitender und nicht-arbeitender Studierender voneinander ab: Arbeitende Studierende planen im Mittel 5,7 Semester, nicht arbeitende 4,8 Semester bis zum Abschluss des Masterstudiums ein. Wie im Bachelor unterscheiden sich die Planungen der Studiendauern der beiden Gruppen also um etwa ein Semester. Bachelor- und Masterstudium zusammen sorgen dann insgesamt für ein um etwa zwei Semester verlängertes Studium von arbeitenden gegenüber nicht arbeitenden Studierenden, eine durchaus akzeptable Verlängerung des Studiums, insbesondere wenn im Rahmen der Arbeitstätigkeit, speziell in der Informatikbranche, zusätzliche Qualifikationen erworben werden.

Angestrebte Leistungspunkte

Zur Absicherung der Ergebnisse aus der Studienplanung fragten wir auch nach der im Wintersemester 2020/2021 angestrebten Zahl von Leistungspunkten (LP). Abweichungen von der Planzahl 30 LP lassen ebenfalls Schlüsse auf die Studienintensität zu.

Knapp die Hälfte der Studierenden (48%, 2011: 52%) strebt die vorgesehene Anzahl an LP oder mehr an (Tab. 3). Im Mittel versuchen die Teilnehmer, 24,7 LP (2011: 18,1 SWS=27,2 LP) zu erzielen. Zugleich erkennt man eine recht große Spannbreite von 5 bis 51 LP in den Planungen. Ein Fall, der Null zu erzielende Leistungspunkte angab und sich am Ende des Studiums befindet, wurde aus der Analyse entfernt.

Tab. 3: Angestrebte Leistungspunkte im WS 2020/2021 (N=97)

LP	5	6	8	12	14	18	21	22	24	27	28	30	32	33	36	48	51
Fälle	1	3	2	7	1	14	3	1	16	1	1	36	1	4	4	1	1

Vergleicht man auch hier wieder die angestrebten Leistungspunkte bei arbeitenden (23 LP) und nicht arbeitenden Studierenden (26 LP), so erkennt man im Mittel eine Differenz von drei LP. Schreibt man diese Angabe für das aktuelle Semester auf das gesamte Studium fort, so kumuliert sich bis zum Ende der Regelstudienzeit bei arbeitenden Studierenden ein Defizit von etwa 20 LP im Bachelor und weiteren 12 LP im Master, das in jeweils einem weiteren Semester aufgeholt werden muss. Diese Tatsache steht im Einklang mit den Aussagen zur Studienplanung der Studierenden im vorherigen Abschnitt.

Gefragt nach den Gründen für eine angestrebte LP-Zahl unter der "Marschzahl" 30 LP, antworteten die Teilnehmer in unterschiedlicher Weise (Tab. 4).

Neben dem Studium zu arbeiten gaben 41% der Studierenden als Grund an (2011: 32%). Bereits am Ende des Studiums befinden sich 19% der Teilnehmer, die nur noch wenige Module absolvieren müssen. Sich nur einen losen Zeitplan für das Studium vorzugeben und daher weniger als 30 LP studieren zu wollen, trifft nur auf 16% (2011: 15%) der Studierenden zu. Insgesamt wird das Studium also von keinem Ereignis so stark verzögert wie von einem Studentenjob, dem man nachgeht.

Bei dieser Frage gab es auch die Möglichkeit der Freitextantwort. Die Analyse der insgesamt 14 Freitexte ergab, dass einige Studierende für ein Modul sehr viel Aufwand betreiben und daher nicht alle vorgesehenen Module belegen können, dass einige sich am Ende des Studiums befinden und weniger Module belegen müssen, dass für einige Arbeit und Freizeit Vorrang haben und dass die Corona-Pandemie und Stress dazu beigetragen haben, die Studienintensität zu reduzieren.

Tab. 4: Begründungen für reduzierte Studienintensität im WS 2020/2021 (N=98)

Gründe (Mehrfachantworten möglich)	absolut	anteilig
Weil ich die von der Prüfungsordnung vorgesehene Anzahl an LP bereits anstrebe.	40	41%
Weil ich neben dem Studium arbeite.	40	41%
Weil mein Studium fast beendet ist und ich in diesem Semester nicht mehr so viele Veranstaltungen besuchen muss.	19	19%
Weil ich mir keinen festen Zeitrahmen für mein Studium gesetzt habe und die Studienzeit frei gestalten möchte.	16	16%
Weil ich aus gesundheitlichen Gründen nicht mehr bewältigen kann.	11	11%
Weil es aufgrund von Terminkollisionen nicht anders möglich war.	9	9%
Weil ich Familienangehörige versorge.	3	3%
Sonstiges	14	14%

3.2 Studium und Arbeit

Studentenjobs und Branche
40 Teilnehmer bzw. ca. 41% der Befragten gaben an, neben dem Studium arbeitstätig zu sein, im Bachelor ist es etwa ein Drittel der Befragten (36%), im Master mehr als die Hälfte (54%). Von den Arbeitstätigen besitzen ca. 65% mindestens einen Job in der Informatikbranche. Die Ergebnisse zur sehr geringen Arbeitstätigkeit Informatikstudierender aus der Studie von Schulmeister [Sc15] können wir nicht bestätigen.

Studierende nehmen offenbar nicht erst in einem späteren Abschnitt ihres Studiums eine Arbeitstätigkeit auf, denn 28% der arbeitenden Studierenden befinden sich erst im ersten Semester.

Beim Übergang vom Bachelor zum Master verdoppelt sich in etwa der Anteil derjenigen, die in der Informatikbranche arbeiten von 45% auf 87%. Möglicherweise suchen Studierende im Master mit Blick auf ihre zukünftige Berufsperspektive bevorzugt nach Jobs in der Informatikbranche; zugleich genügt Arbeitgebern der Bachelorabschluss offenbar vielfach als Einstellungsqualifikation.

Vier Befragte im Bachelor gaben an, sowohl außerhalb als auch innerhalb der Informatikbranche zu arbeiten. Dieser Sachverhalt kommt bei Masterstudierenden nicht vor. Interessanterweise gaben diese vier Fälle zu einer späteren Frage an, eher unzufrieden mit der aktuellen Entlohnung zu sein (Mittelwert 3,5 auf der Notenskala 1 bis 6). Möglicherweise ist die Mehrfachbeschäftigung gerade diesen finanziellen Engpässen geschuldet.

Gegenüber der Studie aus dem Sommersemester 2011 [IfI11] hat sich die Arbeitsintensität der Studierenden deutlich erhöht, zugleich ist sie informatikspezifischer geworden. Damals gingen nur etwa 32% der 119 Befragten neben dem Studium einer Arbeit nach. Das bedeutet einen Anstieg von fast 10% bis 2020. Dabei steigen sowohl der Anteil der Studierenden, welche in der Informatikbranche tätig sind, von 20% auf 27%, als auch der Anteil der außerhalb der Informatik Tätigen von 15% auf 18%. Von den Arbeitstätigen arbeiteten damals 63% in der Informatikbranche und damit ziemlich genau so viele wie heute. Es scheint also besonders im Informatikbereich mehr Unternehmen zu geben, die Studierende einstellen, allerdings können auch saisonale Unterschiede zwischen Sommer und Wintersemester eine Rolle spielen.

Arbeitszufriedenheit
Zu diesem Thema wurden drei Items abgefragt, die wir hier nacheinander analysieren. Die Bewertung erfolgte nach Schulnoten.

Auf die Frage „Glauben Sie, dass Ihre derzeitige Tätigkeit, nach dem Studienabschluss für die Berufstätigkeit nützlich ist?" beurteilten 40% der Fälle, dass ihre aktuelle Tätigkeit sehr vorteilhaft für die Berufstätigkeit nach Studienabschluss ist (Tab. 5). In 2011 waren es etwa 37%. Im Mittel wurde die Note 2,3 (2011: 2,8) vergeben. Ein Vergleich nach Gruppen (Tab. 6) zeigt ein ähnliches Bild. Hierbei sticht hervor, dass Frauen den Einfluss der aktuellen Tätigkeit von allen Gruppen am besten und im Mittel um eine Note besser einschätzen als Männer. Studierende, die in der Informatikbranche arbeiten, beurteilen erwartungsgemäß im Mittel den Einfluss der aktuellen Tätigkeit auf die zukünftige Tätigkeit nach Studienabschluss um eine Note besser als Studierende außerhalb der Informatik. 2011 war hier der Unterschied noch deutlicher, denn die Nützlichkeit eines Jobs außerhalb der Informatikbranche erhielt damals im Mittel nur die Note 4,4. Da, wie im vorherigen Abschnitt gezeigt, anteilig viel mehr Masterstudierende in der Informatikbranche arbeiten, schneidet deren Arbeitszufriedenheit auch fast eine Note besser ab als bei Bachelorstudierenden.

Tab. 5: Nützlichkeit des Studentenjobs für die spätere Berufstätigkeit (N=40)

Note	absolut	anteilig
1=sehr stark	16	40%
2	11	28%
3	5	13%
4	3	8%
5	4	10%
6=gar nicht	1	3%
Mittelwert	2,3	--

Tab. 6: Nützlichkeit der Berufstätigkeit für verschiedene Gruppen (Notenskala 1-6)

Kriterium	Schulnote	Schulnote	Kriterium
weiblich	1,5	2,5	männlich
Arbeit in Informatikbranche	1,8	2,8	Arbeit außerhalb Informatikbranche
Bachelor	2,6	1,7	Master
Lehramt	2,7	2,2	Informatik//CS

Im zweiten Item fragten wir: „Wie gut wird die Arbeit neben dem Studium entlohnt?" 78% der Fälle empfinden die aktuelle Entlohnung als eher positiv (Noten 1-3). In einem Drittel der Fälle wurde die aktuelle Entlohnung als gut bewertet, 20% hielten sie für sehr gut. Im Mittel wurde die Entlohnung mit Note 2,6 bewertet. Interessanterweise zeigt der Vergleich nach Gruppen nur leichte Unterschiede: Studierende, die innerhalb der Informatikbranche arbeiten, fühlen sich nicht besser entlohnt als ihre Kommilitonen, die außerhalb arbeiten.

Das dritte Item lautete: „Welche Höhe der Entlohnung erwarten Sie nach dem erfolgreichen Studienabschluss?" Hier sind die Erwartungen sehr einheitlich: 90% aller Fälle erwarten eine sehr gute oder gute Entlohnung und der Mittelwert der Notenskala 1 bis 6 liegt bei 1,7. In einzelnen Teilgruppen von Studierenden erkennt man nur marginale Unterschiede (Tab. 7). Die einzige Gruppe, die hier etwas heraussticht, sind die Studierenden des Lehramts, welche mit Note 2,1 die zukünftige Bezahlung etwas schlechter sehen, als die Gruppe der Informatikstudierenden, eine sicher realistische Einschätzung angesichts der Diskussionen um Gehälter, Besoldung und Verbeamtung.

Tab. 7: Erwartete Bezahlung nach Studienabschluss für verschiedene Gruppen (Notenskala 1-6)

Kriterium	Schulnote	Schulnote	Kriterium
weiblich	1,6	1,7	männlich
Arbeit in Informatikbranche	1,7	1,7	Arbeit außerhalb Informatikbranche
Bachelor	1,7	1,6	Master
Lehramt	2,1	1,5	Informatik//CS

Insgesamt überwiegen bei allen drei Items die positiven oder neutralen Antworten. Es zeigt sich deutlich, dass zwei Drittel der Befragten annehmen, die aktuelle Tätigkeit wäre günstig für die zukünftige Berufstätigkeit. Die Hälfte der Fälle findet die aktuelle Bezahlung gut, und 90% erwarten eine gute Entlohnung nach Studienabschluss.

3.3 Studentenjobs in Zeiten der Corona-Pandemie

"Hat die Covid-19-Pandemie etwas an Ihrem Arbeitsverhältnis geändert?" fragten wir die Teilnehmer. Die uns sehr überraschenden Ergebnisse zeigt Tab. 8.

Tab. 8: Auswirkungen der Corona-Pandemie auf die studentische Arbeitstätigkeit (N=40)

Ereignis infolge Corona-Pandemie	absolut	anteilig
Nein.	32	80%
Ich arbeite nun mehr Stunden als zuvor.	7	18%
Ich habe die Arbeitsstelle gewechselt und bin weiterhin außerhalb der Informatikbranche tätig.	2	5%
Ich arbeite nun weniger Stunden als zuvor.	1	3%
Ich habe die Arbeitsstelle gewechselt und bin weiterhin im informatikbezogenen Bereich tätig.	0	0%
Ich habe die Arbeitsstelle gewechselt und arbeite nun in einem informatikbezogenen Beruf.	0	0%
Ich habe die Arbeitsstelle gewechselt und arbeite nun außerhalb der Informatikbranche.	0	0%
Ich habe meine Arbeitsstelle verloren.	0	0%

Wider alle Erwartungen scheint Covid-19 kaum Einfluss auf die Arbeitsverhältnisse der Studierenden ausgeübt zu haben. 80% der Fälle gaben an, dass die Pandemie keine Änderungen mit sich gebracht hat. 18% der Fälle äußerten sogar, nun mehr Stunden zu arbeiten als vorher. Zwei Studierende haben ein Arbeitsverhältnis außerhalb der Informatikbranche gegen ein anderes getauscht, und nur in einem Fall wurden die Stunden reduziert. Tatsächlich war eher zu vermuten, dass Studierende ihre Arbeit verloren haben, denn zumeist können Studentenjobs in einer wirtschaftlichen Notlage am einfachsten beendet werden. Andererseits hat sich die Informatikbranche, die sich leichter als andere Branchen auf das Homeoffice umstellen lässt und sich auch gegenüber anderen Einflüssen in der Regel robuster darstellt, die Einschränkungen der Pandemie offenbar besser bewältigt, so dass die Arbeitsverhältnisse der Studierenden praktisch nicht beeinflusst wurden. Die geringen Veränderungen bei den Arbeitszeiten und die zwei Beschäftigungsänderungen können auch bei normalen wirtschaftlichen Verhältnissen vorkommen.

3.4 Freitextantworten zum Verhältnis von Studium und Arbeit

In der letzten Frage "Haben Sie Anmerkungen in Bezug auf die Verbindung zwischen Studium und Arbeit?" konnten die Teilnehmer etwaige Sachlagen, welche in den vorherigen Fragen nicht beleuchtet wurden, frei äußern. Hierbei kristallisierten sich einige Prioritäten der arbeitenden Studierenden heraus. Einige Studierende benötigen die Arbeit zwingend zur Finanzierung und ordnen daher dem Studium eine geringere Priorität zu. Damit einher geht die Problematik, Arbeit und Studium zu koordinieren, und der Wunsch nach mehr Flexibilität. Hier wurde mehrfach die Distanzlehre seit Beginn der Corona-Pandemie positiv hervorgehoben, die es nun oft erlaubt, Studium und Arbeit im Home-Office zu erledigen, verbunden mit der Anregung, diese Lehrform auch in Zukunft optional anzubieten.

4 Zusammenfassung

Informatikstudierende planen ihr Studium weit überwiegend relativ straff und gestehen sich für den Bachelor und den Master nur jeweils etwa eineinhalb Semester zusätzlich zu. Studierende, die sich keinen festen Zeitrahmen gesetzt haben, bilden nur eine geringe Minderheit. Arbeitende Studierende erwarten dabei, etwa ein Semester länger zu studieren als nicht-arbeitende. Informatikfachbereiche hätten es in der Hand, diesen Studierenden mit einer gewissen Anrechnung geeigneter Nebentätigkeiten durch Leistungspunkte entgegen zu kommen, um die Studienzeitverlängerung etwas zu kompensieren. Möglich wäre z.B.die Anrechnung im Rahmen von Schlüsselkompetenzen, zumal die Nebentätigkeit von den Studierenden als förderlich für Studium und Beruf angesehen wird und viele Schlüsselkompetenzen, die in den Verzeichnissen der Universitäten gelistet sind, kaum einen höheren Bildungswert besitzen.

Die Zahl der Studierenden mit Nebenjobs hat im Vergleich zur Umfrage vom Sommersemester 2011 [IfI11] zugenommen. Gut ein Drittel der Bachelorstudierenden und mehr als die Hälfte der Masterstudierenden arbeiten nun neben dem Studium, dabei verschiebt sich der Anteil der Jobs in der Informatikbranche im Bachelorstudium von knapp 50% auf knapp 90% im Masterstudium. Masterstudierende sind also fast doppelt so häufig in der Informatikbranche beschäftigt als ihre Kommilitonen im Bachelor. Das erscheint naheliegend, da sie aufgrund ihres akademischen Werdegangs bereits einige Fähigkeiten erworben haben, welche für Unternehmen interessant sein könnten. Der überwiegende Teil der Studierenden beurteilt ihren Nebenjob als nützlich für die spätere Berufstätigkeit, das gilt noch mehr für diejenigen, die in der Informatikbranche arbeiten. In der Studie von 2011 war das noch ganz anders, als Nebenjobs außerhalb der Informatikbranche mit einer Note von 4,4 kaum noch als förderlich angesehen wurden.

Mit ihrer Entlohnung sind die Studierenden allgemein zufrieden, wobei hier kaum ein Unterschied zwischen der Informatikbranche und der übrigen Arbeitswelt erkennbar ist. Möglicherweise hat die Gesetzgebung zum Mindestlohn inzwischen dafür gesorgt, dass einfache Arbeiten ähnlich gut bezahlt werden wie höherwertige Jobs in der Informatikbranche.

Covid-19 scheint auf die Arbeitsverhältnisse der Studierenden kaum Einfluss gehabt zu haben. Bis auf ein paar Verschiebungen bei der Anzahl der Arbeitsstunden ist hier nichts Auffälliges festzustellen.

Fernlehre in Corona-Zeiten kommt einigen arbeitenden Studierenden entgegen und ermuntert sie z.T., mehr Leistungspunkte zu absolvieren, weil sich Studium und Arbeit besser organisieren lassen. Hier wäre allgemein interessant zu untersuchen, ob durch die Fernlehre Studierende ihr Studium schneller absolvieren und welche Einflüsse das auf ihre Ergebnisse hat. Wird die Arbeit neben dem Studium durch Fernlehre begünstigt? Sorgt Fernlehre für einige Studierende überhaupt erst dafür, eine Nebenbeschäftigung aufzunehmen? Bei der Rückkehr zur traditionellen Studienorganisation nach der Corona-Pandemie sollte die Situation arbeitender Studierender jedenfalls sorgfältig mitbedacht werden.

Nach wie vor ungeklärt ist, in welchem Umfang der Studienabbruch in Informatik auf attraktive Jobangebote aus der Informatikbranche zurückzuführen ist. Mangels intensiver Nachverfolgung von Abbrechern (und auch von Absolventen), die z.T. aus datenschutzrechtlichen Gründen auch gar nicht möglich ist, bleiben entsprechende Erzählungen weiterhin spekulativ.

Literaturverzeichnis

[He20] Hering, A.: Studi-Jobs und Praktika: Student:innen als große Verlierer:innen der Corona-Krise, indeed, www.hiringlab.org/de/blog/2020/10/22/corona-krise-studenten/, Stand: 01.05.2021.

[HRS20] Heublein, U.; Richter, J.; Schmelzer, R.: Die Entwicklung der Studienabbruchquoten in Deutschland. (DZHW Brief 3|2020). Hannover: DZHW, https://doi.org/10.34878/2020.03.dzhw_brief, Stand 01.05.2021.

[IfI11] Institut für Informatik: Umfrage zum Teilzeitstudium, unveröffentlichtes Manuskript, Potsdam, 2011.

[ISA19] Informationssystem Studienwahl & Arbeitsmarkt, Universität Duisburg-Essen, www.uni-due.de/isa/fg_naturwiss/informatik/informatik_hs_frm.htm, Stand: 01.05.2021.

[Me13] Metzger, C.: Zeitbudgets zur Untersuchung studentischer Workload als Baustein der Qualitätsentwicklung, Zeitschrift für Hochschulentwicklung ZFHE 8,2 (2013) 138-156.

[Ra20] Rahaus, H.: Weniger Kontakte, kaum Jobs: Die Corona-Kluft bei Studierenden, Frankfurter Allgemeine Zeitung, 28.12.2020, www.faz.net/aktuell/karriere-hochschule/hoersaal/die-corona-kluft-bei-studenten-weniger-kontakte-und-kaum-jobs-17115763.html, Stand: 01.05.2021.

[SB18] Statistisches Bundesamt: Hochschulen auf einen Blick, Ausgabe 2018, www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Bildung-Forschung-Kultur/Hochschulen/Publikationen/Downloads-Hochschulen/broschuere-hochschulen-blick-0110010187004.pdf, Stand: 01.05.2021.

[Sc15] Schulmeister, R.: Ergebnisse einer Zeitbudget-Erhebung im Fach Informatik der Universität Paderborn, 2015, https://cs.uni-paderborn.de/fileadmin/informatik/Studium/Studienelemente/Zeitbudget-Studie/Abschlussbericht_Schulmeister.pdf, Stand: 04.05.2021.

[Wo20] Wolter, U.: Wegen Corona-Krise nur halb so viele Studentenjobs wie 2019, Personalwirtschaft, www.personalwirtschaft.de/recruiting/artikel/wegen-corona-krisebricht-jobangebot-fuer-studenten-ein.html, Stand: 01.05.2021.

Studienverkürzung durch Anerkennung von Kompetenzen

Eine Interviewstudie über Vorqualifikationen von Informatikstudierenden[1]

Carsten Thorbrügge[2], Jörg Desel[3], Len Ole Schäfer[4]

Abstract:
In diesem Beitrag wird eine Interviewstudie vorgestellt, in der Kompetenzen von Informatikstudierenden aus Vorqualifikationen in Bezug zu Anerkennungsoptionen gesetzt werden. Aufgeführt werden: die pauschale Anerkennung, die auf erworbenen Zertifikaten beruht, die individuelle Anerkennung, bei der individuell erworbene Kompetenzen nachgewiesen werden, und die Adaption von Lernwegen, die Teilkompetenzen der Studierenden berücksichtigt. Jede Anerkennungsoption führt zu einer Verkürzung des Studiums und zu einer besseren Passung zum individuellen Kompetenzprofil. In der Studie wurden fachliche und überfachliche Kompetenzen für ein Sample von Informatikstudierenden mit Vorqualifikation als Fachinformatiker/in erhoben und den Anerkennungsoptionen zugeordnet. Methodisch wurden leitfadengestützte Interviews eingesetzt. Die Interviewdaten wurden transkribiert und inhaltsanalytisch ausgewertet. Die Ergebnisse zeigen, dass die Befragten aus ihrer Vorqualifikation neben erwartbaren fachlichen Kompetenzen auch Selbststeuerungskompetenzen mitbringen. Diese typischen Kompetenzmuster lassen sich den drei Anerkennungsoptionen zuordnen und bestätigen so die getätigte Differenzierung der Anerkennungsoptionen. Die vorliegende Studie schafft zudem eine empirische Basis, die eine präzisere Gestaltung von Anerkennungsprozessen für Informatikstudierende ermöglicht.

Keywords:
Durchlässigkeit; Anerkennung; Studiengestaltung; Berufsausbildung; Informatik B.Sc.; Fachinformatik

1 Einleitung

Universitäten im 21. Jahrhundert stehen vor vielfältigen Herausforderungen. Die Verwertung von im Beruf erworbenen Kompetenzen gilt als transformative Aufgabe von Hochschulen (vgl. [HB12, S. 11]). Personen, die eine Ausbildung abgeschlossen haben, weisen beruflich erworbene Kompetenzen auf, die sie in ihr Studium einbringen (vgl. [BH16, S. 8]). Die aktuell implementierten Anrechnungsverfahren[5] berücksichtigen die Kompetenzen aus dem nicht-akademischen Bereich nicht hinreichend (vgl. [Ha14, S. III]).

Die Förderung von Bildungsdurchlässigkeit stellt ein gesellschaftspolitisches Ziel dar (vgl. [EN20]), welches von der FernUniversität in Hagen mit ihrem Schwerpunkt auf berufstätigen Studierenden in besonderer Weise adressiert wird. Hierauf fußt ein 2019 initiiertes Forschungs- und Gestaltungsprojekt, das den Anstoß für mehr Anerkennung insbesondere beruflicher Vorqualifikationen geben soll. Im Rahmen einer ersten Implementierung werden Kompetenzen, die Fachinformatiker/innen bereits in der Ausbildung erworben haben, im Studiengang Informatik B.Sc. anerkannt. Es wurden drei Anerkennungsoptionen eingesetzt: die pauschale Anerkennung, die individuelle Anerkennung und die Adaption von Lernwegen. Im Rahmen dieses Projekts wurden zunächst Kompetenzen dieser beruflich ausgebildeten Zielgruppe und jene von Studierenden im Studiengang Informatik B.Sc. identifiziert. Letztendlich soll dann anhand von erkannten Kompetenzüberschneidungen ein passgenauer Studiengang entwickelt werden, der durch Verhinderung unnötiger Wiederholung bereits erworbener Kompetenzen Wertschätzung und Anerkennung der Vorkennt-

1 Diese Veröffentlichung ist im Rahmen des Projekts „Durchlässigkeit zwischen beruflicher Ausbildung und Bachelorstudium – vom Fachinformatiker zum Bachelor Informatik durch adaptierte Studiengestaltung“

2 FernUniversität in Hagen, Fakultät für Mathematik und Informatik, Universitätsstraße 11, 58097, Hagen, Deutschland, carsten.thorbruegge@fernuni-hagen.de

3 FernUniversität in Hagen, Fakultät für Mathematik und Informatik, Universitätsstraße 11, 58097, Hagen, Deutschland, joerg.desel@fernuni-hagen.de

4 FernUniversität in Hagen, Fakultät für Kultur- und Sozialwissenschaften, Universitätsstraße 33, 58097, Hagen, Deutschland, len-ole.schaefer@fernuni-hagen.de

5 ‘Anrechnung’ wird für eingebrachte berufliche Leistungen verwendet, ‘Anerkennung’ wird für Leistungen an anderen Hochschulen bzw. im allgemeineren Kontext synonym verwendet.

nisse ermöglicht. Dies ist ein besonderer Anreiz für ein anschießendes Studium und erhöht die Durchlässigkeitserfolge auf institutioneller Ebene.

Aus diesem Hintergrund heraus wurde vom Bundesministerium für Bildung und Forschung (BMBF) die Ankom-Initiative [Fr15] ins Leben gerufen. Diese Initiative markierte einen Wendepunkt in der Forschungsförderung auf Bundesebene, da erstmals eine Förderlinie zur Forschung über Bildungsdurchlässigkeit und seiner Implementierung auf breiter Basis realisiert wurde. Doch die im Rahmen dieser Initiative entwickelten Verfahren zur pauschalen und individuellen Anrechnung stellen nicht den einzig denkbaren Weg zur Wertschätzung beruflich erworbener Kompetenzen dar. So werden zur Reflexion anregende Elemente der individuellen Anrechnung nicht ausreichend berücksichtigt. Die in den ANKOM Projekten entwickelten Verfahren der pauschalen Anrechnung verwenden Kriterien, die nicht hinreichend auf die berufliche Bildung und ihre Anwendungspraxis abgestimmt sind. Neben der Verbesserung der individuellen und pauschalen Anerkennung wurde im Durchlässigkeitsprojekt der FernUniversität in Hagen eine dritte Verkürzungsoption entwickelt, die der adaptierten Lernwege. Die Adaption von Lernwegen bezieht sich auf die Situation, in der ein Modul nicht anrechenbar ist, obwohl wesentliche Teile der Kompetenzen vorliegen. Um beurteilen zu können, ob bzw. welche der genannten Verkürzungsoptionen für ein Modul einsetzbar sind, müssen die typischen Kompetenzen von Informatikstudierenden mit Vorqualifikation ermittelt werden. Dies führt uns zu folgender Forschungsfrage:

- Was sind die typischen Kompetenzen von Informatikstudierenden mit Ausbildung als Fachinformatiker/in?

Hierzu wurde eine Interviewstudie durchgeführt, bei der die relevanten Kompetenzen von Informatikstudierenden mit vorheriger Ausbildung als Fachinformatiker/innen ermittelt wurden. Dies geht einher mit der zweiten Forschungsfrage:

- Welche Hinweise geben die Interviewdaten auf eine passgenaue Zuordnung von Kompetenzen und Verkürzungsoptionen?

Das Ziel der Untersuchung ist die Schaffung einer empirischen Basis für Anrechnungsverfahren, um somit die Passgenauigkeit von Verfahren zur Erhöhung der Durchlässigkeit zu verbessern. Dieser Aspekt ist unmittelbar relevant für das genannte Durchlässigkeitsprojekt der FernUniversität in Hagen. Darüber hinaus besteht aber im Kontext des Lebenslangen Lernens eine Bedeutung für alle Hochschulen, da Anerkennung von den Landeshochschulgesetzen eingefordert wird. Zum einen setzt die vorliegende Studie an der Nahtstelle zur Reduktion des Studienabbruchrisikos an, in dem die Motivation der Studierenden im Hinblick auf die Verwertung bestehender beruflich erworbener Kompetenzen gesteigert wird. Zum anderen wirkt es der Unsicherheit der Studierenden über die eigenen Kompetenzen zu Beginn des Studiums entgegen, da die Reflektion über die eigenen Kompetenzen gesteigert wird.

2 Forschungsstand

Der wesentliche Anstoß zur Anrechnung von außerhalb des Hochschulwesens erworbenen Kenntnissen und Fähigkeiten auf ein Hochschulstudium fand durch einen Beschluss der Kultusministerkonferenz vom 28.06.2002 statt [KM02]. Darauf folgten 2011 die BMBF Initiative „ANKOM – Übergänge von der beruflichen in die hochschulische Bildung“ [Fr15], 2014 das HRK-Projekt „nexus“ zur Übergangsgestaltung [HR14] und 2020 das HRK-Projekt „Modus“ zur Mobilität und Durchlässigkeit [HR20]. In diesen Projekten wurde eine entsprechende Anerkennungspraxis entwickelt bzw. weiterentwickelt. Verwendet werden bspw. die Anerkennungsverfahren nach [MES13] und [SW14], die sich auf die Anerkennungsoptionen der pauschalen und individuellen Anerkennung beziehen, die eine generelle Anerkennung ganzer Studienmodule ermöglichen. Zu einer weitergehenden Studienverkürzung wurde im Durchlässigkeitsprojekt der FernUniversität in Hagen zusätzlich der Ansatz der adaptierten Lernwege [OND21a] entwickelt. Bezugspunkt für die Ermittlung der geforderten Kompetenzen von Fachinformatiker/innen ist dabei das Kompetenzstrukturmodell nach [Op20], in dem die Kompetenzen aus Ausbildungsrahmenplan und Rahmenlehrplan zusammengefasst sind.

Pauschale Anerkennung

Der Kern dieser Anerkennungsart ist das Ersetzen vollständiger Studienmodule (siehe Abb. 1) durch bereits erworbene Kompetenzen. Adressiert werden homogene Studierendengruppen wie die der Fachinformatiker/innen, die in anerkannten Berufsausbildungen Kompetenzen formal erworben haben (vgl. [SRH11, S. 11]. Bezug genommen wird dabei auf Abschlüsse der beruflichen Bildung, so dass nach Vorlage eines formalen Abschlusszertifikates eine unkomplizierte Anerkennung bereits klassifizierter Module ausgesprochen werden kann [ML18]. Der Gegenstand der pauschalen Anerkennung ist ein komplexer, aber nur einmal pro Studienmodul auszuführender Äquivalenzvergleich zwischen erwarteten Kompetenzen aus unterschiedlichen Lernorten. Diese werden in Ordnungsdokumenten wie den Modulhandbüchern der Hochschule und den Rahmenplänen der beruflichen Bildung beschrieben. Konkretere Kompetenzbeschreibungen können dann den jeweiligen Prüfungsaufgaben entnommen werden.

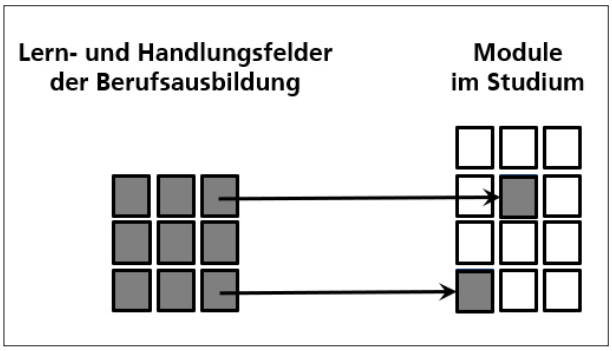

Abb. 1: Kompetenzsubstitution zwischen Berufsausbildung und Studium

Individuelle Anerkennung

Auch an dieser Stelle erfolgt ein Ersetzen vollständiger Studienmodule (siehe Abb. 2) durch bereits erworbene Kompetenzen. Die individuelle Anerkennung richtet sich an einzelne Personen, die Kompetenzen vorwiegend auf nicht-formalem Wege wie in der Weiterbildung oder auf informellem Wege wie durch berufliche Erfahrung erworben haben. Das Portfolio wird als „typische Methode für individuelle Verfahren" [LSRH11, S. 79] der individuellen Anrechnungspraxis von außerhochschulisch erworbenen Lernergebnissen verstanden. Darin werden die zu erwartenden Lernergebnisse des Anerkennungsmoduls vorerfasst, so dass die oder der Studierende über die jeweiligen eigenen beruflichen Vorleistungen reflektieren und diese dann unter Zuhilfenahme von entsprechender Beratung lernergebnisorientiert eintragen kann. Unter Einbeziehung entsprechender Belege und ggf. eines Beurteilungsgespräches mit Fachexperten der Universität erfolgt dann der Bescheid. Weitere Ausführungen zur individuellen Anerkennung sind in [HEO21] zu finden.

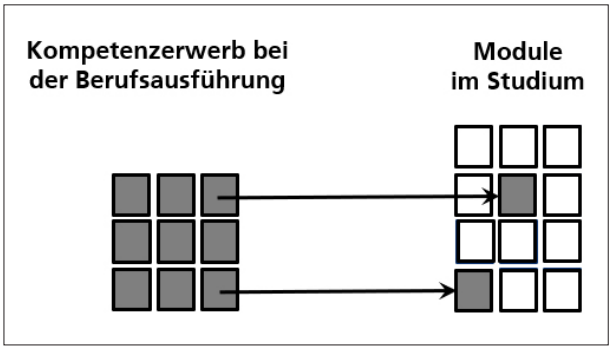

Abb. 2: Kompetenzsubstitution zwischen Berufserfahrung und Studium

Adaptierte Lernwege

Der Ansatz der adaptiven Studiengestaltung kommt zum Tragen, wenn Kompetenzen vorliegen, die kein vollständiges Modul, sondern nur Teile des Moduls ersetzen können (siehe Abb. 3). Angesprochen ist im Rahmen des genannten Projekts zunächst die Gruppe der Studierenden mit formal erworbenen Kompetenzen aus einer Berufsausbildung. Eine Ausweitung auf Studierende mit individuellen Kompetenzen aus anderen Quellen wie zum Beispiel beruflicher Erfahrung ist jedoch denkbar. Es wurde ein alternatives adaptiertes Kursmaterial entwickelt und zur Verfügung gestellt, das an die Vorkenntnisse Studierender angepasst wurde und somit kürzere Lernwege ermöglicht. Lernfortschrittskontrollen und Lernwegsempfehlungen erfolgen dabei automatisiert aus Antwortanalysen zu integrierten Aufgaben. Außerdem ist der Einsatz von Mitteln des Learning Analytics geplant. Der zu Grunde liegende Vergleich zwischen beruflich und hochschulisch erworbenen Kompetenzen erfolgt dabei über das Bewertungstool AsTRA [OND21b], welches auch für eine generellere Anwendung geeignet ist.Weitere Ausführungen zu adaptierten Lernwegen sind in [OND21a] zu finden.

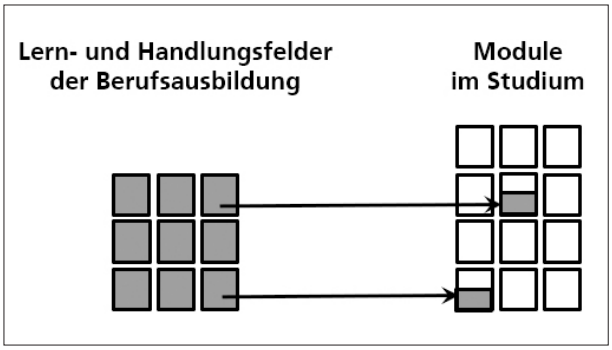

Abb. 3: Kompetenzsubstitution zwischen Berufsausbildung und Studium

3 Methoden

Das methodische Vorgehen beim Design der Studie folgt einem retrospektiven Design, bei dem zehn Bachelorstudierende in Informatik zu ihrer Ausbildung in Fachinformatik und zu ihrer Studieneingangsphase des Bachelorstudiums in Form von Leitfadeninterviews befragt wurden (vgl. [FKS17, S. 255;349-360]). Zwei Informatikstudierende, davon eine weibliche und eine männliche Person, wurden im Rahmen eines Pre-Tests vorab interviewt, so dass das der Analyse zugrundeliegende Sample acht Interviewpersonen umfasst. Zu diesem Vorgehen siehe [Po14, S. 190-205].

Im Hinblick auf das Sampling wurden die Fälle nach dem Prinzip der maximalen Variation ausgewählt (vgl. [HSE13, S. 195]). Es wurde ein sogenanntes *purposeful sampling* durchgeführt (vgl. [Pa02, S. 233-242]). Dies wird vollzogen, indem Bachelorabsolvent/innen mit einer Ausbildung als Fachinformatiker/in über einen Alumniverteiler ausgewählt wurden. Für die Untersuchung war es notwendig, Personen, die über eine Ausbildung in Fachinformatik verfügen, zu identifizieren. Aufgrund des erleichterten Feldzugangs wurden Studierende aus der FernUniversität rekrutiert. Dies wurde von der Leiterin des Absolventenkreises übernommen, die für die Absolventen an der FernUniversität in Hagen verantwortlich ist. Die Leiterin des Absolventenkreises leitete das von den Forschern entwickelte Anschreiben an den Absolventenkreis weiter und ermöglichte den Durchführenden der Studie somit den Zugang zum Forschungsfeld und den Untersuchungspersonen. Des Weiteren wurden über das sogenannte *snowball sampling* Interviewpersonen aus einem Programmierkurs gewonnen. Es handelte sich um Teilnehmende am Grundpraktikum der Programmierung. Dies war eine Lehrveranstaltung im Rahmen des Bachelorstudiengangs Informatik an der FernUniversität in Hagen im Sommersemester 2019.

Im Rahmen der explorativen Studie wurden personenbezogene Daten erhoben. Über diese Verwendung von personenbezogenen Daten für wissenschaftliche Forschungszwecke wurden die Untersuchungspersonen informiert und es wurde eine Einwilligung für die Erhebung von personenbezogenen Daten vor Durchführung des Interviews eingeholt. Bei der Datenerhebung wurde von den Teilnehmenden eine E-Mail-Adresse zur Kontaktaufnahme und Identifizierung aufgenommen. Diese Daten werden in einem getrennten Datensatz gespeichert und nicht mit den weiteren Daten in Zusammenhang gebracht.

Im Rahmen der explorativen Studie wurden die Befragten vor Beginn des Interviews mit Hilfe eines Projektblatts über das Forschungsprojekt informiert. Es folgt die Einstiegsfrage nach dem Verhältnis von Ausbildung und Studium. Ein erster Fragenblock befasst sich mit dem Übergang von der Ausbildung ins Studium und zielt darauf ab, die betrachteten Kompetenzen aus den Kursinhalten abzuleiten. Die Inhalte der Kursmaterialien wurden mit jenen der Ausbildung abgeglichen, indem die Teilnehmenden nach ihren erworbenen Kompetenzen in der Ausbildung befragt wurden. Ein weiterer Baustein war die Zuordnung der erworbenen Kompetenzen zu den drei Verkürzungsoptionen. Die Interviews wurden face-to-face oder per Videotelefonie durchgeführt.Die Interviewdaten wurden unter Berücksichtigung von Transkriptionsregeln transkribiert, die in [Sc19, S. 380] definiert wurden. Die Daten wurden mit der Software MAXQDA inhaltsanalytisch ausgewertet [Ku18]; [RK19]. Es wurde ein Kategoriensystem erstellt, das für jedes einzelne Interview angewendet wurde, somit über die Untersuchungspersonen hinweg konstant blieb und einen thematischen Vergleich ermöglichte. Die Auswertungskriterien wurden deduktiv aus den Themenbereichen der Informatikmodule erstellt sowie induktiv aus dem gewonnenen Datenmaterial der acht Interviews. Das Kategoriensystem wurde mit Hilfe der konsensuellen Kodiertechnik gebildet. Zur Qualitätssicherung wurde das Kategoriensystem danach durch zwei Autoren nach dem Vier-Augen-Prinzip überprüft. Das bedeutet, dass das Interviewmaterial zwischen den Kodierern aufgeteilt wurde und die jeweiligen Kodes und die Entsprechung im Textmaterial bei Widersprüchen diskutiert wurde.Das Interviewmaterial wurde nach den jeweiligen Kursen selektiert und analysiert. Es wurden fachliche und überfachliche Kompetenzen unterschieden. Des Weiteren wurde eine Fallanalyse für einen Vergleich der Fälle zueinander durchgeführt, um Regelmäßigkeiten abzuleiten [Ku18]; (vgl. [FKS17, S. 447-456]).

4 Ergebnisse

Die vorliegende Untersuchung greift die Vorkenntnisse aus der Ausbildung als Fachinformatiker/ in auf und stellt diese jenen des Studiums gegenüber. Damit werden die erworbenen Kompetenzen retrospektiv zugänglich gemacht. Des Weiteren werden die Kompetenzen für die Fachrichtungen der Ausbildung (Anwendungsentwicklung und Systemintegration) differenziert betrachtet. Auf dieser Basis ist es möglich, Anerkennungsverfahren zu verbessern und die Passgenauigkeit von Kompetenzen zu Anerkennungsverfahren zu erhöhen. Dies adressiert die Themen der Fachcommunity, die auf eine effizientere Studiengestaltung und Verbesserung der Studierendensituation ausgerichtet sind.

Fachinformatiker/innen der beiden Fachrichtungen Anwendungsentwicklung und Systemintegration berichten, dass sie in ihrer Ausbildung und in ihrem Beruf in vielen Kernthemen der Informatik zahlreiche Kompetenzen erworben haben. Die Herkunft der Kompetenzen wurde dabei genauer unterschieden, weil Kompetenzen, die in späterer Berufsausübung erworben werden, nicht

zur Anerkennung der beruflichen Ausbildung herangezogen werden können. Zu den geschilderten Kompetenzbereichen zählen insbesondere Programmierung, Betriebssysteme und Rechnernetze sowie Datenbanken. Auffallend ist, dass die Befragten von sich aus ihr Interesse und ihre Kompetenz bzgl. IT-Sicherheit betont haben.

Die Schilderungen der Befragten waren zunächst eher allgemein gehalten. Im Verlaufe der Gespräche wurden diese jedoch – begleitet von unseren Nachfragen - immer detaillierter. Dann wurde auch auf speziellere Programmiertechniken wie die Nutzung bestimmter Schleifenanweisungen eingegangen (I07, Z. 68). Sämtliche Befragten wiesen übereinstimmend auf ihre Programmierkenntnisse vorzugshalber in Java bzw. C# hin (bspw. I03, Z. 61; I04 Z. 79). Das gilt insbesondere für die Fachinformatiker/innen der Fachrichtung Anwendungsentwicklung. Bei den Systemintegratoren handelte es sich vorwiegend um kleinere Script-Programmierungen im Betriebssystem- und Netzwerkbereich, in dem die Anwendung komplexerer Programmierkonzepte nicht notwendig ist. Von vertieften Kenntnissen im Testen von Programmen wurde nicht berichtet. Der - wenn auch begrenzte - Bereich der Rekursionstechnik war den Befragten so gut wie nicht bekannt [I04, Z. 154].

Das Modul „imperative Programmierung" des Studiengangs Informatik B.Sc. vermittelt grundlegende Programmierkonzepte. Die Antworten auf die entsprechenden Fragen des Leitfadeninterviews deuten an, dass die in der Ausbildung als Fachinformatiker/in erworbenen Programmierkompetenzen die im Modul „imperative Programmierung" zu erwerbenden Kompetenzen in Bezug auf Inhalt und Niveau hinreichend ersetzen können. Dies wird bestärkt durch den diesbezüglich genannten Verkürzungswunsch sowie der fachlichen Einschätzung des Interviewers. Damit kommt die Verkürzungsoption der pauschalen Anerkennung in Betracht, die eine vollständige Ersetzbarkeit von Studienmodulen durch in der Ausbildung erworbene Kompetenzen vorsieht. Es werden Themenbereiche der Ausbildung genannt, die sich im betrachteten Curriculum wiederholen.

Fachinformatiker/innen der Fachrichtung Systemintegration geben Wissen über Betriebssysteme wie Unix und Windows an. Eine Person bezieht sich dabei auf das Großrechner-Betriebssystem z/OS (I05, Z. 166). Bei Betriebssystemen und bei Rechnernetzen sind Systemintegratoren mit der Administration vertraut und kennen sich mit Kommunikationsschichten und -protokollen aus (I03, Z. 138; I09, Z. 70). Damit deuten sich erworbene Kompetenzen an, die Teile des Moduls „Softwaresysteme" im Studiengang Informatik B.Sc. abdecken, zu dem der Kurs „Betriebssysteme und Rechnernetze" gehört. Diese Situation findet bei adaptierten Lernwegen Berücksichtigung.

Bezugnehmend zum Kurs „Datenbanken I" des Moduls „Softwaresysteme" geben die Befragten beider Fachrichtungen der Fachinformatik übereinstimmend Kenntnisse in Datenbanken wie MySQL und MS SQL an. Sie können Daten über SQL abfragen, jedoch sind mathematische Beschreibungssprachen einer Datenanfrage wie die Relationenalgebra und das Relationenkalkül nicht bekannt (I06, Z. 145; I04 Z. 174). Weil an dieser Stelle nicht von vollständigen Kompetenzen berichtet wurde, bietet sich hier der Ansatz der adaptierten Lernwege an. Adaptierte Lernwege stellen einen innovativen Ansatz zur Passung des Lehrmaterials an die vorhandenen Studierendenkompetenzen dar.

Systemintegratoren berichteten außerdem von Kenntnissen in Verschlüsselungsverfahren (I03, Z.78) und Tätigkeiten im Bereich Firewalling (I03, Z.83), die in Bezug zum Modul „IT-Sicherheit" gesetzt werden können. Damit ist die Verkürzungsoption der individuellen Anerkennung relevant, die die Ersetzung eines Moduls über in der Berufsausführung erworbenen Kompetenzen ermöglicht. Die gewonnenen Informationen leisten bezüglich der individuellen Anerkennung einen Beitrag zur Entwicklung und Einführung eines Kompetenzportfolios, das den Studierenden eine Verwaltung und Reflektion über die eigenen Kompetenzen ermöglicht.

Einen weiteren wichtigen Punkt stellen überfachliche Kompetenzen dar. Hierzu wurden von den Befragten durchgängig sogenannte Selbststeuerungskompetenzen in Form der Selbststrukturierung von Arbeitsprozessen, der Selbstlernkompetenz und der Selbstorganisation hervorgehoben (I03, Z.69; I05, Z. 86), die sich mit der Theorie zur Selbststeuerungskompetenz nach Weinert decken. Nach Weinert nimmt die Entscheidungsfähigkeit des Einzelnen im Hinblick auf das selbstgesteuerte Lernen einen zentralen Stellenwert ein. So kann beispielsweise die Art und Weise, der Lerninhalt, das Lernziel sowie die zeitliche Fixierung des Lernens festlegt werden (vgl. [We82, S. 102]). Das Lernen basiert erstens auf dem Prinzip der Selbständigkeit. Jedes Individuum fokussiert sich auf Lernziele und wählt geeignete Lernmethoden in einer zur Verfügung stehenden Zeit aus. Zweitens wird als unbewusster oder bewusster Vorgang eine entsprechende Lernhandlung realisiert. Die Überwachung der Lernvorgänge wird durch die Rolle des Lehrenden ausgefüllt, die der Einzelne selbst ausübt (vgl. [We82, S. 102]). Die Selbststeuerungskompetenz stellt eine Kernkompetenz der Studierenden dar, die sich Studieninhalte selbständig erarbeiten und organisieren müssen.

5 Zusammenfassung und Diskussion

Die typischen Kompetenzen von Informatikstudierenden mit einer Ausbildung als Fachinformatiker/ in setzen sich aus fachlichen und überfachlichen Kompetenzen zusammen. Fachinformatiker/innen der Fachrichtung Anwendungsentwicklung besitzen bereits fundierte Programmierkenntnisse vorzugsweise in Java bzw. C#, während Systemintegratoren mit Kommunikationsschichten und Protokollen vertraut sind und kleinere Script-Programme entwickeln. Stilgerechtes Programmieren und Testen von Programmen haben eine geringere Bedeutung. Auf Qualitätssicherung wird in der Ausbildung im größeren Rahmen nicht eingegangen.

In überfachlicher Hinsicht können Selbststeuerungskompetenzen identifiziert werden wie die Fähigkeit, Arbeitsprozesse zu strukturieren und Selbstlernvorgänge zu aktivieren. Die vorliegende Studie deutet daraufhin, dass es sich um eine Kompetenz handelt, die insbesondere in der Ausbildung erworben wird. Des Weiteren ist es eine Kompetenz, die von den Studierenden erwartet und entwickelt wird. Das heißt, dass die Studierenden, die bereits über ausgeprägte Selbststeuerungskompetenzen aus ihrer Ausbildung verfügen, einen Startvorteil für die Aneignung von Studieninhalten in der Studieneingangsphase haben. Diesen Startvorteil für die Adaption von Lehrformen zu verwenden, kann von wesentlicher Bedeutung für die adäquate Berücksichtigung von zuvor erworbenen Kompetenzen von Informatikstudierenden sein und würde die Adaption von Lernwegen innovativ in didaktischer Hinsicht ergänzen.

Des Weiteren liefert die Interviewstudie Hinweise auf die Zuordnung von Modulen zu den jeweiligen Verkürzungsoptionen. Für sämtliche der im Projekt betrachteten Verkürzungsoptionen zu Modulen fanden sich in den Schilderungen der Befragten Anwendungsbeispiele. Erstens kann die pauschale Anerkennung unterschieden werden, die eine generelle Abdeckung der Kompetenzen ermöglicht und zweitens die individuelle Anerkennung, die Berufserfahrung ablichtet, indem Kompetenzen beispielsweise in IT-Sicherheit abgebildet werden können und drittens die Adaption von Lernwegen, bei denen Teilkompetenzen in Betriebssystemen und Rechnernetzen sowie Datenbanken identifiziert werden.

Erkenntnisreich waren die geschilderten beruflichen Kompetenzen aus anderen Domänen und Bereichen, die sich so in unserem Curriculum für den Studiengang Informatik B.Sc. nicht wiederfinden. Das eröffnet uns den Blick für eine vierte Säule der Anerkennung. Im Rahmen dieser kontextuellen Anerkennung werden Kompetenzen, die in anderen Kontexten erworben worden sind, auf Gleichwertigkeiten überprüft. Dies ist beispielsweise für Nebenfächer relevant.

Auch wenn es sich um keinen größeren Sampleumfang handelt, gab es zum Teil Ähnlichkeiten in den Aussagen der Studierenden, so dass die Interviewstudie wichtige Informationen für das genannte Durchlässigkeitsprojekt zur Umsetzung der Anerkennung von Vorqualifikationen liefern konnte. Zusätzliche Analysen aus Referenzdokumenten der beruflichen und hochschulischen Informatikausbildung wie Rahmenpläne, Modulhandbücher, Lehr- und Prüfungsmaterialen, die die erwarteten Kompetenzen aufzeigen, bieten sich an.

Die erzielten Erkenntnisse lassen sich auch auf andere Hochschulen übertragen. Dabei geht es um Sensibilität gegenüber vorhandenen fachlichen und überfachlichen Kompetenzen. Dies erlaubt eine präzisere Studiengestaltung, insbesondere am Übergang in die Studieneingangsphase. Auf Basis dieser empirischen Ergebnisse kann die Studiensituation beruflich Vorqualifizierter genauer bewertet und durch Anwendung von Anerkennungsoptionen verbessert werden, in dem Lernwiederholungen, Unter- und Überforderungen vermieden werden. Letztendlich werden damit Impulse für wichtige gesellschaftliche Aspekte wie Durchlässigkeit, Lebenslanges Lernen und Diversität gegeben.

Das Projekt wird gefördert durch den *Stifterverband* im Rahmen eines *Fellowship für Innovation in der Hochschullehre*.

Literaturverzeichnis

[BH16] Birke, Barbara; Hanft, Anke: Anerkennung und Anrechnung non-formal und informell erworbener Kompetenzen: Empfehlungen zur Gestaltung von Anrechnungs- und Anerkennungsverfahren. facultas, Wien, 2016.

[EN20] Elsholz, Uwe; Neu, Ariane: Hybride Bildungsformate – Instrumente zur Förderung von Durchlässigkeit? (http://denk-doch-mal.de/wp/profdr-uwe-elsholz-ariane-neu-hybride-bildungsformate-instrumente-zur-foerderung-von-durchlaessigkeit, Zugegriffen am 18.05.21), 2020.

[FKS17] Flick, Uwe; Kardorff, Ernst von; Steinke, Ines, Hrsg. Qualitative Forschung: ein Handbuch. Rororo Rowohlts Enzyklopädie 55628. Rowohlt Verlag, Hamburg, 12. Auflage, 2017. [Fr15] Freitag, Walburga; Buhr, Regina; Danzeglocke, Eva-Maria; Schröder, Stefanie; Völk, Daniel, Hrsg. Übergänge gestalten: Durchlässigkeit zwischen beruflicher und hochschulischer Bildung erhöhen. ANKOM Übergänge von der beruflichen in die hochschulische Bildung. Waxmann, Münster New York, 2015.

[Ha14] Hanft, Anke; Brinkmann, Katrin; Gierke, Willi B.; Müskens, Wolfgang: Anrechnung außerhochschulischer Kompetenzen in Studiengängen. Studie: AnHoSt „Anrechnungspraxis in Hochschulstudiengängen" (https://uol.de/fileadmin/user_upload/anrechnungsprojekte/Anhost.pdf, Zugegriffen am 18.05.21), 2014.

[HB12] Hanft, Anke; Brinkmann, Katrin: Offene Hochschulen. Die Neuausrichtung der Hochschulen auf Lebenslanges Lernen: Die Neuausrichtung der Hochschulen auf Lebenslanges Lernen. Waxmann Verlag, Münster, 2012.

[HEO21] Hunyh, Hoai Nam; Elsholz, Uwe; Opel, Simone: Individuelle Anrechnung außerhochschulischer Kompetenzen am Beispiel eines informatischen Studiengangs. In: HDI 2021 Hochschuldidaktik Informatik, Dortmund, 2021. September 2021.

[HR14] HRK 2014: Projekt nexus - Übergänge gestalten, Studienerfolg verbessern. www.hrk-nexus.de. Zugegriffen am 18.05.2021.

[HR20] HRK 2020: Projekt Modus - Mobilität und Durchlässigkeit stärken. www.hrkmodus.de. Zugegriffen am 18.05.2021.

[HSE13] Hussy, Walter; Schreier, Margrit; Echterhoff, Gerald: Forschungsmethoden in Psychologie und Sozialwissenschaften. Springer Verlag, Berlin und Heidelberg, 2. Auflage, 2013.

[KM02] KMK 2002: KMK (2002). Anrechnung von außerhalb des Hochschulwesens erworbenen Kenntnissen und Fähigkeiten auf ein Hochschulstudium. Bonn. 2002.

[Ku18] Kuckartz, Udo: Qualitative Inhaltsanalyse: Methoden, Praxis, Computerunterstützung. Grundlagentexte Methoden. Beltz Juventa, Weinheim Basel, 4. Auflage, 2018.

[LSRH11] Loroff, Claudia; Stamm-Riemer, Ida; Hartmann, Ernst: Anrechnung: Modellentwicklung, Generalisierung und Kontextbedingungen. In (Freitag, Walburga Katharina; Hartmann, Ernst A.; Loroff, Claudia; Stamm-Riemer, Ida; Völk, Daniel; Buhr, Regina, Hrsg.): Gestaltungsfeld Anrechnung: Hochschulische und berufliche Bildung im Wandel, S. 77–117. Waxmann, Münster, 1. Auflage, 2011.

[MES13] Müskens, Wolfgang; Eilers-Schoof, Anja: Application of the Module Level Indicator (MLI) in the context of transnational comparisons of qualifications. In (Tutschner, Roland; Wittig, Wolfgang, Hrsg.): Level Assessments of Learning Outcomes in Health Care and Nursing. Institut für Technik und Bildung (ITB), Bremen, 2013.

[ML18] Müskens, Wolfgang; Lübben, Sonja: Die Anrechnung non-formalen und informellen Lernens auf Hochschulstudiengänge in Deutschland. Zeitschrift für Weiterbildungsforschung, 41(2-3):109–124, 2018.

[OND21a] Opel, Simone; Netzer, Cajus Marian; Desel, Jörg: Adaption von Lernwegen in adaptierten Lehrmaterialien für Studierende mit Berufsausbildungsabschluss. In: HDI 2021 Hochschuldidaktik Informatik, Dortmund, 2021. September 2021.

[OND21b] Opel, Simone; Netzer, Cajus Marian; Desel, Jörg: AsTRA - An Assessment Tool for Recognition and Adaption of Prior Professional Experience and Vocational Training. In: Proceedings of OCCE 2021 DTEL-IFIP TC3 Conference. Tampere, Finnland (in Druck), 2021.

[Op20] Opel, Simone: Entwicklung eines arbeitsprozessorientierten Kompetenzstrukturmodells für die Ausbildung zum Fachinformatiker bzw. zur Fachinformatikerin. Dissertation, DuEPublico: Duisburg-Essen Publications online, University of Duisburg-Essen, Germany, Oktober 2020.

[Pa02] Patton, Michael Quinn: Qualitative Research & Evaluation Methods. Sage Verlag, 3. Auflage. Thousand Oaks, 2002.

[Po14] Porst, Rolf: Fragebogen. Ein Arbeitsbuch. Springer Verlag, Wiesbaden, 2014.

[RK19] Rädiker, Stefan; Kuckartz, Udo: Analyse qualitativer Daten mit MAXQDA: Text, Audio und Video. VS Verlag für Sozialwissenschaften, Wiesbaden, 2019.

[Sc19] Schäfer, Len Ole: Universitäten im Leistungswettbewerb: Forschungsevaluation in Großbritannien. VS Verlag für Sozialwissenschaften, Wiesbaden, 2019.

[SRH11] Stamm-Riemer, Ida; Hartmann, Ernst A.: Entwicklungen und Trends im ANKOM-Kontext zu Anrechnung und Durchlässigkeit zwischen beruflicher und hochschulischer Bildung in Deutschland und Europa. In (Freitag, Walburga, Hrsg.): Gestaltungsfeld Anrechnung: hochschulische und berufliche Bildung im Wandel. Waxmann, Münster, 2011.

[SW14] Seger, Mario; Waldeyer, Christina: Qualitätssicherung im Kontext der Anrechnung und Anerkennung von Lernergebnissen an Hochschulen: Standards für transparente und nachvollziehbare Analyseverfahren und Anrechnungsprozesse: inkl. Musteranrechnungsleitfaden und Musteranrechnungsordnungen: Entwicklungsergebnisse aus dem F&E-Projekt Open Competence Center for Cyber Security im BMBF-Wettbewerb Aufstieg durch Bildung: offene Hochschulen. Darmstädter Studien zu Arbeit, Technik und Gesellschaft Band 14. Shaker Verlag, Aachen, 2014.

[We82] Weinert, Franz E: Selbstgesteuertes Lernen als Voraussetzung, Methode und Ziel des Unterrichts. In Unterrichtswissenschaft, Zeitschrift für Lernforschung, Unterrichtswissenschaft Weinheim 10. S. 13, 1982.

Adaption von Lernwegen in adaptierten Lehrmaterialien für Studierende mit Berufsausbildungsabschluss

Simone Opel[1], Cajus Marian Netzer[1], Jörg Desel[1]

Abstract:
Obwohl immer mehr Menschen nicht direkt ein Studium aufnehmen, sondern zuvor eine berufliche Ausbildung absolvieren, werden die dort erworbenen Kompetenzen von den Hochschulen inhaltlich und didaktisch meist ignoriert. Ein Ansatz, diese Kompetenzen zu würdigen, ist die formale Anrechnung von mitgebrachten Kompetenzen in Form von Leistungspunkten. Eine andere Variante ist der Einsatz von speziell für die Zielgruppe der Studierenden mit Vorkenntnissen adaptiertes Lehr-Lernmaterial. Um darüber hinaus individuelle Unterschiede zu berücksichtigen, erlaubt eine weitere Adaption individueller Lernpfade den Lernenden, genau die jeweils fehlenden Kompetenzen zu erwerben. In diesem Beitrag stellen wir die exemplarische Entwicklung derartigen Materials an Hand des Kurses „Datenbanken" für die Zielgruppe der Studierenden mit einer abgeschlossenen Ausbildung zum Fachinformatiker / zur Fachinformatikerin vor.

Keywords:
Informatik; Anrechnung; Adaption; Individuelle Lernwege; Vorwissen; Kompetenz; Datenbanken; Hochschule; Fachinformatiker

1 Einführung

Nicht nur bessere berufliche Chancen, sondern auch die wachsende Durchlässigkeit zwischen den einzelnen Bildungswegen (vgl. bspw. [De14]) motiviert immer mehr Menschen, nach einer Berufsausbildung ein Studium aufzunehmen. Gerade an der FernUniversität in Hagen ist der Anteil der Studierenden mit Berufsausbildung oder Berufserfahrung recht hoch (rd. 80% berufstätige Studierende [Fe21]). Dort wie anderswo bringen diese Personen ihre Vorkenntnisse und Erfahrungen in ein Studium ein, ohne dass dies von den Hochschulen explizit gewürdigt wird.

Seit einigen Jahren gibt es seitens der HRK [Ku02] Bestrebungen, solche mitgebrachten Vorkenntnisse und Kompetenzen systematisch zu honorieren. Insbesondere Hochschulen, die sich an den Projekten ANKOM oder HRK Nexus beteiligten, rechnen äquivalente Kompetenzen formal als Ersatzleistungen für einzelne Module an[2]. Derartige Regelungen sind ein erster Schritt in die richtige Richtung, aber nicht ausreichend: Bei Studierenden, die direkt nach Abschluss einer beruflichen Ausbildung ein Studium im gleichen Bereich aufnehmen, sind Kompetenzen aus vielen verschiedenen Modulen vorhanden, die jedoch nicht ausreichen, um ein gesamtes Modul anzurechnen. Diese Studierenden benötigen ein passgenaues, strafferes und motivierenderes Studienangebot, das ihnen erlaubt, nur die Kompetenzen zu erwerben, die ihnen tatsächlich fehlen – das sind fachliche, aber auch überfachliche Kompetenzen, zum Beispiel Aspekte des wissenschaftlichen Arbeitens.

Im Rahmen von Blended-Learning-Szenarios und Distanzlehre, die elementar für die Studiengänge der FernUniversität in Hagen sind, stehen hierzu viele Wege offen. Durch den großen Anteil an Selbstlernelementen passen die Studierenden auf Basis ihrer Selbsteinschätzungen ihre Lernwege – meist ohne sich darüber Gedanken zu machen – selbst immer wieder an. Auch außerhalb der FernUniversität werden Verfahren zur Adaption eingesetzt, die aber in der Regel nicht spezifisch für bestimmte Gruppen von Lernenden entwickelt wurden, sondern auf allgemeinen Vorkenntnissen basieren und somit einen Ansatz in Richtung „One fits all" darstellen.

1 alle Autoren: FernUniversität in Hagen, Fakultät für Mathematik und Informatik, Universitätsstraße 11, 58097 Hagen, simone.opel|cajus-marian.netzer|joerg.desel@fernuni-hagen.de

2 „Anrechnung" wird hier im Einklang mit dem HRK Nexus-Projekt (vgl. [Be19, S. 6]) für eingebrachte berufliche Leistungen verwendet, „Anerkennung" für nachgewiesene Leistungen an anderen Hochschulen.

Ein anderer Weg, der insbesondere für Hochschulen mit heterogener Studierendenschaft gewinnbringend ist, ist die *Adaption der persönlichen Lernwege* durch Studierende in zielgruppenspezifischem Lehr-Lernmaterial, das durch eine entsprechende hochschulseitige *Adaption des bestehenden Lehr-Lernmaterials* entstanden ist. Diese Adaption ist aber nur gewinnbringend möglich, wenn man die von der Zielgruppe mitgebrachten Kompetenzen samt Variabilität und daraus resultierend die generelle, wie individuelle Kompetenzlücke zum gewünschten Learning-Outcome kennt oder bestimmen kann.

In diesem Beitrag stellen wir einen ersten Ansatz für eine derartige Adaption am Beispiel eines Kurses „Datenbanken“[3] exemplarisch für Studierende mit einer Ausbildung als Fachinformatikerin bzw. Fachinformatiker vor, die darauf aufbauend ein Bachelor-Studium in Informatik an der FernUniversität in Hagen absolvieren. Die Adaption der persönlichen Lernwege durch Adaption des bestehenden Lehr-Lernmaterials ist in das Projekt „Durchlässigkeit“[4] des Forschungsschwerpunkts D²L² (Digitalisierung, Diversität und Lebenslanges Lernen. Konsequenzen für die Hochschulbildung) eingebettet, das die Verbesserung der Durchlässigkeit zwischen beruflicher und akademischer (Informatik-)Bildung zum Ziel hat.

2 Grundgedanken zur Adaption des Lehr-Lernmaterials

Mitgebrachte Kompetenzen der Fachinformatikerinnen und Fachinformatiker
Studierende im Bachelor-Studiengang Informatik mit einer abgeschlossenen Berufsausbildung zum Fachinformatiker bzw. zur Fachinformatikerin[5] haben eine umfassende Ausbildung in Informatik und IT, deren Stärke in der Lernortkooperation zwischen Betrieb und Berufsschule liegt: Im schulischen Teil der Ausbildung wird ein breites berufspraktisches und auch fachtheoretisches Wissen vermittelt, das sich auf längerfristig geltende Prinzipien und Fachkonzepte stützt. Der Ausbildungsbetrieb hingegen fördert darauf aufbauend die Entwicklung von spezifischen, berufsbezogenen Kompetenzen, die die Fachinformatiker/innen im Lauf ihrer Berufstätigkeit noch vertiefen oder verbreitern. Insbesondere durch die Tätigkeit im Ausbildungsbetrieb lernen die Auszubildenden, in beruflichen Handlungssituationen zu denken und alleine oder im Team Lösungen auch für unbekannte Probleme zu entwickeln. Durch die formal und inhaltlich gesetzlichen Rahmen der Berufsausbildung (Berufsbildungsgesetz [Bu20], für den Ausbildungsbetrieb: Ausbildungsordnung [BM20], sowie Rahmenlehrplan [Ku19] für den schulischen Teil) kann für die *Gruppe der Fachinformatiker/innen* eine durchschnittlich zu erwartende *Kompetenz*[6] für die verschiedenen Bereiche der Informatik postuliert werden.

Allerdings können die Auszubildenden trotz Lücken in Themenbereichen, die auch für das Studium relevant sind, die Berufsabschlussprüfung bestanden haben. Andere Auszubildende erwerben während der Ausbildung in bestimmten, für sie relevanten Bereichen vertiefte Kompetenzen, die weit über das Niveau der Ausbildung hinausgehen und vielleicht dem Niveau des Bachelor-Studiums genügen. Hieraus folgt, dass das bestehende Lehr-Lernmaterial so adaptiert werden muss, dass nicht nur die durch die formalen Vorgaben zu erwarteten Kompetenzen der Gruppe der Fachinformatiker/innen beachtet werden, sondern dass eine zusätzliche individuelle Adaption ihrer Lernwege es ihnen erlaubt, maximal von der auf ihrer Ausbildung basierenden Adaption zu profitieren.

Mitgebrachte Kompetenzen im Bereich „Relationale Datenbanken“
Fachinformatiker/innen erwerben während ihrer Ausbildung grundlegende Kompetenzen über den Prozess der Modellierung von relationalen Datenbanken. Sie können verschiedene Modellierungsprinzipien (Entity-Relationship Model (ERM), logische Modellierung) anwenden, beschäftigen sich je nach Fachrichtung[7] in verschiedener Tiefe mit SQL, und können auch Konzepte wie Normalformen, Integritätsregeln und ähnliches anwenden. Zudem erlernen sie die Grundlagen der Datenbankadministration und von Berechtigungssystemen, in der Tiefe abhängig von der jeweiligen Fachrichtung.

Didaktische Überlegungen zur Gestaltung des adaptierten Lehr-Lernmaterials
Vergleicht man die Systeme *berufliche* und *hochschulische Bildung*, erkennt man, dass sie von unterschiedlichen Sichtweisen ausgehen: Während ein Hochschulstudium nach fachlich abgegrenzten Modulen strukturiert ist (vgl. auch [De16]), orientiert sich die berufliche Bildung in ihren Ordnungsdokumenten an Arbeitsprozessen, die im Rahmenlehrplan als Lernfelder abgebildet werden und didaktisch reduzierte berufliche Handlungsfelder darstellen (vgl. bspw. [BS98] oder [KS01]).

3 Dieser Kurs bildet zusammen mit einem weiteren ein Modul des zweiten Studienjahres.

4 www.fernuni-hagen.de/forschung/schwerpunkte/d2l2/projekte/durchlaessigkeit.shtml

5 Im Weiteren werden diese Personen kurz Fachinformatiker/innen genannt.

6 Diese Kompetenzen stellen auch einen Maßstab für eine pauschale Anrechnung von Modulen dar.

7 Fachrichtungen der Fachinformatiker/innen Anwendungsentwicklung oder Systemintegration, die Neuordnung der IT-Berufe trat erst zum Ausbildungsjahr 2020/21 in Kraft.

Um diese beiden Sichtweisen zusammenzuführen, und auch die Vorerfahrungen der Fachinformatiker/innen einzubeziehen, sollten sie insbesondere in den frühen Phasen des Studiums „dort abgeholt werden, wo sie sich befinden". Betrachtet man daher das eher praktisch orientierte Vorwissen der Studierenden als Präkonzept im Sinne des Conceptual Change [CSD94], wird das bestehende Konzept mit Hilfe von sinnvoll gestaltetem Material durch die Studierenden zu einem theoretisch und mathematisch untermauerten Konzept aktiv umgebaut. In der Umsetzung der Adaption heißt das, dass ihnen (ähnlich dem Unterricht an einer Berufsschule) eine Lernsituation als didaktisch reduzierter Handlungsprozess angeboten wird. Die Studierenden werden hierdurch dazu motiviert, die für die Lösung dieser Aufgaben notwendigen, ihnen aber noch fehlenden fachtheoretischen Kenntnisse zu erwerben und damit ihre berufspraktischen Kompetenzen zu fundieren. Das Anknüpfen an die schon aus dem schulischen Teil der Ausbildung bekannten Wissenskomponenten hat noch einen weiteren Vorteil: es erlaubt ihnen, ihren sog. Cognitive Load zu vermindern [Sw11], der durch das benötigte vernetzte und aktive Lernen im Studium relativ hoch ist, indem die Lernenden eigene Wissenslandkarten aufbauen (vgl. u.a. [Re12]). Der Cognitive Load sinkt, wenn das neue in das schon vorhandene Wissen eingebaut werden kann. Daher sind explizite Bezüge im Lehr-Lernmaterial zum schon bekannten Inhalt der Ausbildung positiv zu bewerten und bei der Adaption vorzusehen.

3 Adaption des Lehr-Lernmaterials

Im vorherigen Abschnitt sehr allgemein wurde dargestellt, welche Kompetenzen Fachinformatiker/innen ins Studium der Informatik mitbringen und welche generellen Folgen dies für eine Adaption des Materials nach sich zieht. Aus diesen generellen Überlegungen ergeben sich anschließend die folgenden Fragen, die in diesem Abschnitt beantwortet werden:

- Welche fachlichen Defizite aus hochschulischer Sicht hat die Gruppe der Fachinformatiker/innen?
- Wie muss Lehr-Lernmaterial gestaltet werden, damit die Studierenden ihre individuellen Defizite erkennen und somit in die Lage versetzt werden, ihren Lernweg so gestalten, dass sie dieses fehlende Wissen auch erwerben können?

Um die bestehenden Lehr-Lernmaterialien für die Studierenden gewinnbringend zu adaptieren, wird ein Design-Science-Ansatz [DH16] verwendet, in dessen Rahmen die Adaption des Kurses „Datenbanken" protoypisch umgesetzt und evaluiert wird.

3.1 Analyse bestehender Materialien

Struktur des Kurses „Datenbanken"
Einen ersten Schritt der Adaption stellte die grundlegende Analyse der vorhandenen Lehr-Lerntexte dar. Hierzu wurde der Lehrtext an Hand seiner (nummerierten) Abschnitte zerlegt und die darin zu erwerbenden Kompetenzen erfasst. Anschließend wurden die logischen Abhängigkeiten und Verweise innerhalb der Abschnitte in einem Satz von Concept-Maps dargestellt (vgl. Abb. 1). Auf diese Weise war es möglich, sowohl einen Gesamtüberblick als auch ein tieferes Verständnis für die sachlogischen Zusammenhänge des bestehenden Lehr-Lernmaterials zu erhalten.

Kompetenzlücke: Fehlende Inhalte der Ausbildung
Um einen Inhalts- und Niveauvergleich zwischen Ausbildungsinhalten und den Qualifikationszielen des Kurses „Datenbanken" durchzuführen, wurden die Ordnungsdokumente der Berufsausbildung durch ein arbeitsprozessorientiertes Kompetenzstrukturmodell [Op20] ergänzt, das die Kompetenzen beschreibt, die ein/e Fachinformatiker/in am Ende der Ausbildung entwickelt haben sollte. Der Vergleich selbst wurde mittels des im Rahmen des Projekts „Durchlässigkeit" entwickelten Tools AsTRA [OND21] durchgeführt. AsTRA vereinfacht Inhalts- und Niveauvergleiche der Learning Outcomes verschiedener Bildungssysteme durch die Einführung eines algorithmisierten, nachvollziehbaren und verlässlichen Verfahrens.

Weite Teile des Kurses werden durch die Ausbildung sowohl hinsichtlich des Inhaltes als auch des Niveaus abgedeckt. Allerdings sind die verschiedenen mathematischen Verfahren nicht Bestandteil der Ausbildung: für den adaptierten Kurs „Datenbanken" sind das im Wesentlichen das Relationenkalkül und die Relationenalgebra, sowie alle Themen, die diese Verfahren anwenden, wie die Definition bzw. der Nachweis von (voller) funktionaler Abhängigkeit oder die Optimierung von Abfragen. Auch Elemente weiterer Inhaltsbereiche fehlen, wie die erweiterte Darstellung des ERM oder einige Aspekte von SQL (Subqueries und Transaktionen werden oft nur in der Fachrichtung Anwendungsentwicklung behandelt).

Die in der Ausbildung fehlenden Inhaltselemente finden sich im bestehenden Lehr-Lernmaterial an sehr unterschiedlichen Stellen. Sie stehen aber in einem logischen Zusammenhang mit anderen Themenfeldern (vgl. Abb. 1), wodurch die Gestaltung der Adaptionsmöglichkeiten für die unterschiedlichen Lernwege der Studierenden eine Herausforderung darstellt, die nicht durch „Weglassen" oder einfache Hinweistexte gelöst werden kann.

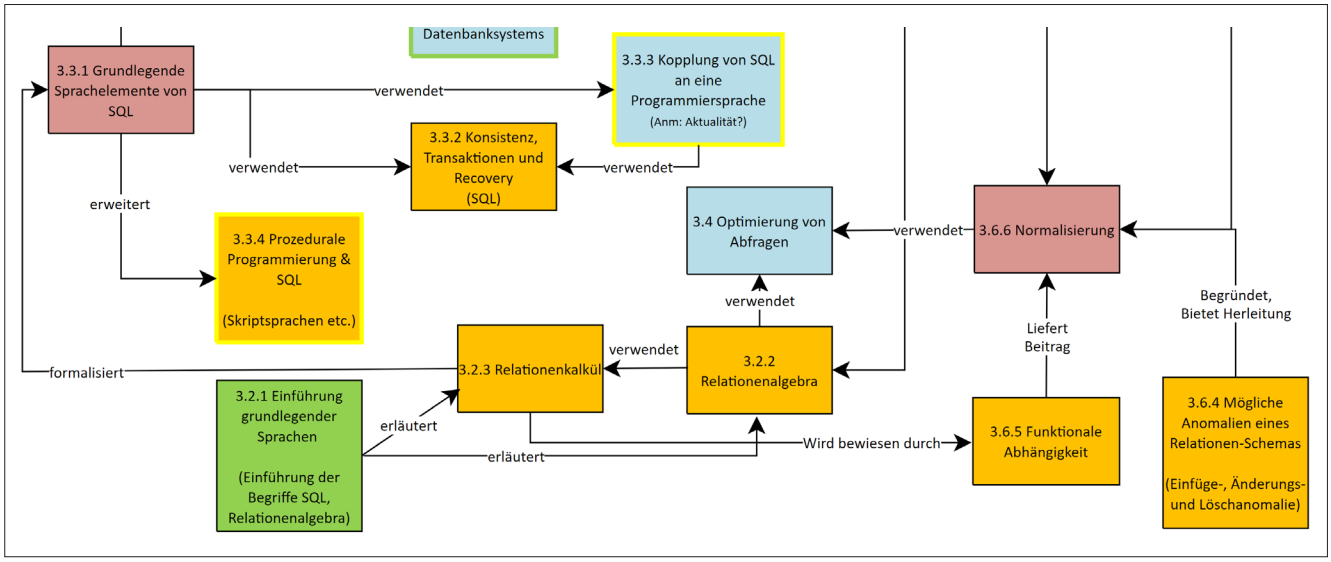

Abb. 1: Ausschnitt aus der Concept Map des Kurses *Datenbanken*. Farbcodierung: gelb, blau – Exkurs-ähnliche Texte, rotbraun – große, weiter in Detailmaps zu zerlegende Abschnitte, orange – inhaltlich abgeschlossene Abschnitte, grün – ergänzende Abschnitte.

3.2 Grundlegende Konzeption der Lehrtextadaption

Die bisherigen Gedanken und Erkenntnisse wurden in einem strukturierten Rahmenkonzept vereinigt:

Die Fachinformatiker/innen erhalten das adaptierte Lehr-Lernmaterial in einer *Moodle-Umgebung*, ergänzend zum originären, gedruckten Lehr-Lerntext. Blended-learning und insbesondere Moodle als Lernumgebung sind an der FernUniversität in Hagen flächendeckend eingeführt und daher allen Studierenden vertraut, so dass hierdurch kein neuer Cognitive Load geschaffen wird.

Um die neuen Wissenselemente in die Landkarte des individuellen Vorwissens einbauen zu können, wird analog zu den Methoden des Berufsschulunterrichts eine durchgängige *Lernsituation* entwickelt, die den gesamten Entwicklungsprozess einer Datenbank umfasst und die Studierenden entlang dieses Prozesses durch die Lehr-Lernmaterialien führt. Lernen an Hand einer beruflichen Handlungssituation ist den Studierenden vertraut, und die Verknüpfung mit bekannten Methoden erleichtert den Konzeptwechsel von einem eher berufspraktischen hin zu einem akademischen Wissenskonzept.

Die mit dieser Lernsituation verbundenen *Aufgabenstellungen* werden kompetenzorientiert formuliert. Es wird kein Detailwissen erwartet, sondern *Verständnis* für das Thema. Bei der Formulierung wird beachtet, welche Probleme und Fehlvorstellungen von Lernenden zum Thema Datenbanken bekannt sind. Ntshalinthali et al. [NC20] benennen verschiedene Fehlvorstellungen sowohl auf deklarativer als auch konzeptueller Ebene, die auch bei den Fachinformatiker/inne/n zu erwarten sind (z.B. Verstehen der 3NF als Nachteil für die Datenabfrage, Schlüssel als immer eindeutig (jeweils deklarativ), fehlende Fähigkeit, die Konzepte Tabelle und Relation zu verbinden (konzeptuell)). Poulsen et al. [Po20] untersuchten digital eingereichte Lösungen von Studierenden zu SQL-Statements. 27% der Fehler, die zum Abbruch der Aufgabe führten, stellen in dieser Untersuchung Syntax-Fehler dar, semantische Fehler beziehen sich besonders auf die Verwendung von JOIN, GROUP BY sowie Unterabfragen – Konzepte, die damit von den Autoren als schwierig identifiziert wurden.

Weiter betonen beide Autorengruppen die Wichtigkeit eines positiven und konstruktiven *Feedbacks*. Ntshalintshali et al. empfehlen außerdem, die *Fehler* als solche zu benennen und eine Formulierung des Feedbacks zu wählen, die „den Lernenden bewusst macht, dass das, was sie als nächstes lesen werden, vermutlich in Konflikt zu ihrem bestehenden Wissen stehen wird“ (ebd., S. 2166, Übersetzung durch Erstautorin). Auch Gusukama et al. [Gu18] empfehlen dieses Vorgehen, so dass auch wir diesem folgen. Die *Rückmeldungen* zu den Aufgaben werden daher so gestaltet, dass sie nicht nur Empfehlungen zu den nächsten Lernschritten geben, sondern auch die Fehler erläutern – oder im Fall der richtigen Lösung ein positives Feedback geben. Eine derartige Rückmeldung ist nicht nur hilfreich zur individuellen Adaption des Lernweges, ein direktes Feedback nach jeder Aufgabe, das an der Lösung orientiert ist, erhöht auch die Motivation und das Durchhaltevermögen der Studierenden [Ma20] [MOB13].

3.3 Konstruktion der Lernsituation und Aufgabenstellungen

Als Lernsituation und Kontextualisierung wurde die *„Entwicklung der Datenbank für ein hauseigenes Ticketsystem namens ‚doTics'"* eines fiktiven Unternehmens gewählt. Zur Umsetzung des Datenbank-Entwicklungsprozesses[8] wurden insgesamt 12 Szenarien mit unterschiedlichen Aufgabenstellungen entwickelt, Abb. 2 zeigt einen Ausschnitt zum Thema *Datendefinition / SQL*.

Anschließend wurden die verschiedenen Abschnitte des bestehenden Lehrtextes den Szenarien der Lernsituation zugeordnet. Die Aufgabenstellungen wurden so entwickelt, dass nur Inhalte, die Fachinformatiker/inne/n typischerweise nur teilweise bekannt oder als schwierig angesehen sind, thematisiert werden. Zu Inhalten, die nicht Teil der Ausbildung sind, wurden keine Aufgabenstellungen entwickelt. Es zeigte sich, dass neben dem Ablauf des Entwicklungsprozesses bei der Empfehlung der Abfolge der Lernschritte immer wieder ein Abgleich mit der entwickelten Concept Map (vgl. Abb. 1) stattfinden muss, um die jeweiligen Lernvoraussetzungen zu erfüllen. Auf diese Weise erhalten die Fachinformatiker/innen speziell zu den für den Kurs relevanten Themen[9] des Datenbankentwicklungsprozesses auf den Kontext der Lernsituation bezogene Aufgabenstellungen.

„Ok, die Datenbank passt wie vereinbart zu unserem Softwarestand, sehr schön! Das heißt, vielleicht könntet ihr ein paar Testdatensätze anlegen, damit wir etwas haben, mit dem wir die Software testen können?" Mit diesen Worten steht Ihr Kollege Guido van Montyp bei Ihnen. Sie vereinbaren, sowohl Agenten als auch Kunden testweise anzulegen. Wie immer erhalten Sie hierbei Unterstützung durch die Auszubildende.

Sie legen den Kunden *Schorsch Schlau, schschlau@zpinky.com* an, der bei der Firma *ZimtPinky GmbH* arbeitet, die sich im *Salzweg 42* in *58090 Hagen*

„Schauen Sie mal, irgendetwas ist falsch bei meinen Agenten, ich bekomme immer Fehlermeldungen, dass die Datensätze nicht angelegt werden können. Können Sie mir sagen, was ich falsch mache? Hier ist meine Syntax:"

```
INSERT INTO agent
(agentID, agentFachabt, agentVorname, agentNachname )
VALUES
(4, ‚Marvin', ‚Robott'),
(‚Softwareentwicklung', ‚Arthur', ‚Tend')
```

Abb. 2: Ausschnitt des Texts zu Szenario 08: „Eintragen und Ändern von Daten mit SQL". Fehler ist hier semantisch: falsche Datentypen für *agentID* und *agentFachabt*; beides ist nur durch Analyse des DB-Modells zu erkennen

Eine Limitation des verwendeten Moodle-Systems ist, dass zum aktuellen Zeitpunkt nur Single- und Multiple-Choice-Aufgaben verwendet werden können. Um dennoch kompetenzorientierte Fragestellungen höherer Anforderungsniveaus zu formulieren, wurden verschiedene Arten von Fragetypen verwendet:

- *Zuordnungs- oder Sortieraufgaben:*
  Bspw. Beschriftung von Elementen eines ERM, oder die Ermittlung der Reihenfolge von Arbeitsschritten. Dies wird durch das Anbieten verschiedener Begriffspaare (z.B. Entity – Rechteck) oder Reihenfolgen von Prozessschritten realisiert, woraus die richtigen gewählt werden müssen.

- *Entwicklung und Beschreibung eines Modells:*
  Modelle sind nicht automatisierbar auswertbar. Diese werden daher von den Studierenden entwickelt, anschließend beantworten sie Fragen zu den Modelleigenschaften, z.B. Anzahl von Entities, Vorgehensschritte (z.B. Normalisierung), oder Struktureigenschaften (z.B. Entitytypen).

- *Analyse von SQL-Statements:*
  Die Studierenden erläutern die Funktion oder Ausgabe einer Abfrage. In einer anderen Variante müssen die Studierenden syntaktische oder semantische Fehler finden (vgl. Abb. 2), um damit ihr Verständnis von SQL zu überprüfen.

- *Synthese von SQL-Abfragen:*
  Die Studierenden wählen das richtige Statement zur Lösung eines gegebenen Problems.

In allen Fragetypen beschreiben die unterschiedlichen Distraktoren Fehlvorstellungen, die je nach Fehler zu verschiedenen Empfehlungen zur Weiterarbeit führen.

Themen, die nicht Teil der Ausbildung sind, wurden durch *Selbsteinschätzungen* abgefragt. Beispiel ist hier *„Ja, Ich kann Relationen und verschiedene Operationen der Relationenalgebra darstellen und zur Darstellung von Tupeln, Verbünden oder Abfragen verwenden."* oder *„Nein, …"*. Im ersten Fall wird empfohlen, zur Bestätigung die zugehörigen Übungsaufgaben des Lehrtextes zu bearbeiten, im zweiten Fall erfolgt der Verweis auf das entsprechende Kapitel im Material.

8 verwendete Struktur: Anforderungsanalyse – Konzeptioneller Entwurf – Logischer Entwurf – Datenbankdefinition – Physischer Entwurf – Implementierung und Einführung (– Wartung und Nutzung).

9 Aspekte der Fachinformatikerausbildung wie Benutzerverwaltung, Sicherheit oder Betreuen eines Datenbankservers sind nicht Teil des Kurses „Datenbanken", und werden daher ausgeklammert.

3.4 Umsetzung der adaptierten Lehr-Lernmaterialien in Moodle

Die für die Kontextualisierung der Adaption des Kurses „Datenbanken" entwickelte Lernsituation wurde im Anschluss an die inhaltliche Entwicklung mit den Lehrtexten zu einer Einheit zusammengeführt. Um den Cognitive Load der Studierenden nicht weiter zu erhöhen, wurde die Gestaltung der Druckmaterialen ohne Änderungen in einen Markup-Text übernommen, der den Studierenden vollständig zur Verfügung steht. Da teilnehmende Fachinformatiker/innen, die durch das adaptierte Material ihre Lernwege adaptieren, die gleiche Modulabschlussprüfung wie alle anderen Studierenden ablegen, ist dies auch ein rechtlich notwendiges Vorgehen. Eine sehr detaillierte Labelung erlaubt eine genaue Adressierung einzelner Textabschnitte, so dass auch weitere Unterstützung oder Weiterverwendung für andere Vorausbildungen leicht möglich sind (vgl. [DKS99]). Die Aufgaben selbst liegen als JSON-Dateien vor und werden zusammen mit der Beschreibung der Situation über Skripte aufgerufen, sobald die Lernenden an die entsprechend gelabelte Textstelle – in der Regel Überschriften – kommen.

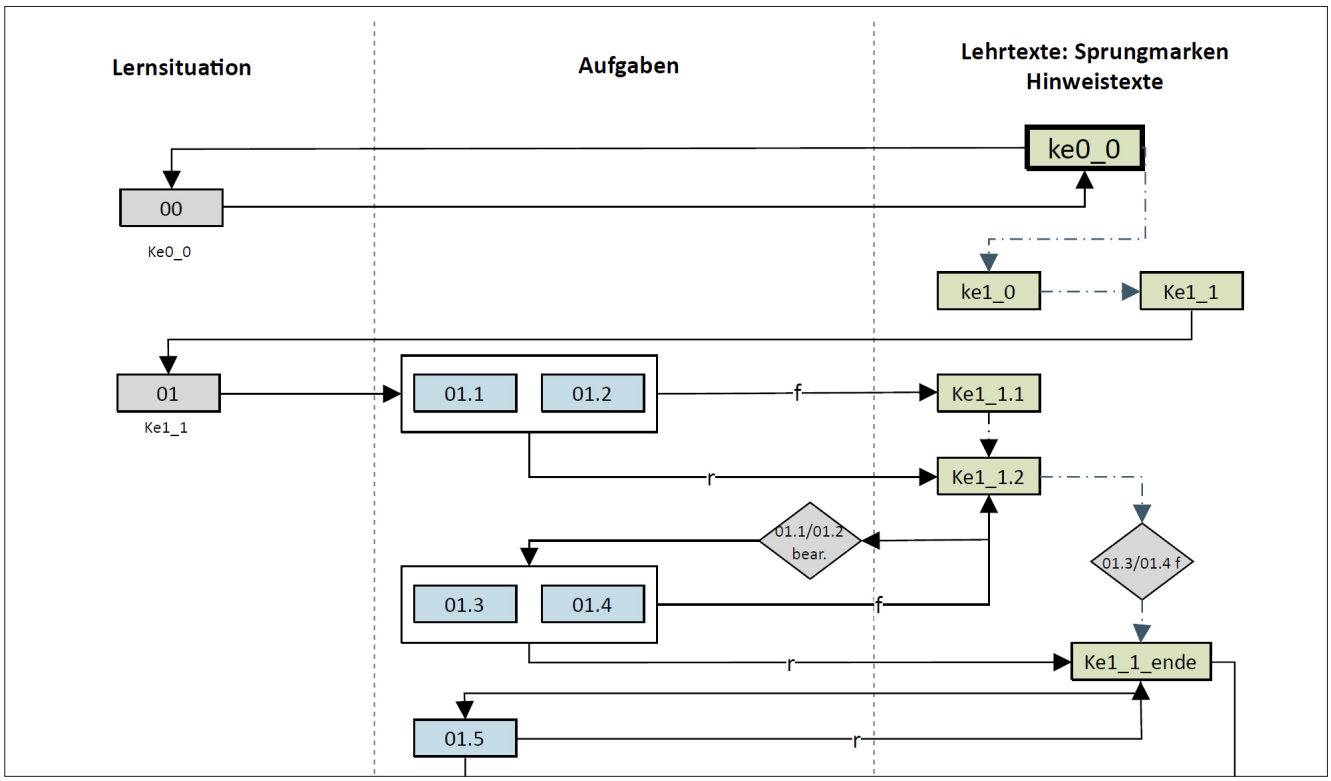

Abb. 3: Ausschnitt aus dem Ablaufdiagramm des adaptierten Lehr-Lernmaterials. Zu jedem Szenario der Lernsituation (00, 01) gibt es mindestens eine Aufgabe (01.1, 01.2, ...). Lehrtextstellenwerden durch ihre Label benannt. Linien: Verweise/Sprünge, gestrichtelte Linien: zu bearbeitende Textabschnitte.

Auch das Feedback wird Skript-basiert aufgerufen. Es besteht immer aus mindestens zwei Teilen: zunächst erhalten die Studierenden eine Information, ob sie die Aufgabe richtig gelöst haben (z.B. *„Sehr gut! Sie haben den Fehler in dieser komplexen Abfrage gefunden! Weiter so!" oder „Leider ist die Lösung falsch. Sie haben übersehen, dass [fachlicher Hinweis] notwendig ist."*). Falls sinnvoll, werden auch noch weitere Hinweise zur richtigen Lösung gegeben. Im nächsten Teil erhalten sie eine Empfehlung, wie Sie weiterarbeiten sollten (z.B. *„Wie es aussieht, haben Sie aus der Ausbildung noch umfassendes Wissen über das Thema. Daher können Sie Abschnitt ... überspringen und wir empfehlen Ihnen, bei Abschnitt ... mit dem nächsten Schritt der Datenbankentwicklung weiterzumachen." oder „Es scheint, dass Ihnen noch Wissen über ... fehlt. Daher empfehlen wir Ihnen, im nächsten Schritt Abschnitt ... zu bearbeiten, um das eventuell Fehlende aufzuarbeiten."*).

Dieser Ablauf wurde für die gesamte Lernsituation modelliert (vgl. Abb. 3) und anschließend in Moodle implementiert.

4 Adaption der individuellen Lernwege durch die Studierenden

Die *Adaption der individuellen Lernwege* wird durch die in Abschnitt 3.4. beschriebenen Rückmeldungen des Moodle-Systems initiiert. Folgen die Studierenden den Vorschlägen, vermeiden sie ein „doppeltes Lernen" von ihnen gut bekannten Inhalten, ohne dass sie selbst das Lehr-Lernmaterial diesbezüglich durchsuchen müssen.

Der Moodle-Kurs mit den adaptierten Lehr-Lernmaterialien wird allen Studierenden der Informatik zur Verfügung gestellt. Um die Adaption der eigenen Lernwege gezielt den Fachinformatiker/inne/n anzubieten, wird in einer einführenden Kurzbefragung die Vorbildung bzw. die Vorkenntnisse erhoben. Nur für die Personen, die angeben, eine Fachinformatikerausbildung zu besitzen, wird die Lernsituation sichtbar.

Der oder die Studierende beginnt, den adaptierten Lehrtext zu bearbeiten. Nach einer generellen Einführung in den Lehrtext erreicht man das erste Label, dort wird zunächst die Einführung zur Lernsituation (Situation 00, vgl. Abb. 3), anschließend die erste richtige Lernsituation (01) eingeblendet, anschließend hierzu die Aufgabenstellungen (01.1 und 01.2). Sobald die Studierenden die Aufgaben beantwortet haben, erhalten sie sofort ein Feedback einschließlich der Empfehlung zur Weiterarbeit. Da die Sicherheit bei der Beantwortung der Fragen nicht bekannt ist, wird ihnen freigestellt, ob sie der gegebenen Empfehlung folgen möchten – auch eine richtige Antwort kann geraten sein, eine falsche nur ein „Verklicken". Folgen sie keiner der Empfehlungen, lesen sie den Lehrtext linear und erhalten (allerdings nicht in der Reihenfolge des Datenbankentwicklungsprozesses) dennoch alle Szenarien der Lernsituation. Beantworten die Studierenden alle Fragen falsch, werden sie ebenfalls alle Teile des Lehrtextes bearbeiten, allerdings in der Reihenfolge des verwendeten Datenbankentwicklungsprozesses. Können sie alle Aufgabenstellungen aus dem Bereich der Ausbildung richtig lösen, bearbeiten sie nur die Bereiche des Lehr-Lernmaterials, die für die Gruppe der Fachinformatiker/innen neue Inhalte (vgl. Abschnitt 3.1) enthalten.

5 Zusammenfassung und Ausblick

In diesem Beitrag stellen wir eine Adaption von Lernwegen durch die Nutzung adaptierter Lern-Lernmaterialien für Studierende mit einer Ausbildung zum/r Fachinformatiker/in vor. Die Einführung von derartigen Lehr-Lernmaterialien stellt insbesondere für die große Gruppe von Studierenden, die im Beruf stehen, eine sicherlich wichtige Anerkennung ihrer bisherigen Berufserfahrung dar. Sie nutzen ihr Vorwissen und können je nach mitgebrachter Kompetenz ihren Workload für die Bearbeitung des Kurses signifikant senken, auch wenn natürlich die Bearbeitung der Lernsituation selbst Aufwand bedeutet. Dieser wird von den Autoren aber wesentlich geringer eingeschätzt, als die potentielle Ersparnis an Zeit und Lernaufwand, sowie der Zugewinn an Motivation, die durch die Wertschätzung der beruflichen Tätigkeit erfolgen kann.

Im aktuellen Sommersemester wird die Implementierung abgeschlossen, anschließend umfassend getestet und optimiert. Dem Gedanken der Design Science Research folgend, wird in einem weiteren Zyklus das Material weiterentwickelt und danach auch allen Studierenden mit Vorkenntnissen unabhängig von ihrer Vorbildung zur Adaption auf individueller Ebene bereitgestellt. Sollte dieser Kurs erfolgreich angenommen werden, werden weitere Kurse der FernUniversität in Hagen auf ähnlicher Weise adaptiert, zunächst in der Fakultät für Mathematik und Informatik.

Eine große Chance, mehr über unsere Studierenden zu erfahren, stellt die verwendete Plattform dar: die Studierenden stimmen zu, dass automatisiert umfassend Daten über ihre Nutzung des Systems erhoben werden. Durch den Einsatz entsprechender Tools hoffen wir erkennen zu können, wie die Studierenden mit der Adaption umgehen, ob es Unterschiede zwischen den Studierenden auf Gruppenebene gibt oder ob sich unabhängig von der Ausbildung Cluster von Lernern identifizieren lassen – und was zu beachten ist, wenn die Adaption auf weitere Kurse der FernUniversität in Hagen übertragen werden soll.

Das Projekt "Durchlässigkeit" wird gefördert durch den *Stifterverband* im Rahmen eines *Fellowship für Innovation in der Hochschullehre*.

Literatur

[Be19] Benning, A.; Bischoff, W.; Dörr, T.; Dreyer, M.; Fähndrich, S.; Jost, C.; Müskens, W.; Musil, A.; Pape, A.; Preusker, C.; Wiese, M.; Wilms, A.: Anrechnung an Hochschulen: Organisation – Durchführung – Qualitätssicherung. Hochschulrektorenkonferenz, Berlin, 2019.

[BM20] BMWI für Wirtschaft und Energie: Verordnung über die Berufsausbildung zum Fachinformatiker und zur Fachinformatikerin (Fachinformatikerausbildungsverordnung – FIAusbV) vom 28. Februar 2020, 2020.

[BS98] Bader, R.; Schäfer, B.: Lernfelder gestalten. Vom komplexen Handlungsfeld zur didaktisch strukturierten Lernsituation. Die berufsbildende Schule 50/7–8, S. 229–234, 1998.

[Bu20] Bundesministerium für Bildung und Forschung (BMBF): Berufsbildungsgesetz (BBiG), Berlin, 2020.

[CSD94] Chi, M. T.; Slotta, J. D.; De Leeuw, N.: From things to processes: A theory of conceptual change for learning science concepts. Learning and Instruction 4/1, S. 27–43, 1994.

[De14] Deutsches Zentrum für Hochschul- und Wissenschaftsforschung (GmbH): ANKOM – Übergänge von der beruflichen in die hochschulische Bildung. http://ankom.dzhw.eu/. Aufruf am 01.09.2015, 2014.

[De16] Desel, J.; Falkenberg, E.; Forbrig, P.; Kastens, U.; Koubek, J.; Magenheim, J.; Siegel, G.; Weicker, K.; Zukunft, O.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016). Gesellschaft für Informatik e.V., Bonn, 2016.

[DH16] Drechsler, A.; Hevner, A.: A four-cycle model of IS design science research: capturing the dynamic nature of IS artifact design. In: Breakthroughs and Emerging Insights from Ongoing Design Science Projects: 11th International Conference on Design Science Research in Information Systems and Technology (DESRIST) 2016. DESRIST 2016, S. 1–8, 2016.

[DKS99] Desel, J.; Klein, M.; Stucky, W.: Virtuelle Kurse durch Wiederverwendung didaktischer Lehrmodule. Institut für Angewandte Informatik und Formale Beschreibungsverfahren an der Universität Karlsruhe, Karlsruhe, 1999.

[Fe21] FernUniversität in Hagen: Hochschulstatistik, [aufgerufen: 2021-05-04], 2021, url: www.fernuni-hagen.de/uniintern/organisation/statistik/.

[Gu18] Gusukuma, L.; Bart, A. C.; Kafura, D.; Ernst, J.: Misconception-Driven Feedback: Results from an Experimental Study. In: Proceedings of the 2018 ACM Conference on International Computing Education Research. ICER '18, Association for Computing Machinery, Espoo, Finland, S. 160–168, 2018.

[KS01] Kremer, H.; Sloane, P.: Lernfelder Implementieren. Eusl, Paderborn, 2001.

[Ku02] Kultusministerkonferenz (KMK): Anrechnung von außerhalb des Hochschulwesens erworbenen Kenntnissen und Fähigkeiten auf ein Hochschulstudium (Beschluss der Kultusministerkonferenz, 28.06. 2002), 2002.

[Ku19] Kultusministerkonferenz (KMK: Rahmenlehrplan für die Ausbildungsberufe Fachinformatiker und Fachinformatikerin, IT-System-Elektroniker und ITSystem-Elektronikerin. KMK Referat für berufliche Bildung, Bonn, 2019.

[Ma20] Marwan, S.; Gao, G.; Fisk, S.; Price, T.W.; Barnes, T.: Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In: Proceedings of the 2020 ACM Conference on International Computing Education Research. ICER '20, Association for Computing Machinery, Virtual Event, New Zealand, S. 194–203, 2020.

[MOB13] Mitrovic, A.; Ohlsson, S.; Barrow, D. K.: The effect of positive feedback in a constraint-based intelligent tutoring system. Computers & Education 60/1, S. 264–272, 2013.

[NC20] Ntshalintshali, G. M.; Clariana, R. B.: Paraphrasing refutation text and knowledge form: examples from repairing relational database design misconceptions. Educational Technology Research and Development, S. 1–19, 2020.

[OND21] Opel, S.; Netzer, C. M.; Desel, J.: AsTRA – Assessment Tool for Recognition and Adaptation of Prior Professional Experience and Vocational Training. In: OCCE 2021 DTEL – ifip TC3 Conference (in print). Tampere, Finnland, 2021.

[Op20] Opel, S.: Entwicklung eines arbeitsprozessorientierten Kompetenzstrukturmodells für die Ausbildung zum Fachinformatiker bzw. zur Fachinformatikerin, Diss., University of Duisburg-Essen, Sep. 2020.

[Po20] Poulsen, S.; Butler, L.; Alawini, A.; Herman, G. L.: Insights from Student Solutions to SQL Homework Problems. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE '20, Association for Computing Machinery, Trondheim, Norway, S. 404–410, 2020.

[Re12] Reich, K.: Konstruktivistische Didaktik, das Lehr- und Studienbuch mit Online-Methodenpool. Beltz, Weinheim; Basel, 2012.

[Sw11] Sweller, J.: Cognitive load theory. In: Psychology of learning and motivation. Bd. 55, Elsevier, S. 37–76, 2011.

Individuelle Anrechnung außerhochschulisch erworbener Kompetenzen am Beispiel eines informatischen Studiengangs

Hoai Nam Huynh[1], Uwe Elsholz[1], Simone Opel[2]

Abstract:
Die Anrechnung beruflich erworbenenen Kompetenzen auf ein Hochschulstudium ist ein Ansatz, Durchlässigkeit im Bildungssystem zu fördern. Dieser gewinnt in Zeiten des Fachkräftemangels und einer sich schnell verändernden Arbeitswelt zunehmend an Bedeutung. Die Anrechnung bedeutet für Studierende eine Studienzeitverkürzung und bietet individuellere Studienverläufe an, die sich an die jeweiligen Biografien anpassen. Der Beitrag zeigt anhand eines Beispiels, wie das individuelle Anrechnungsverfahren im Studiengang der Informatik umgesetzt werden kann. Neben möglichen Hürden wird außerdem die Bedeutung individueller Anrechnungen in informatiknahen Studiengängen beleuchtet werden.

Keywords:
Individuelle Anrechnung; Portfolio; Durchlässigkeit; Hochschule; Anrechnungsverfahren

1 Einführung

Lebenslanges Lernen wird in der Informatik in unterschiedlichen Formen thematisiert. Oft werden damit IT-Weiterbildungen nach beruflichen Erstausbildungen verknüpft. Auch in der Hochschulbildung taucht das Thema verstärkt auf. Während 2008 darunter die Rolle der Hochschule und deren Beitrag zur IT-Weiterbildung an Hochschulen diskutiert wurde [Al08], wird zunehmend auch die Durchlässigkeit und Verzahnung zwischen verschiedenen Bildungsbereichen berücksichtigt. So greifen die jüngsten „Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen" der GI explizit die bildungspolitische Forderung auf, die Durchlässigkeit zwischen der beruflichen und akademischen Bildung zu fördern [De16, S. 53]. Damit wird auch an einen Beschluss der Kultusministerkonferenz (KMK) von 2002 [Ku02] angeschlossen, der die Anrechnung außerhochschulisch erworbener Kompetenzen regelt und dessen Ausgestaltung in der Verantwortung der jeweiligen Hochschulen liegt. Anrechnungen erlauben eine Studienzeitverkürzung, sollen die Schwelle für einen Studieneinstieg senken und die Attraktivität einer Studienaufnahme steigern. Frühere Programme wie NEXUS, ANKOM und „Aufstieg durch Bildung: offene Hochschulen", verfolgten vergleichbare Ziele (vgl. u.a. [Ce16b]; [Fr15]).

Der vorliegende Beitrag greift die oben formulierten Bestrebungen innerhalb der Informatik auf, Bildungsdurchlässigkeit im Übergang zur Hochschule zu fördern und stellt die Umsetzung der Anrechnung, speziell der sog. individuellen, beispielhaft vor. Der Beitrag verfolgt dabei folgende Fragestellung: *Wie kann die individuelle Anrechnung von außerhochschulisch erworbenen Kompetenzen in der Informatik strukturiert gestaltet werden?*

Dafür werden allgemeine Ziele der individuellen Anrechnung vorgestellt. Der Beitrag skizziert anhand eines laufenden Projekts die Lösungsansätze für einen Studiengang B.Sc. Informatik. Abschließend werden Herausforderungen und Lösungsansätze diskutiert und die Relevanz für informatiknahe Studiengänge erörtert.

1 Beide Autoren: FernUniversität in Hagen, Fakultät KSW, Universitätsstraße 33, 58097 Hagen, hoai-nam.huynh | uwe.elsholz@fernuni-hagen.de

2 FernUniversität in Hagen, Fakultät M+I, Universitätsstraße 27, 58097 Hagen, simone.opel@fernuni-hagen.de

2 Ziele, Arten und Probleme der Anrechnung

Es existieren verschiedene Wege der Anrechnung von außerhochschulisch erworbenen Kompetenzen. Erstens kann man von „pauschaler Anrechnung" sprechen, die es ermöglicht, aufgrund von Zertifikaten und einer Gleichwertigkeitsprüfung der relevanten Ordnungsmittel Studieninhalte anzurechnen. Meist bezieht sich das pauschale Anrechnungsverfahren auf formale Zertifikate (vgl. [ML18]). In informatiknahen Studiengängen sind damit etwa Berufsabschlüsse zum/zur Fachinformatiker/in gemeint. Das zweite Verfahren ist jenes der „individuellen Anrechnung" durch Prüfung des Einzelfalls. Die individuelle Variante eignet sich für Anrechnungen von Kompetenzen, die in der Berufspraxis etwa durch die Mitarbeit und durch die Leitung von Projekten erworben wurden. Man spricht dann von informell erworbenen Kompetenzen. Zusätzlich können in der individuellen Betrachtung auch nicht-formell erworbene Kompetenzen berücksichtigt werden. Diese haben nicht zu einem formalen Bildungsabschluss geführt, wurden aber trotzdem zertifiziert (z.B. im Rahmen von betrieblichen Fort- und Weiterbildungen).

Um die genannten außerhochschulisch, also nicht-formell und informell, erworbenen Kompetenzen anzurechnen, hat sich international das Portfolioverfahren etabliert [Ce16a]. Im europäischen Raum sind insbesondere Frankreich, Deutschland, die Länder Nordeuropas, sowie Österreich und Schweiz aktiv und verfügen über breite Erfahrungswerte, die über den Hochschulkontext hinausgehen (vgl. [Gu20], S. 464). Für Studierende bedeutet das Portfolioverfahren, dass sie dazu aufgefordert werden, ihre eigenen Leistungen zu belegen und diese mit den Qualifikationszielen eines Moduls in Bezug zu setzen.

2.1 Individuelle Anrechnung im Informatikstudium – ein Beispiel

Im Projektkontext, in dem die Entwicklung des individuellen Anrechnungsverfahrens verankert ist, wird mithilfe unterschiedlicher Ansätze versucht, die Durchlässigkeit von Studierenden im Fach Informatik im Übergang von Ausbildung zum Bachelorstudium zu erhöhen. Neben der pauschalen Anrechnung ist die individuelle Anrechnung die zweite Anrechnungsart, auf die nun näher eingegangen werden soll.
Den Kernbestandteil der individuellen Anrechnung stellt dabei ein *Portfolioverfahren* dar. Die Studierenden erstellen dabei eine Gegenüberstellung, durch die der Bezug zwischen den bereits erworbenen Kompetenzen und den erwarteten Kompetenzen eines Moduls erkennbar werden soll. Dabei werden Studierende beratend unterstützt. Letztlich muss der Bezug im Portfolio jedoch in Eigenleistung hergestellt werden und Selbstauskünfte sind von aussagekräftigen Nachweisen zu belegen (bspw. Arbeitszeugnisse). Die erwähnte Eigenleistung der Anrechnungsinteressierten ist die Grundlage der Beurteilung durch Prüfungsamt und Modulverantwortliche. Erscheinen die Selbstauskünfte nicht ausreichend für eine Beurteilung, kann ein ergänzendes Fachgespräch geführt werden. Gleichzeitig dienen diese Fachgespräche als zusätzliche Validierungsmaßnahmen der subjektiven Selbstbeschreibungen von Studierenden.
Mit diesem Vorgehen soll Studierenden in erster Linie eine klare Struktur und transparente Vorgehensweise kommuniziert werden. Für die Hochschulen sind hingegen Zuverlässigkeit und Präzision der gebotenen Informationen vorrangig. Für diese Zwecke wurde in interdisziplinärer Arbeit ein Leitfaden inklusive einem beispielhaften Anwendungsfall eines Portfolios erarbeitet. Es wurde dabei das Modul „Einführung in die Objektorientierte Programmierung" (Abb. 1) gewählt.

Der Auszug innerhalb der Abbildung zeigt, welchen Grad der Detailtiefe die Beschreibung eigener Lernergebnisse besitzen muss und wie hoch die Anforderungen für eine gelungene Anrechnung einzuschätzen sind. Die Anlagen, auf die rekurriert wird, sind ein Arbeitszeugnis (1) und eine Präsentation mit Konzeptdarstellungen (2).

Bei der Erstellung des beispielhaften Anwendungsfalls wurden zuvor Überlegungen angestellt, welches Modul sich zur Veranschaulichung innerhalb von Leitfäden eignet. Die Auswahl dieses Moduls begründet sich darin, dass unter den Studierenden ein hoher Anteil vorhanden ist, der Arbeitserfahrungen im Bereich der objektorientierten Programmierung nachweisen kann. Es wird daher erwartet, dass Anfragen zur Anrechnung des Moduls „Einführung in die Objektorientierte Programmierung" verhältnismäßig häufig von Studierenden gestellt werden. Die Auswahl dieses speziellen Moduls im Leitfaden soll in dem Sinne zu einer hohen Quote von gut beurteilbaren Portfolios beitragen und die Einführung des Portfolioverfahrens in der Breite erleichtern.

Lernergebnisse des Moduls	Eigene Lernergebnisse	Nachweise
Nach erfolgreicher Bearbeitung des Moduls haben die Studierenden ein Grundverständnis der Konzepte der objektorientierten Programmierung	In meiner mehrjährigen Tätigkeit als Softwareentwickler habe ich selbstständig Anwendungen in der Spiele-Branche entworfen, implementiert und getest (Anlage 1). Dabei umfasste die Code-Basis, innerhalb derer ich gearbeitet habe, einen umfangreichen, nach objektorientierten Prinzipien gestalteten C++-Quellcode. In diesem waren auch die gängigen Konzepte der objektorientierten Programmierung umgesetzt. Das reichte von grundlegenden Konzepten wie Objekten, Klassen, Vererbung und Datenkapselung bis zur der Definition und Verwendung komplexer Schnittstellen und zur eigenen Implementierung üblicher Entwurfsmuster (in meinem Fall beispielsweise „abstrakte Fabrik“ und „Singleton“) (Anlage 2). Das Gesamtsystem bestand aus zwei Software-Komponenten, die unabhängig voneinander auf unterschiedlicher Hardware liefen (Komponente 1: Spielsystemsteuerung, Komponente 2: Repräsentation des Spiels mit GUI). Diese kommunizierten untereinander und mit externen Hardware-Komponenten über einen Bus. Dabei musste die Synchronisation sichergestellt und theoretisch mögliche Deadlocks mussten verhindert werden. Insofern kann ich auch grundlegende Probleme verteilter und paralleler Systeme klären.	Anlage 1 Anlage 2

Abb. 1: Ausschnitt aus einem beispielhaften Portfolio zur Anrechnung des Moduls „Objektorientierte Programmierung“.

Was im gezeigten Ausschnitt des Anwendungsbeispiels aufgrund des großen Gesamtumfangs nicht gezeigt wird, sind weitere Lernergebnisse und Lerninhalte des Moduls. Auch zu diesen Anforderungen, die zusätzliche Aufschlüsse zum Umfang, Tiefe und Konkretisierung der Inhalte geben, muss ein Bezug hergestellt werden.

Dieses eingesetzte Verfahren, mit Hilfe von Portfolios Kompetenzen auf einen Studiengang der Informatik anzurechnen, ist jedoch durchaus anspruchsvoll. Zu den Hürden des Portfolioverfahrens gehört die nicht vorauszusetzende Akzeptanz des Portfolios als Methode. Für Studierende bedeutet dies, dass sie nicht ohne Weiteres den wahrscheinlich neuartigen Anforderungen an einem qualitativ hochwertigen Portfolio gerecht werden können. Eine gelungene, zielgruppenorientierte Beratung wird im Portfolioverfahren als notwendig betrachtet und wird im Projektkontext abschließend evaluiert. Ziel der Evaluation ist es, den Erfolg der Beratung zu überprüfen. Dabei sollen die unterschiedlichen Informationsangebote auf Transparenz, Anschaulichkeit und adressatengerechte Sprache geprüft werden.

3 Potenziale individuelle Anrechnungsverfahren in informatiknahen Studiengängen

Die skizzierten Chancen und Herausforderungen, die mit der individuellen Anrechnung verbunden sind, lassen sich fachübergreifend wiederfinden. Im Folgenden soll erörtert werden, warum sich gerade in Studiengängen der Informatik eine verstärkte Anstrengung dieser speziellen Anrechnungsart lohnenswert sei. In der informatiknahen Domäne lässt sich in den letzten Jahren ein konstant hohes Weiterbildungsverhalten beobachten. Diese Bildungsprozesse sind vorwiegend dem non-formalen Kontext zuzuordnen, also vorwiegend aus innerbetrieblichen Weiterbildungen stammend (vgl. [St17]). Berufliche Abschlüsse, die im Deutschen Qualifikationsrahmen [Bu13] dem Niveau eines Bachelorabschlusses nahekommen, werden kaum in Anspruch genommen (siehe IT-Professionals). Aufgrund der für die Informatik typischen Dynamik der Kompetenzprofile ist davon auszugehen, dass weiterhin mit hohen Weiterbildungsaktivitäten versucht wird, kurzfristig darauf zu reagieren. Unternehmen und Hochschulen sind dazu angehalten, regelmäßig Lehrinhalte zu evaluieren und, wenn es sinnvoll erscheint, anzupassen. Ein anhaltend hohes Interesse an einer (individuellen) Anrechnung und entsprechenden Anträgen wird für wahrscheinlich gehalten – sofern Informationen und entsprechende Beratung dazu für alle Studierende verfügbar sind.

Im Übergang von der beruflichen Informatikausbildung zum -studium kann davon ausgegangen werden, dass sich Berufe des Informatikbereichs „in Bezug auf den Fachinhalt ihrer Lehr- und Ausbildungspläne nur gering von denen des ersten Teils des Bachelorstudiums Informatik unterscheiden“ ([ODM16, S. 67]). Die Nähe von informatiknahen Ausbildung und Teilen des Hochschulstudiums wurde auch in der Empfehlung des AK DQR ausgedrückt, in der erfolgslos empfohlen wurde, die IT-Berufe auf der Stufe 5 einzuordnen [Ar11].

4 Zusammenfassung und Ausblick

Die individuelle Anrechnung von außerhochschulisch erworbenen Kompetenzen kann dazu beitragen, die Durchlässigkeit im Übergang zur Hochschule zu erhöhen und damit zu mehr Bildungsgerechtigkeit beitragen. Sie kann in der Informatik gelingen, wenn für Studierende der Eindruck von Transparenz und Anschaulichkeit entsteht. Aufwand und Erfolgschancen eines Antrags werden durch zur Verfügung gestellte Informationsmaterialien realistisch abschätzbar. Strukturierte Verfahren mithilfe eines Portfolios geben Studierenden eine zusätzliche Chance, die Aussagekraft ihrer eingereichten Unterlagen zu erhöhen. Für Hochschulmitarbeitende, die die Anträge prüfen, können Entscheidungen systematischer und inhaltlich informierter getroffen werden als ohne Portfolios. Innerhalb eines Projektrahmens wurden für diesen Zweck Dokumente in interdisziplinärer Zusammenarbeit erstellt. Das Portfolioverfahren bietet die Chance, dass vermehrt vor dem Studium erworbene Kompetenzen einen strukturierten und qualitätsgesicherten Eingang in das Hochschulstudium finden. Eine Öffnung von Hochschulen wird konkretisiert, indem auch Kompetenzen angerechnet werden, die nicht aus traditionellen Bildungsbiografien entstammen.

Die Realisierung individueller Anrechnungsverfahren ist jedoch mit unterschiedlichen Hürden verbunden. So lassen sich Schwierigkeiten in der Qualität der Selbstauskünfte durch Studierende finden, welche als Bewertungsgrundlage der Gleichwertigkeitsprüfung fungieren. Dieses hohe Maß an Eigenverantwortung der Anrechnungsinteressierten stellt sowohl für Studierende wie Hochschulmitarbeitende eine ungewohnte Herausforderung dar. Auch hier soll mithilfe der Evaluation empirisches Material zur Frage gewonnen werden, wie gut Studierende bei dieser Art von Eigenleistung unterstützt werden. Die GI geht davon aus, dass Vertrauen in die gegenseitigen Bildungssysteme ein Baustein für das Gelingen ist ([De16, S. 53]). In diesem speziellen Kontext ist ein Vertrauen in Portfolios als Instrument der Anrechnung zu entwickeln, aber auch ein Vertrauen in die Vergleichbarkeit von beruflicher und hochschulischer Bildung in definierten Bereichen im Allgemeinen. Diese Vertrauensbildung sollte mit fachlicher Kompetenz in der Beurteilung für beide Bildungsbereiche und personellen Ressourcen begleitet werden. Studiengänge wie die Informatik können so mit strukturierten Anrechnungsverfahren einen spürbaren Teil dazu beitragen, den Öffnungsprozess der Hochschulen für breitere Studierendengruppen aktiv mitzugestalten.

Das Projekt „Durchlässigkeit" wird gefördert durch den Stifterverband im Rahmen eines Fellowship für Innovation in der Hochschullehre.

Literatur

[Al08] Altmann, W.: Lebenslanges Lernen in der Informatik. Gesellschaft für Informatik, Bonn, 2008.

[Ar11] Arbeitskreis DQR: DQR – Deutscher Qualifikationsrahmen für lebenslanges Lernen verabschiedet vom AK DQR am 22. März 2011. Bund-Länder-Koordinierungsstelle für den DQR, Berlin, 2011.

[Bu13] Bund-Länder-Koordinierungsstelle für Lebenslanges Lernen: Handbuch zum Deutschen Qualifikationsrahmen: Struktur – Zuordnungen – Verfahren – Zuständigkeiten, 2013.

[Ce16a] Cedefop: Europäische Leitlinien für die Validierung nicht formalen und informellen Lernens. Amt für Veröffentlichungen der Europäischen Union, Luxemburg, 2016.

[Ce16b] Cendon, E.; Mörth, A.; Pellert, A.; Al-Ani, A.; Basner, T.; Brohm, M.; Coghlan, D.; Costley, C.; Dehnbostel, P.; Deiser, R.; Elsholz, U.; Filloque, J.-M.; Fischer, R.; Grassl, R.; Heese, R.; Köster, K.; Major, D.; Pavlicek, Y.; Pechar, H.; Power, T. M.; Rheinländer, K.; Robes, J.; Rundnagel, H.; Schäfer, M.; Schenker-Wicki, A.; Schiedhelm, M.; Schiller, E.; Schöne, S.; Stettner, J.; Tait, A.; Talbot, J.; Wanken, S.; Zechlin, L.; Zuber-Skerritt, O.: Theorie und Praxis verzahnen Lebenslanges Lernen an Hochschulen: Ergebnisse der wissenschaftlichen Begleitung des Bund-Länder-Wettbewerbs Aufstieg durch Bildung: offene Hochschulen. 1. Auflage. Waxmann, Münster New York, 2016.

[De16] Desel, J.; Falkenberg, E.; Forbrig, P.; Kastens, U.; Koubek, J.; Magenheim, J.; Siegel, G.; Weicker, K.; Zukunft, O.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016). Gesellschaft für Informatik e.V., Bonn, 2016.

[Fr15] Freitag, W. K.; Buhr, R.; Danzeglocke, E.-M.; Schröder, S.; Völk, D.: Übergänge gestalten: Durchlässigkeit zwischen beruflicher und hochschulischer Bildung erhöhen. Waxmann Verlag, Münster, 2015.

[Gu20] Gutschow, K.: Validierung und Anerkennung informell erworbener Kompetenzen. In (Arnold, R.; Lipsmeier, A.; M., R., Hrsg.): Handbuch Berufsbildung. Springer VS, Wiesbaden, 2020.

[Ku02] Kultusministerkonferenz (KMK): Anrechnung von außerhalb des Hochschulwesens erworbenen Kenntnissen und Fähigkeiten auf ein Hochschulstudium (Beschluss der Kultusministerkonferenz, 28.06. 2002), 2002.

[ML18] Müskens, W.; Lübben, S.: Die Anrechnung non-formalen und informellen Lernens auf Hochschulstudiengänge in Deutschland. Zeitschrift für Weiterbildungsforschung 41/2, S. 109–124, 2018.

[ODM16] Opel, S.; Desel, J.; Magenheim, J.: Aspekte horizontaler und vertikaler beruflicher Informatikausbildung. In (Dreher, R.; Jenewein, K.; Neustock, U.; Schwenger, U., Hrsg.): Wandel der technischen Berufsbildung: Ansätze und Zukunftsperspektiven. Bd. 41. Berufsbildung, Arbeit und Innovation, W. Bertelsmann Verlag, Bielefeld, S. 67–76, 2016.

[St17] Statistisches Bundesamt (Destatis): Berufliche Weiterbildung in Unternehmen. Fünfte Europäische Erhebung über die berufliche Weiterbildung in Unternehmen (CVTS5), [Aufgerufen am 19.05.2021], 2017,
url: www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Bildung-Forschung-Kultur/Weiterbildung/Publikationen/Downloads-Weiterbildung/weiterbildung-unternehmen-5215201159004.pdf.

Nutzung der Personas-Methode zum Umgang mit der Heterogenität von Informatik-Studierenden

Dietrich Gerstenberger[1], Felix Winkelnkemper[2], Carsten Schulte[3]

Abstract:
Informatik-Studiengänge verzeichnen hohe Abbruchquoten innerhalb der ersten beiden Semester, die neben Leistungsdefiziten häufig mit Motivationsproblemen begründet werden. Eine Ursache dafür, dass trotz jahrelanger Bemühungen um bessere Lehre und motivationsfördernde Maßnahmen diese Situation im Wesentlichen unverändert bleibt, könnte darin liegen, dass nicht die eine Maßnahme oder der eine Ansatz hilft, sondern vielmehr unterschiedliche Maßnahmen für eine heterogene Studierendenschaft benötigt werden. Bisher findet sich kaum Forschung zu differenzierten Studierendentypen in der Informatik. Wir stellen daher in dieser Arbeit einen Ansatz und erste Ergebnisse vor, die Heterogenität der Informatik-Studierenden zu ergründen. Um die große Anzahl von Studierenden auf eine überschaubare Anzahl von Typen mit gleichartigen Merkmalen zu reduzieren, wird dazu die im Produktmanagement bewährte Personas-Methode adaptiert. Im Rahmen einer Befragung von 170 Informatik-Studierenden konnten hierzu bereits einige Personas mit unterschiedlichen Merkmalshäufungen ausgearbeitet werden, die eine gute Basis bilden, um differenzierte Interventionen zur Förderung und Motivation der Informatik-Studierenden daraus abzuleiten.

Keywords:
Informatik; Studium; Studienabbruch; Lernzentrum; Interventionen; Personas; Identität

1 Einleitung

In einer Studie des DZHW [He17] berichten Heublein et al. über die Gründe für vorzeitige Studienabbrüche an deutschen Hochschulen. 33% der Befragten gaben Leistungsprobleme als ausschlaggebenden Grund für den Abbruch ihres mathematisch-naturwissenschaftlichen Studiums an, während immerhin 17% ihren Studienabbruch mit mangelnder Studienmotivation begründeten. In einem Vergleich der Abbruchzahlen der Abschlussjahrgänge von 2010, 2011 und 2014 bescheinigen Heublein et al. eine Verbesserung an den Universitäten in den Ingenieurwissenschaften von 48% auf 32%, während die Fachbereiche der Mathematik/Naturwissenschaften, zu denen die Informatik gezählt wird, eine konstante Abbruchquote von 39% aufwiesen.

Um die Abbruchquoten zu senken, setzt man im Fachbereich Informatik neben anderen Angeboten auf die Institutionalisierung extracurricularer Zusatz- und Hilfsangebote für Informatik-Studierende. So richtete auch das Institut für Informatik der Universität Paderborn ein Lernzentrum ein, um den Studierenden sowohl fachspezifische Unterstützung zu bieten als auch motivierende Angebote machen zu können. Mittlerweile ist das Angebot umfangreich. Die Notwendigkeit, das Angebot stetig an sich ändernde, individuelle Anforderungen der heterogenen Studierendenschaft anzupassen, steht bisher noch wenig im Fokus und basiert nicht auf einer systematisch erhobenen Grundlage. Zwar ist es durchaus wahrscheinlich, dass nicht alle Studierenden dieselben Bedürfnisse haben und nicht alle mit denselben Angeboten gleich gut gefördert werden können, doch fehlt es an Wissen, welche unterschiedlichen „Typen" oder „Identitäten" von Informatik-Studierenden es gibt, welche spezifischen Bedürfnisse diese mitbringen und wie Hilfsangebote für die einzelnen Typen aussehen können.

Im Rahmen eines systematischen Literaturreviews zum Thema „Identity in Computer Science Education", dessen Teil mit Fokus auf K12 bereits veröffentlicht wurde [Gr21], fanden sich nur wenige Studien, die Hinweise auf differenzierte Computer Science Identitäten lieferten. Für den Hochschulbereich arbeiteten Peters und Pears in einer ihrer Studien zwei Gruppen von Informatik-Studie-

1 Universität Paderborn, DDI, Fürstenallee 11, D-33102 Paderborn, dietrich.gerstenberger@uni-paderborn.de

2 Universität Paderborn, DDI, Fürstenallee 11, D-33102 Paderborn, felix.winkelnkemper@uni-paderborn.de

3 Universität Paderborn, DDI, Fürstenallee 11, D-33102 Paderborn, carsten.schulte@uni-paderborn.de

renden heraus [PP13]: Eine Gruppe hat den Fokus auf die Anwendung des Computers, für die andere Gruppe haben Technologien und deren gesellschaftliche Implikationen eine größere Bedeutung. Davis et al. dagegen fokussieren sich in ihrer Studie auf das Stereotyp des Nerds [DYB14] und zeigen auf, dass unter dieser Personengruppe Identitäten differenzierter betrachtet werden müssen. Die Autoren unterscheiden hier beispielhaft die Varianten CS-Mathe-Nerd, CS-Spiele-Nerd und CSSport-Nerd. Große-Bölting et al. gruppieren die Informatik-Studierenden gemäß ihren Vorstellungen über den Fachbereich Informatik [GSM19]. Sie finden dabei Typen wie Mathematiker:in, Techniker:in oder Creator:in, wobei letzterer z.B. in der Informatik ein Mittel sieht, um Einfluss auf die Umwelt nehmen zu können.

Die hier skizzierten Untersuchungen zu Informatik-Identitäten beschreiben diese auf unterschiedlichen Ebenen und leisten für sich genommen noch keine kohärente Unterstützung bei der Entwicklung angepasster Unterstützungsangebote. Es fehlen systematische Untersuchungen und die dazu notwendige Methodik, die unterschiedlichen Erwartungen, Fähigkeiten und Bedürfnissen der Informatik–Studierenden strukturiert, schnell und mit geringem Aufwand erfassen zu können. Die hier vorgestellte Studie und das dazu entwickelte Studiendesign sollen dazu einen Beitrag leisten.

Es ergeben sich folgende Fragestellungen:

RQ1: Wie kann man unterschiedliche Typen von Informatik-Studierenden mit ihren spezifischen Merkmalen strukturiert ermitteln und beschreiben?
RQ2: Können differenzierte Lern- und Unterstützungsbedarfe aus dieser Typisierung abgeleitet werden?

2 Theoretische Grundlagen

Die Ansprüche und Vorkenntnisse von Studierenden ändern sich ebenso wie das Berufs- und Wissenschaftsfeld der Informatik stetig. Analysen zur Optimierung des Unterstützungsangebots, die die genannten Forschungsfragen beantworten können, müssen daher regelmäßig angestellt werden. Aus dieser Notwendigkeit lässt sich der Bedarf nach einem leichtgewichtigen Instrument ableiten, das es ermöglicht, das Angebot des Lernzentrums kontinuierlich anhand der Bedürfnisse der Studierenden überprüfen zu können. Eine komplexe qualitative Analyse wäre für diesen Anspruch zu schwerfällig, ein einfacher Ankreuzfragebogen hingegen nicht aussagekräftig und vor allem nicht flexibel genug. Als Mittelweg steht im Mittelpunkt des hier vorgestellten Ansatzes die Idee, die in der Wirtschaft bewährte Personas-Methode zur Typisierung von Nutzer:innen [CRC07] auf das Themenfeld der universitären Lernunterstützung zu übertragen und zudem so zu adaptieren, dass sie ressourcenschonend regelmäßig angewendet werden kann, um auch mit den zu erwartenden (niedrigen) Teilnehmendenzahlen aussagekräftige Eindrücke über die Bedürfnisse verschiedener Informatiker:innen-Typen zu erlangen. Diese Idee basiert auf einem Artikel der Autorinnen Nicola Marsden und Monika Pröbster, die in einer Studie den Zusammenhang von Personas und den Identitäten der durch sie verkörperten Personengruppen herausarbeiten [MP19]. Sie beschreiben dort ein Verfahren, um die verschiedenen Identitäten der zu betrachtenden Nutzer:innen in den Eigenschaften und Verhaltensmustern der Personas abzubilden. Mittelfristig soll versucht werden, aus den spezifischen Personas der Informatik-Studierenden auf deren differenzierte Identitäten zu schließen.

2.1 Personas-Methode

Zum Repertoire der Anforderungserhebung im Bereich der Produktentwicklung gehört die von Alan Cooper entwickelte Personas-Methode [Co04], deren Ziel es ist, aus Anwenderbefragungen mittels qualitativer Analyse und Clustering eine sehr viel kleinere Anzahl von prototypischen Nutzer:innen zu generieren. Diese sogenannten Personas werden dann durchgängig vom Entwicklungsteam eingesetzt, um Produkt-Konzepte darauf aufzubauen und Prototypen des Produkts gegen die Anforderungen und Bedürfnisse der Personas zu prüfen. Am Anfang der Personas-Methode steht dabei immer eine umfangreiche Datenerhebung durch Interviews mit potenziellen Nutzer:innen. Ergänzt werden diese Daten ggf. um Informationen aus Marktanalysen, Daten aus Literatur-Reviews etc. Im nächsten Schritt werden aus diesen Daten Merkmale extrahiert. Man analysiert dazu die Aussagen der Personen bzgl. der Bereiche Demographie, familiäres Umfeld, Aktivitäten, Ansichten, Begabungen, Motivationen und Fähigkeiten, wodurch sich deduktiv die ersten Kategorien für eine qualitative Analyse ergeben. Die gefundenen Merkmale werden dann jedem Befragten zugeschrieben und für ihn ganz spezifisch mit einem Wert belegt (z.B. die Eigenschaft „Computerkenntnisse“ auf einer Skala [Anfänger, . . . Profi]). Im Folgenden werden die diskreten Werte aller Befragten betrachtet, um Cluster zu bestimmen. Cooper empfiehlt, bei einer Übereinstimmung in sechs bis acht Merkmalen ein sogenanntes „behavior pattern“ zu vermerken [CRC07].

Hat man mehrere solcher Verhaltensmuster identifiziert, bilden diese die Basis für eine erste Sammlung von Personas. Sofern die gefundenen Verhaltensmuster einer Persona noch nicht alle untersuchten Kategorien abdecken, werden in einem Folgeschritt fehlende Merkmale mit Daten aus den Befragungen der Nutzer:innen synthetisiert. Falls Lücken bleiben, kann es nötig werden,

weitere Daten zu erheben und gegebenenfalls den Fokus in Interviews auf bestimmte Personentypen und deren speziellen Eigenschaften zu legen. Erst eine derart vollständige Beschreibung lässt die Persona lebendig werden und erlaubt es, sie als prototypische Nutzer:in im Produktdesign zu verwenden.

Dadurch, dass Personas möglichst idealtypisch sein sollen, ergibt sich zwangsläufig die Gefahr, dass sie ein Stereotyp beschreiben. Phil und Susan Turner diskutieren diesen Sachverhalt, betonen jedoch, dass Stereotypen in den Personas nicht grundsätzlich schlecht seien [TT11]. Hill et al. halten in diesem Zusammenhang aber insbesondere typische Zuschreibungen in Bezug auf Geschlechter und Herkünfte für problematisch [Hi17] und verweisen unter anderem auf Studien von Marsden et al., die z.B. in [MLB14] diese Problematik speziell aus der Sicht der Geschlechterrollen beleuchten und Handlungsempfehlungen für eine genderbewusste Entwicklung und Nutzung von Personas geben. Wichtig sind demnach sowohl die vorurteilsfreie Wahrnehmung als auch die Gruppenzugehörigkeit der Personen, die an der Erstellung der Personen beteiligt sind. Das Entwicklungsteam sollte daher alle zu berücksichtigenden Geschlechterrollen abdecken. Weiterhin müssen bei der Ausarbeitung der Personas soziale Identitäten, Detaillierungsgrad und das Vorkommen stereotypischer Merkmale immer wieder kritisch hinterfragt werden.

2.2 Fachspezifische Identität von Informatik Studierenden

Die Grundlage der Personas-Methode ist das Vorhandensein umfangreicher Daten potenzieller Nutzer:innen. Klassischerweise werden diese wie oben beschrieben durch Interviews und eine spezifische Auswertungsmethodik erhoben. Durch selbstbeschreibende Interviews an genug Daten über eine Vielzahl von Personas zu kommen ist allerdings aufwändig und steht dem Anspruch, leichtgewichtig und damit regelmäßig anwendbar zu sein, entgegen. Um auch mit wenigen Teilnehmer:innen an vielschichtige Daten zu kommen wurde daher, wie eingangs erwähnt ein Weg gesucht, die Studierenden im Rahmen eines Fragebogens über die verschiedenen Informatiker:innen-Typen ihres Fachbereichs reflektieren zu lassen. Die Antworten eines einzelnen Teilnehmenden liefern dann die Grundlage für mehrere Personas. In vier Vorstudien wurde das Design des Fragebogens weiterentwickelt und er enthält nun (unter anderen) die folgenden Fragen:

Item G01Q01: „Als erstes darfst Du jetzt ein paar Informatiker:innen-Typen aufzählen, denen Du hier immer wieder begegnest:"
Item G03Q06ff.: „Jetzt wäre es schön, wenn Du einen dieser Typen ‚mit Leben füllst', indem Du ein paar Details hinzufügst:"
Item G03Q18ff.: „Welcher Typ bist Du? Beschreibe jetzt bitte auch Deinen eigenen Typ anhand der folgenden Fragestellungen:"

Zu den Items ab G03Q06 bzw. G03Q18 gibt der Fragebogen Impulse durch verschiedene Leitfragen zu folgenden Kategorien: Beruf/Bildung/Familie, Demographie, Identifikatoren, Erwartungen/Ziele, Bedürfnisse/Probleme/Herausforderungen und Fähigkeiten.

Die theoretische Grundlage zu den gewählten Fragestellungen basiert auf den Beschreibungen der fachspezifischen Identität, wie man sie im soziokulturellen Strang der Identitätstheorie finden kann, wo besonders auf die soziale Situiertheit des Individuums in einer Gemeinschaft fokussiert wird (siehe auch [LW91], [We99], [Ge00]). Gee beschreibt die Identität z.B. als „being recognized as a certain ‚kind of person,' in a given context" [Ge00]. Ein zentrales Element der Definition von Identität nach Gee ist das Umfeld der Person, das die spezielle Identität wahrnimmt und dadurch erst zur Identität werden lässt. Eine Identität kann also, ähnlich wie bei Lave und Wenger [LW91] beschrieben, nicht isoliert entwickelt werden, sondern erfordert immer die Teilnahme an einer Gruppe von Menschen, die sich dadurch gegenseitig prägen. Die Autorinnen führen den Begriff „Communities of Practice" (CoP) ein, den Etienne Wenger weiter ausarbeitet [We99] und dabei herausstellt, dass es eine klare Verknüpfung von Identitätsbildung und der gemeinsamen praktischen Arbeit mit „Gleichgesinnten" gibt. Der zu entwickelnde Fragebogen sollte es den Befragten also erlauben, sowohl die eigene Identität als Informatiker:in zu reflektieren, als auch die Möglichkeit bieten, mitzuteilen, wie man die Kommiliton:innen als Informatiker:innen wahrnimmt.

3 Methode

Im Folgenden soll die von uns adaptierte Personas-Methode beschrieben werden, um Typen von Informatik-Studierenden finden und benennen zu können. Den Startpunkt bildet der oben erwähnte Fragebogen, in dem die Studierenden zunächst angeleitet werden, einige „Typen" zu benennen, denen sie in ihrem Fachbereich immer wieder begegnen. Einen dieser Typen sollen sie dann anhand vorgegebener Kategorien und Leitfragen detailliert beschreiben. Diese Kategorien bilden die Basis für die Erstellung der gesuchten Informatiker:innen-Personas und umfassen Demographie, das familiäre Umfeld, Motivationen, Fähigkeiten, Ziele, Erwartungen, Probleme und Herausforderungen der zu beschreibenden Archetypen. In einem abschließenden Teil des Fragebogens sollen die Teilnehmenden dann ihren eigenen Typ (bzw. ihre Identität) benennen und anhand der Leitfragen ausformulieren. Die beiden Fragestellungen sind eng an Gee's Definition von Identity angelehnt [Ge00] und sollen den Befragten ermöglichen, zu reflektieren, wie sie andere Studierende bzw. wie sie sich selbst als Informatiker:in sehen.

Im ersten Auswertungsschritt werden die von den Befragten aufgezählten Typen zusammengetragen und Cluster gebildet. Die Adaption der Personas-Methode, die sie in ein leichtgewichtigeres Instrument verwandelt, ist es, diese Cluster als erste Sammlung von Personas zu betrachten, die in den Folgeschritten gemäß der bekannten Methodik weiter verfeinert werden. Dazu müssen die Daten aus den detaillierten Beschreibungen der Informatiker:innen-Typen qualitativ analysiert werden, wobei sich durch die thematisch strikt gegliederten Leitfragen schon die initialen Kategorien ergeben, die im Folgenden induktiv um weitere Kategorien, Sub-Kategorien und deren Merkmale erweitert werden. Anschließend werden die Kodierungen der detaillierten Beschreibung pro Cluster betrachtet, um anhand der Merkmalshäufungen die Eigenschaften der aus den Clustern entstandenen Personas zu beschreiben. Eventuell bedarf es dann noch dem abschließenden Schritt der Personas-Methode, in dem fehlende Details einiger Kategorien nacherfasst werden müssen.

Im Wintersemester 2020/21 wurde der Fragebogen mit den Teilnehmenden zweier Erstsemester-Veranstaltungen erprobt. Zum einen bearbeiteten ihn 45 Teilnehmende des Informatik-Vorkurses der Universität Paderborn. Weiterhin konnten 125 Erstsemester-Studierende einer Informatik-Nebenfach-Veranstaltung der Universität Kiel gewonnen werden, Informatiker:innen-Typen zu beschreiben, denen sie in ihrem Fachbereich begegnet sind. Die Aufforderung, zunächst eine Reihe von Typen aufzuzählen, führte bereits zu einer Liste mit 178 Nennungen, die im nächsten Schritt geclustert wurden. Aus verschiedenen Bezeichnungen wie z.B. Programmierer, Hobby-Programmierer, Freizeitprogrammierer, der Coder oder der Programmierorientierte ergab sich im ersten Schritt der Auswertung der Cluster „Programmierer:in".

Die Analyse der Nennungen führte zudem zu einem Cluster „beruflich", in dem im Rahmen der Auswertung zahlreiche Nennungen eingefügt wurden, mit der die Studierenden eine spätere Berufsrichtung der Informatiker bezeichneten (z.B. Fachinformatiker:in, Systemadministrator:in, Webdesigner:in, Netzwerkadministrator:in). Dieser erinnert an differenzierte *Professional Identities*, wie sie z.B. schon in Studien zur beruflichen Identität von Informatikern mit „IT-Admin, Helpdesk, QA Testers and Developers" benannt wurden [TSS19].

Im nächsten Schritt wurden die von den Befragten detailliert beschriebenen Informatiker:innen-Typen in separate Dokumente überführt und in eine Software zur qualitativen Datenanalyse importiert. Angefangen mit den vorgegebenen Kategorien aus den Leitfragen des Fragebogens wurden zunächst alle Aussagen zu den Typen, für die im ersten Schritt ein Cluster gefunden werden konnte, deduktiv und in einem zweiten Schritt dann induktiv mit weiteren Subkategorien und Merkmalen kodiert. Fast alle Typen, zu denen mehr als drei detaillierte Beschreibungen verfügbar waren, zeigten so spezifische Merkmalshäufungen, dass sie benutzt werden konnten, um erste Versionen sogenannter Sedcards, dem Endprodukt der Personas-Methode, anzulegen. Zur Vermeidung stereotyper Zuschreibungen in Bezug auf Geschlecht und Herkunft wurde dabei ein Vorschlag von Hill et al. [Hi17] aufgegriffen, die Persona-Beschreibung mit Bildern mehrerer möglicher Nutzer:innen verschiedener Geschlechter zu hinterlegen. Zusammen mit einer gender-neutralen Beschreibung aller Merkmale soll dadurch im weiteren Prozess eine möglichst wenig durch entsprechende Stereotype überlagerte Entwicklung ermöglicht werden. Aus unserer Analyse der Daten ergeben sich so zunächst die folgenden fünf Informatiker:innen-Typen:

- Programmierer:in (24 Nennungen, 8-mal detailliert)
- Nerd (18 Nennungen, 5-mal detailliert)
- Informatik-Interessent:in (12 Nennungen, 5-mal detailliert)
- Informatik-Profi (11 Nennungen, 5-mal detailliert)
- Hardwareexpert:in (10 Nennungen, 4-mal detailliert)

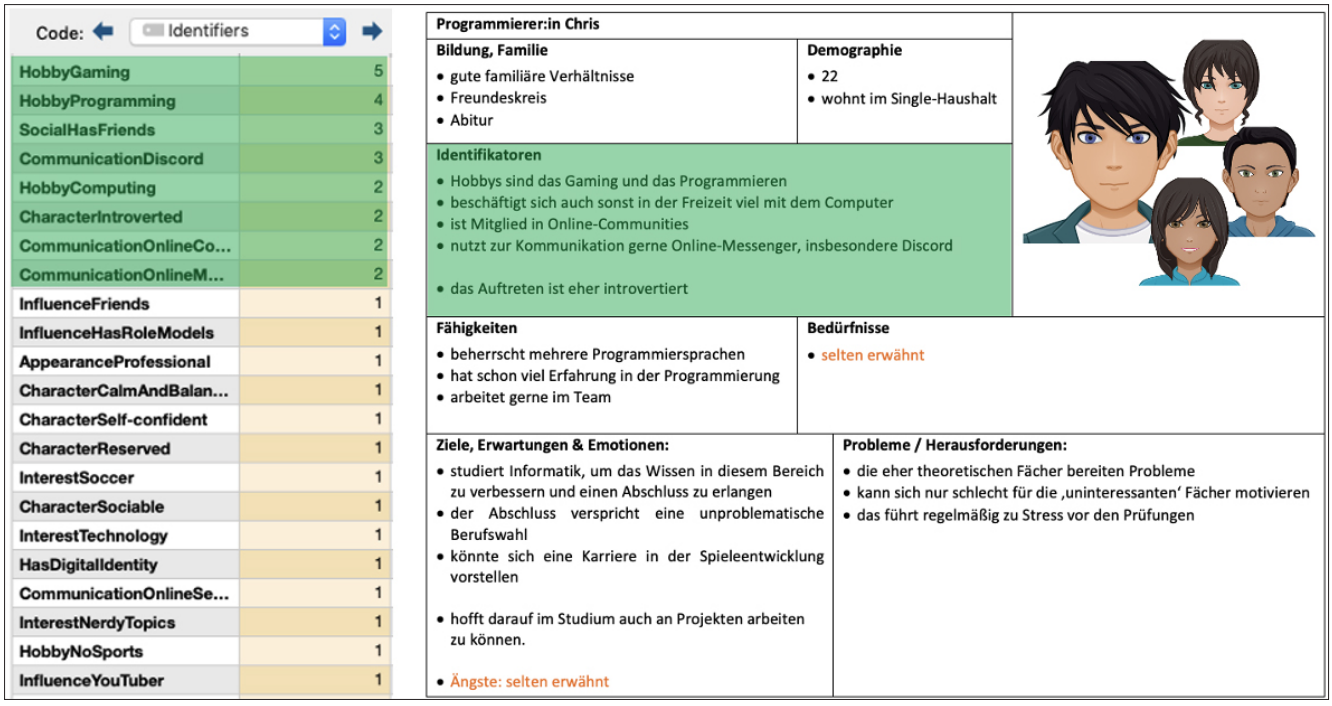

Abb. 1: Persona Entwicklung - Programmierer:in Chris

Die vorliegenden Detailbeschreibungen zum Typ Programmierer:in, Informatik-Interessent:in und Hardwareexpert:in werden in der Zusammenstellung sowohl aus Aussagen der Probanden zum eigenen Typ als auch durch Aussagen zum Typ von anderen Informatik-Studierenden gewonnen. Die Typen Nerd und Profi wurden von den Probanden jedoch nur aus ihrer Sicht auf die anderen Studierenden detailliert beschrieben. Die meisten Details ließen sich zur Programmierer:in ausarbeiten, da in diesem Cluster mit acht detaillierten Beschreibungen die größte Anzahl an Fragebögen zur Auswertung verfügbar waren. Beispielhaft wird in der Abbildung 1 die Auswertung der Codes zu den Identifikatoren (grün hinterlegt) des Typen „Programmierer:in" gezeigt. Aus einer Liste von insgesamt 61 Merkmalen für Identifikatoren wurden in Summe über alle Beschreibungen 22 verschiedene Codes vergeben, von denen 8 in mindestens zwei verschiedenen Beschreibungen genutzt wurden und als Grundlage für die Zusammenstellung der Identifikatoren der Persona „Programmierer:in Chris" dienen.

Um die Belastbarkeit der so entwickelten Persona „Programmierer:in Chris" zu verifizieren, sollte versucht werden, ob neue hinzugefügte Beschreibungen des Typs „Programmierer:in" zur Veränderung dieser Sedcard führen würden. Dazu wurden sechs Teilnehmende eines Mentoring-Programms befragt und es ergaben sich zwei zusätzliche Beschreibungen dieses Typen. Beide Fragebögen wurden der Auswertung hinzugefügt und nach denselben Richtlinien kodiert. Die Abbildung 2 zeigt beispielhaft die acht Merkmale, die für die Programmierer:in mehr als einmal genannt wurden und daher für die Detaillierung der Persona benutzt wurden. 14 weitere Merkmale wurden ursprünglich nur einmal als Identifier kodiert, weshalb sie nicht zur Beschreibung der Persona genutzt wurden.

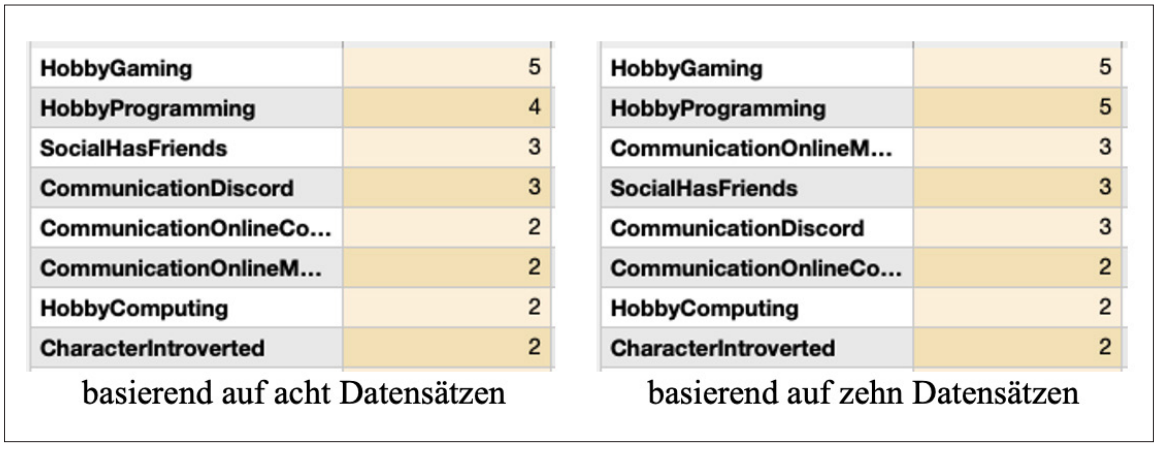

Abb. 2: Merkmalshäufungen in der Kategorie „Indikatoren"

Man kann in Abbildung 2 erkennen, dass die Auswahl der Merkmale zur Beschreibung der Persona sich nicht verändert hat. Durch das Hinzufügen zusätzlicher Daten wurde keines der bisher unberücksichtigten Merkmale verstärkt. Nur innerhalb der bereits ausgewählten Merkmale gab es weitere Verstärkungen durch die neuen Daten. Die Erweiterung der zugrundeliegenden Daten-um25% führte also zu keiner Veränderung der Persona Sedcard. Im Falle der „Programmierer:in" sieht es aus, als hätten acht bis zehn detaillierte Beschreibungen eines Typs ausgereicht, um daraus eine belastbare Persona zu generieren.

4 Diskussion

Als Antwort auf die Fragestellung RQ1 („Wie kann man die unterschiedlichen Typen von Informatik-Studierenden mit ihren spezifischen Merkmalen strukturiert ermitteln und beschreiben?") wurde die seit Jahrzehnten bewährte Personas-Methode adaptiert. Sie liefert eine Typisierung von Informatik-Studierenden gemäß Merkmals-Kategorien, die durch geeignete Leitfragen gesteuert wurden, um Aussagen über die Kenntnisse und Fähigkeiten aber auch über die Erwartungen und Bedürfnisse der Studierenden zu bekommen. Alleine die simple Nennung bekannter Typen durch die Befragten deutete schon eine Vielzahl verschiedener Informatiker:innen-Typen an. Auch die folgende Analyse der Merkmale zeigte spezifische Häufungen, die eine differenzierte Benennung der Typen rechtfertigen.

Betrachtet man im Detail, wie die Beschreibungen der verschiedenen Sedcards entstanden sind, sieht man, dass nur die Codes mit den meisten Nennungen zu einer Erwähnung innerhalb der Sedcard führten. Zum Beispiel bescheinigten dem Typ „Programmierer:in" vier der acht Befragten, dass er/sie in einem Single-Haushalt lebt, während alle fünf Befragten dem „Nerd" als „vermutlich bei den Eltern lebend" einordneten. Genauso wurde bei allen anderen Kodierungen vorgegangen. Um die einzelnen Personas mit weiteren Details lebendiger zu machen, müssen in einer weiteren Iteration alle Beschreibungen eines Typs im Zusammenhang betrachtet werden, um zu entscheiden, ob vielleicht noch ein paar der seltener erwähnten Merkmale zur Charakterisierung des Typs hinzugezogen werden sollten.

Wie eingangs erwähnt haben Studien aus dem Bereich der fachspezifischen Identitätsforschung einige differenzierte Betrachtungen von Informatiker:innen-Identitäten aufgezeigt. Das hier vorgestellte Verfahren lieferte schon bei der ersten Anwendung einen reichen Fundus an sehr spezifischen Personas. Weiteres Datenmaterial wird nun dafür sorgen, diese Personas noch weiter zu detaillieren, neue Personas zu entdecken und vielleicht einige der Typen zusammenzufassen. Die von Marsden und Pröbster [MP19] beschriebene Beziehung von Personas und Identität kann dann dazu beitragen, einen ebenso umfangreichen Satz an differenzierten Informatiker:innen-Identitäten bereitstellen zu können.

Eine Gefahr der hier vorgestellten Adaption der Personas-Methode liegt in einer Tendenz zu Stereotypen. Die Befragten sollten zum einen Ihnen bekannte Informatiker:innen-Typen beschreiben, was viele dazu verleitet hat, Klischees wie z.B. Nerd oder Gamer:in anzuführen. Mit den detaillierten Beschreibungen gerade dieser Typen wird man mit Bedacht umgehen müssen, weil sie natürlich angefüllt sind mit stereotypischen Merkmalen, die ebendiesen Personen üblicherweise zugeschrieben werden. Allerdings lässt sich auch kaum leugnen, dass diesen Personas in vielen Aspekten entsprechende Studierende im Fachbereich Informatik tatsächlich existieren und diese ebenso in ihrem Studienvorhaben unterstützt werden müssen. Da die Befragten im Rahmen des hier beschriebenen Prozesses zusätzlich gebeten werden, ihren eigenen Informatiker:innen-Typ detailliert zu beschreiben, kann erwartet werden, durchaus auch reflektiertere und damit potenziell wertvollere Informationen für die Personas zu finden. Schon die Bezeichnungen der eigenen Typen wurden hier meist ‚vorsichtiger' gewählt (z.B. der/die Begeisterte, der/die Unsichere, Systematiker:in, Perfektionist:in), dennoch konnten wir unter den Beschreibungen des eigenen Typs einige Gamer:innen und Hardwareexpert:innen entdecken. Eine höhere Anzahl an Befragten wird ergeben, ob sich auch Studierende selbst als Nerd charakterisieren. Aus zwanglosen Gesprächen im Umfeld von Lehre und Lernzentrum sind uns solche Selbstzuschreibungen zumindest bekannt.

Zwar dienten die bisher erhobenen Daten vornehmlich der Überprüfung der Methode, aus ihren Ergebnissen lassen sich allerdings schon jetzt Schlüsse ziehen, um die zweite Fragestellung (RQ2: „Können differenzierte Lern- und Unterstützungsbedarfe aus dieser Typisierung abgeleitet werden?") zu beantworten. Um jetzt Interventionen maßzuschneidern, müssen zum einen die Kategorien Bedürfnisse und Probleme/Herausforderungen der verschiedenen Personas betrachtet werden, aber auch die spezifischen Ziele/Erwartungen der Personas spielen hier eine wichtige Rolle. Man erkennt z.B., dass die Personas Programmierer:in und Nerd beim Arbeitsschritt der Typisierung deutliche Merkmalshäufungen im Bereich des Wunsches nach Projektarbeit hatten. Um insbesondere den hohen Studienabbruchquoten im ersten Jahr gerecht zu werden, wäre es zu wünschen, dass schon innerhalb der ersten beiden Semester Möglichkeiten zur Projektarbeit geschaffen werden. Zumindest extra-curricular kann eine Institution wie das Lernzentrum kurzfristig auf solche Bedarfe eingehen. In Zusammenarbeit mit umliegenden Firmen sollten sich Umgebungen für überschaubare IT-Projekte schaffen lassen, die den Wünschen der Studierenden entgegenkommen. Ein bereits umgesetzter Hackathon vor dem Hintergrund einer realen Produkteentwicklung hatte in diesem Zusammenhang bereits für sehr gutes Feedback gesorgt.

Im Unterschied dazu wurde dieses Bedürfnis bei den Typen „Hardwareexpert:in", „Profi" und „Informatik Interessent:in" nur sehr vereinzelt erwähnt und daher nicht in die spezifische Typbeschreibung übernommen. Ferner hebt sich der weniger oft beschriebene Typ „UI-Designer:in" deutlich von den anderen Personas ab. Hier werden weder der Wunsch nach Projektarbeit noch Probleme mit sozialen Kontakten thematisiert, jedoch zum Beispiel eine notwendige Unterstützung im mathematischen Bereich. Hier sollte die für Lernzentren typische fachliche Unterstützung ausreichend vorhanden sein, sofern es gelingt, die Studierenden auf dies Angebote aufmerksam zu machen und zur Nutzung dieser Angebote zu motivieren.

Diese konstruktive Perspektive ist schon dem ursprünglichen Personas-Konzept inhärent. So endet Coopers Beschreibung der Personas Methode [CRC07] nicht mit der Bereitstellung der endgültigen Persona–Beschreibungen. Diese sind vielmehr nur ein Teil einer umfassenden Vorgehensweise, Softwareprodukte für die Anforderungen eben dieser Personas zu entwerfen. Eine Orientierung an Coopers Verfahren kann also Hinweise liefern, wie basierend auf den Personas geeignete Interventionen entwickelt werden können.

Neben der bereits diskutierten Problematik der Personas-Methode, auch Stereotype der betrachteten Personengruppen zu erfassen, muss an dieser Stelle noch ein spezielles Problem in Bezug auf die Zusammenstellung der befragten Studierenden angeführt werden. Für den ersten Einsatz des adaptierten Personas-Fragebogens boten sich zwei Erstsemester-Veranstaltungen des Wintersemesters 2020 an. Zum einen hatten die Befragten durch die Corona-Situation zu diesem Zeitpunkt noch nicht viele Gelegenheiten, Bekanntschaften in Ihrem neuen Fachbereich zu schließen. Zum anderen basieren die durch geschlechter-neutrale SedCards dargestellten Personas momentan überwiegend auf männlichen Sichtweisen auf die Informatiker:innen Typen. In den kommenden Semestern sollen daher weitere Informatik-Studierende befragt werden, um den Datenbestand zu vergrößern und zu diversivizieren. Die dann folgenden Auswertungen müssen zeigen, ob man den Weg mit zusammengefassten Personas weitergehen kann, oder ob zumindest für einige Typen sehr spezifische Merkmale für eine Separation sprechen.

5 Schlussfolgerungen und Ausblick

Die adaptierte Personas-Methode führte schnell zu ersten Clustern, weil die Befragten selbst dazu animiert wurden, mögliche Typen zu benennen. Dadurch wurden die Schritte der Personas-Methode ersetzt, in denen zunächst Merkmalshäufungen bezüglich aller Einzelfälle gesucht wurden und bestimmte Kombinationen aus diesen Häufungen zu den Verhaltensmustern führten, die dann eine erste Auswahl von Personas ergaben. Die Aussagekraft dieser leichtgewichtigen Vorgehensweise gilt es nun zu überprüfen, indem zum Vergleich die klassische Typenbildung (z.B. nach Kluge/Kelle [KK10]) durchgeführt wird, ohne die von den Befragten selbst genannten Typen zur Hilfe zu nehmen. Die durch den initialen Schritt stark verkürzte Auswertung der Daten würde höhere Probandenzahlen ermöglichen und dadurch präzisere Beschreibungen der Personas ergeben. Ein langfristiges Ziel wäre es, anhand dieser detaillierten Personas Rating-Skalen zu entwickeln und Fragebögen abzuleiten, um mit deren Hilfe Studierende sehr schnell ihren Personas zuordnen zu können. Das könnte dann als Grundlage dienen, passende Angebote für die jeweiligen Typen zu entwickeln und anzubieten.

Die ersten Ergebnisse der adaptierten Personas-Methode versprechen, die Heterogenität der Informatik-Studierenden aus Sicht ihrer Erwartungen, Fähigkeiten und Bedürfnisse spezifischer betrachten zu können. Bisher versucht man dieser Heterogenität im Wesentlichen dadurch zu begegnen, in der Studieneingangsphase Vorkurse, Tutorien und unterstützende Angebote, wie die der Lernzentren einzurichten. Diese Mittel sollen dafür sorgen, das fachliche Niveau der neuen Studierenden anzugleichen. Es deutet sich aber an, dass sich hinter den sehr spezifischen Personas der Informatik-Studierenden ebenso differenzierte Identitäten von ihnen verbergen. In Studien wurde darüber hinaus gezeigt, dass die Ausprägung einer fachspezifischen Identität die Motivation und das Durchhaltevermögen der Studierenden verbessert (z.B. [UMH10], [PP13]). Die Fortführung der Persona-Methode und das daraus abgeleitete Design maßgeschneiderter Interventionen könnten dazu beitragen, die Identitäten der Studierenden zu stärken, dadurch Hilfestellung leisten und den Studienerfolg zu verbessern. Genau betrachtet würde das bedeuten, die Studierenden auch über die Studieneingangsphase hinaus in ihrer Heterogenität zu bestärken und diese sogar zu fördern.

Insbesondere im extra-curricularen Bereich, also z.B. im Umfeld eines fachspezifischen Lernzentrums, hat man die Flexibilität kurzfristig auf Bedarfe einzugehen. Weiterführende Studien müssten prüfen, ob die Nutzung dieser maßgeschneiderten Interventionen dazu beitragen kann, die differenzierten Identitäten weiter auszuprägen und schlussendlich, die Abbruchquoten zu verringern und einen langfristigen Studienerfolg zu sichern.

Literatur

[Co04] Cooper, A. et al.: The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity. Sams Indianapolis, 2004.

[CRC07] Cooper, A.; Reimann, R.; Cronin, D.: About face 3: the essentials of interaction design. John Wiley & Sons, 2007.

[DYB14] Davis, D.; Yuen, T.; Berland, M.: Multiple case study of nerd identity in a CS1 class. In: Proceedings of the 45th ACM technical symposium on Computer science education. S. 325–330, 2014.

[Ge00] Gee, J. P.: Chapter 3: Identity as an analytic lens for research in education. Review of research in education 25/1, S. 99–125, 2000.

[Gr21] Große-Bölting, G.; Gerstenberger, D.; Gildehaus, L.; Mühling, A.; Schulte, C.: Identity in K-12 Computer Education Research: A Systematic Literature Review. In: Proceedings of the 2021 ACM Conference on International Computing Education Research. ICER '21, Association for Computing Machinery, New York, NY, USA, 2021.

[GSM19] Große-Bölting, G.; Schneider, Y.; Mühling, A.: It's like computers speak a different language: Beginning Students' Conceptions of Computer Science. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research. S. 1–5, 2019.

[He17] Heublein, U.; Ebert, J.; Hutzsch, C.; Isleib, S.; König, R.; Richter, J.; Woisch, A.: Zwischen Studienerwartungen und Studienwirklichkeit. In: Forum Hochschule. Bd. 1. 1, S. 318, 2017.

[Hi17] Hill, C. G.; Haag, M.; Oleson, A.; Mendez, C.; Marsden, N.; Sarma, A.; Burnett, M.: Gender-Inclusiveness Personas vs. Stereotyping: Can we have it both ways? In: Proceedings of the 2017 chi conference on human factors in computing systems. S. 6658–6671, 2017.

[KK10] Kelle, U.; Kluge, S.: Vom Einzelfall zum Typus: Fallvergleich und Fallkontrastierung in der qualitativen Sozialforschung, 2010.

[LW91] Lave, J.; Wenger, E.: Situated learning: Legitimate peripheral participation. Cambridge university press, 1991.

[MLB14] Marsden, N.; Link, J.; Büllesfeld, E.: Personas und stereotype Geschlechterrollen. In: Gender-UseIT. De Gruyter Oldenbourg, S. 91–104, 2014.

[MP19] Marsden, N.; Pröbster, M.: Personas and Identity: Looking at Multiple Identities to Inform the Construction of Personas. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. S. 1–14, 2019.

[PP13] Peters, A.-K.; Pears, A.: Engagement in Computer Science and IT–What! A Matter of Identity? In: 2013 Learning and Teaching in Computing and Engineering. IEEE, S. 114–121, 2013.

[TSS19] Taylor-Smith, E.; Smith, S.; Smith, C.: Identity and Belonging for Graduate Apprenticeships in Computing: The experience of first cohort degree apprentices in Scotland. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education. S. 2–8, 2019.

[TT11] Turner, P.; Turner, S.: Is stereotyping inevitable when designing with personas? Design studies 32/1, S. 30–44, 2011.

[UMH10] Ulriksen, L.; Madsen, L. M.; Holmegaard Henriette, T.: What do we know about explanations for drop out/opt out among young people from STM higher education programmes? Studies in science education 46/2, S. 209–244, 2010.

[We99] Wenger, E.: Communities of practice: Learning, meaning, and identity. Cambridge university press, 1999.

Adaptierung von Beratungsangeboten auf der Basis von Erkenntnissen aus der Analyse von Studienverlaufsdaten

Axel Böttcher, Veronika Thurner, Tanja Häfner, Sarah Ottinger[1]

Abstract:
Viele Studierende stoßen im Rahmen ihres Informatik-Studiums auf Probleme und benötigen individuell bedarfsgerechte Unterstützung, um trotz gewisser Startschwierigkeiten ihr Studium erfolgreich zu Ende zu führen. In die damit verbundene Lern- bzw. Studienberatung fließen Empfehlungen zur weiteren Studienverlaufsplanung ein. Anhand einer Datenanalyse über den Prüfungsleistungsdaten der Studierenden überprüfen wir die hinter diesen Empfehlungen liegenden Hypothesen und leiten aus den dabei gewonnenen Erkenntnissen Konsequenzen für die Beratung ab. Insgesamt zeigt sich, dass sich nach den ersten Semestern ein mittlerer Bereich von Studierenden identifizieren lässt, bei denen Studienabbruch und Studienerfolg etwa gleich wahrscheinlich sind. Für diese Personengruppe ist Beratungsbedarf dringend gegeben. Gleichzeitig stößt die Datenanalyse auch an gewisse Grenzen, denn es zeigen sich insgesamt keine echt trennscharfen Muster, die frühzeitig im Studium eindeutig Erfolg oder Misserfolg prognostizieren. Dieses Ergebnis ist jedoch insofern erfreulich als es bedeutet, dass jede:r Studierende auch nach einem suboptimalen Start ins Studium noch eine Chance auf einen Abschluss hat.

Keywords:
Datenanalyse; Studienverläufe; Erfolgsmessung

1 Einleitung

Hohe Studienabbruchquoten sind ein Phänomen in vielen MINT-Studiengängen. Im Bereich der Informatik sind deutschlandweit Abbruchquoten von um die 45% an Universitäten und knapp 40% an Hochschulen für angewandte Wissenschaften verbreitet [DI18] (Tab. F4-1A). Gleichzeitig ist die Nachfrage nach Absolventinnen und Absolventen der Informatik auf dem Arbeitsmarkt seit Jahren ungebrochen hoch. Diejenigen Studierenden, die ihr Informatikstudium erfolgreich zu Ende führen, reichen bei Weitem nicht aus, um diesen Bedarf zu decken. Um dem entgegenzuwirken haben viele Hochschulen Maßnahmen ergriffen, um die Studierenden bedarfsgerecht zu unterstützen und so insgesamt mehr Studierende der Informatik zu einem erfolgreichen Abschluss zu führen.

Auch an der Fakultät für Informatik und Mathematik der Hochschule München wurden vielfältige Angebote etabliert, mit besonderem Fokus auf die Studieneingangsphase, um den Studierenden den Einstieg in den Studierprozess zu erleichtern. In diesem Rahmen wurden auch die Beratungsangebote an der Fakultät auf- und ausgebaut. Die Möglichkeiten zur fachlichen Beratung durch die Lehrenden sowie zur psychosozialen Beratung durch eine entsprechend ausgebildete Mitarbeiterin werden nun ergänzt durch eine Lernberatung für Studierende. Diese wird durchgeführt durch zwei Referentinnen für Lehren und Lernen, deren Stellen aus Studienzuschüssen geschaffen wurden. Die Referentinnen für Lehren und Lernen haben keine Lehrverpflichtung, treffen also nicht selbst in der Rolle als Lehrende auf die Studierenden. So ist sichergestellt, dass im Beratungskontext erworbenes Wissen über persönliche Details der Lernenden nicht eine objektive Beurteilung von studentischen Leistungen verfälscht.

Diese Beratungsangebote werden zunehmend stark nachgefragt, insbesondere auch von Studierenden im ersten oder zweiten Semester, deren Einstieg ins Studium eher holprig verlaufen ist, die aber motiviert sind ihr Studium weiterzuverfolgen und erfolgreich zu Ende zu bringen. Gemeinsame Ausgangsbasis der Beratungskonstellationen ist dabei die Existenz einer Problemsituation. Wie genau diese aussieht – und welche Lösungsmöglichkeiten ggf. offen stehen – ist dagegen hochgradig individuell, da jede Person als Hintergrund ihren individuellen Studienverlauf mit Erfolgen und Misserfolgen mitbringt, sowie ihr spezifisches Kompetenzprofil und ihre individuellen Persönlichkeitseigenschaften. Dieser Heterogenität müssen die Beratungsangebote Rechnung tragen und in Einzelgesprächen individuell passende

1 Hochschule München, Fakultät für Informatik und Mathematik, D-80335 München, vorname.nachname@hm.edu

Strategien für die nächsten Schritte im Ausbildungsprozess erarbeiten. Deren Bandbreite reicht von der Entwicklung geeigneter Strategien für ein erfolgreiches Weiterstudieren des initial gewählten Studienganges über einen Wechsel des Studienfaches bis hin zu einem frühzeitigen Studienabbruch, ggf. mit Aufnahme einer kaufmännischen oder gewerblichen Berufsausbildung als alternativem Werdegang.

Aufgrund der Vielschichtigkeit der Beratungskonstellationen ist eine individuell bedarfsgerechte Beratung eine hochkomplexe Aufgabe. Damit die Beratungen einen möglichst großen Nutzen bewirken (d. h. den betroffenen Studierenden das Finden und Wählen des individuell für sie passenden und gangbaren Weges ermöglichen) ist eine inhaltliche Systematisierung der Beratungsangebote erforderlich.

2 Ziele

Übergeordnetes Ziel ist also, diejenigen Studierenden mit Unterstützungsbedarf individuell bedarfsgerecht zu beraten und dadurch die Studienerfolgsquote zu verbessern.

Dazu sind zum einen diejenigen Studierenden zu identifizieren, die tatsächlich Unterstützungsbedarf haben. Zum anderen sind typische Erfolgsmuster und Best Practices herauszuarbeiten, die Studierende anwenden können, um ihr Studium wieder auf einen erfolgreichen Weg zu bringen. Darauf abgestimmt sind Heuristiken zu entwickeln, die in der Beratung helfen, aus der Menge der erarbeiteten Erfolgsmuster und Best Practices diejenigen als Empfehlung herauszufiltern, die für eine konkrete Person und deren konkrete Problemkonstellation tatsächlich individuell passend und zielführend sind und diese zu konkreten nächsten Schritten für die Gestaltung des eigenen Lernprozesses zu präzisieren.

Die bis dato etablierten Beratungsangebote adressieren auf der Micro-Ebene diverse „handwerkliche" Tipps zur Lernorganisation, wie beispielsweise Zeitmanagement, Selbstorganisation oder das Identifizieren des eigenen Lerntyps. Diese sind nun zu ergänzen um Empfehlungen auf der Macro-Ebene, d. h. der Gestaltung des individuellen Studienverlaufs derjenigen Studierenden, die wegen nicht angetretener bzw. nicht bestandener Prüfungen aus dem Regelverlauf herausfallen bzw. sich eine Welle von „Altlasten" aufbauen, die rein mengenmäßig nicht zusätzlich zum „normal" vorgesehenen Semesterprogramm zu bewältigen sind. Hier ist also individuell bedarfsgerecht zu entscheiden, welche Module bzw. Prüfungen in welcher Reihenfolge fokussiert werden sollen und was verschoben werden muss, um so das im Semester zu absolvierende Pensum in einem schaffbaren Rahmen zu halten. Diese Überlegungen müssen die existierenden verwaltungs- und prüfungsrechtlichen Regeln hinsichtlich Fristen für das erste Antreten oder das Wiederholen von Prüfungen berücksichtigen.

Ergänzend ist zu überprüfen, inwieweit häufiger auftretende Problemkonstellationen strukturell bedingt sind und wie diese ggf. durch entsprechende Änderungen in der Organisation der Studienangebote vermieden oder zumindest entschärft werden können.

3 Stand der Forschung

Die im Studierprozess anfallenden Daten werden von vielen Institutionen mit unterschiedlichen Zielrichtungen und unterschiedlichen Kontexten analysiert. So werden im Bereich Learning Analytics detaillierte Informationen zum Verhalten der Studierenden in einzelnen Kursen erhoben und ausgewertet. Diese Analysen sind sehr feingranular und häufig auf MOOCs bezogen. Ziel ist einerseits die Steuerung von Lehr-/Lernprozessen auf Ebene eines Moduls. Außerdem wird versucht, Leistungen sowie die Gefahr eines vorzeitigen Abbruchs der Arbeit am Modul zu prognostizieren; siehe dazu z. B. [FY15].

Andererseits werden in vielen Studien auf einer abstrakteren Ebene ganze Studienverläufe analysiert. Ziel ist dabei, Modelle für frühzeitige Prognosen des individuellen Studienerfolgs zu erstellen. Die Arbeiten von Hinkelmann et al. [HMT16], Berens et al. [Be18] sowie von Kemper et al. [KVW20] sind für die vorliegende Arbeit insofern besonders relevant, als sie mit Blick auf die Zielsetzung vergleichbare Ansätze verfolgen und sich hinsichtlich rechtlicher und organisatorischer Rahmenbedingungen in einem ähnlichen Kontext bewegen. Die Autorinnen und Autoren wenden verschiedene statistische Analyseverfahren oder Machine-Learning-Ansätze an, um anhand soziodemographischer Daten und Daten über Studienverläufe frühzeitig individuelle Wahrscheinlichkeiten für Erfolg oder Misserfolg im Studium zu erstellen. In der aktuell vorliegenden Literatur werden bislang allerdings nur die tatsächlich erfolgten Prüfungen, nicht jedoch die vorgesehenen Prüfungen in die Analysen einbezogen.

Mit diesen Mechanismen lässt sich bereits anhand der bei der Immatrikulation vorhandenen Daten ein Drop-out mit 67% Genauigkeit vorhersagen [Be18]. Darüber hinaus zeigt sich, dass nach dem ersten Fachsemester zu über 80% zutreffende Vorhersagen über Erfolg bzw. Misserfolg getroffen werden können [Be18, KVW20].

Als aus den Ergebnissen abzuleitende Handlungen und Maßnahmen wird die Entwicklung von Frühwarnsystemen für Drop-out-gefährdete Studierende genannt [Be18], bzw. die Etablierung von Beratungsangeboten [HMT16].

Studienabbrüche können vielfältige Ursachen haben. Diese können sich zum einen auf die betroffene Person selbst beziehen, können zum anderen aber auch strukturell bedingt sein, beispielsweise durch die Gestaltung der Studiengänge [HSW06].

4 Vorgehensweise und Rahmenbedingungen

Um uns den in Abschnitt 2 definierten Zielen anzunähern erschließen wir uns als Datenbasis das hochschulische System zur Verwaltung von Prüfungsleistungen, in dem auch Informationen zur Hochschulzugangsberechtigung der Studierenden hinterlegt sind. Generell erfordert die Auswertung personenbezogener Daten ein Datenschutzkonzept und entsprechende datenschutzrechtliche Freigaben. Des Weiteren treffen bei der Auswertung dieser personenbezogenen Daten aus dem Prüfungsverwaltungssystem die Grundrechte auf informationelle Selbstbestimmung der Studierenden und die Freiheit von Wissenschaft und Forschung aufeinander.

Zur Lösung dieses Spannungsfeldes haben wir unter Mitwirkung des Datenschutzbeauftragten die in [BTH20] detailliert beschriebene Vorgehensweise etabliert. Diese führt die Rolle "Datentreuhänder:in" ein, die von einer Person eingenommen wird, die selbst nicht in die Lehre involviert ist. Der bzw. die Treuhänder:in bereitet die personenbezogenen Daten entsprechend der datenschutzrechtlichen Vorgaben durch Pseudonymisierung auf, und stellt sie erst dann für die wissenschaftliche Auswertung zur Gewinnung allgemeiner Aussagen zur Verfügung. So wird sichergestellt, dass Lehrende im Rahmen der wissenschaftlichen Auswertung dieser Daten kein Zusatzwissen über solche Studierende erhalten können, mit denen sie in der Lehre noch konfrontiert sind.

Anhand dieses Datenbestandes aus dem Prüfungsverwaltungssystem analysieren wir Studienverläufe, um Muster zu identifizieren – sowohl für erfolgreiche Verläufe als auch für typische Problemkonstellationen. Mit Blick auf die Beratung steht dabei die Frage im Mittelpunkt, ob die Studienwege erfolgreicher Studierender Gemeinsamkeiten aufweisen, aus denen sich „Good Practices" bzw. „Dos and Don'ts" ableiten lassen, die in Beratungsprozesse als Empfehlung einfließen können. Ein besonderer Fokus liegt dabei auf den Studienwegen derjenigen Studierenden, die in der Startphase ihres Studiums selbst Schwierigkeiten hatten, ihr Studium jedoch trotzdem zu einem erfolgreichen Abschluss geführt haben.

Als Indikatoren für Schwierigkeiten bzw. die Existenz einer Problemsituation ziehen wir Prüfungen heran, die von den Studierenden entweder nicht bestanden oder zum im Studienplan vorgesehenen Zeitpunkt nicht angetreten wurden. Entsprechend lässt sich diese Betrachtung erst nach der ersten Prüfungsperiode eines Studienverlaufes durchführen. Ergänzend überprüfen wir, inwieweit sich bereits aus der prä-hochschulischen Bildungshistorie der Studierenden Indikatoren für potenziell auftretende Problemsituationen ergeben, beispielsweise anhand der Noten aus der Hochschulzugangsberechtigung oder aus dem Schultyp, an dem die Hochschulzugangsberechtigung erworben wurde.

Für diese Analysen fokussieren wir in dieser Arbeit den Bachelorstudiengang Informatik an der Hochschule München. Tabelle 1 listet die Module auf, die gemäß Studienplan in den ersten drei Semestern zu absolvieren sind. Die mit den Modulen verbundenen ECTS-Punkte sind jeweils in Klammern angegeben. Jedes Semester umfasst genau ein Wahlpflichtfach, alle anderen Fächer sind Pflichtfächer. Studienbeginn ist jeweils im Wintersemester. Die einzelnen Module werden einmal pro Jahr angeboten, die Module des ersten und dritten Semesters im Winter, die Module des zweiten Semesters im Sommer. Prüfungen zu den Pflichtmodulen werden in der Regel jedes Semester angeboten. Wiederholungsprüfungen laufen somit gegenläufig, also in einem Semester, in dem die Lehr-Lernveranstaltungen zum Modul nicht durchgeführt werden und lediglich eine Prüfung stattfindet.

Tab. 1: Module der ersten drei Semester. Credits (ECTS) jeweils in Klammern.

1. Semester	2. Semester	3. Semester
Softwareentwicklung I (8)	Softwareentwicklung II (8)	Algorithmen und Datenstrukturen (5)
Analysis (5)	Diskrete Mathematik (5)	Datenbanken I (5)
Lineare Algebra (5)	Angewandte Mathematik (5)	Software Engineering I (5)
Technische Informatik (5)	Theoretische Informatik (5)	Netzwerke I (5)
IT-Systeme Grundlagen (5)	IT-Systeme (5)	Wahlpflichtfach Mathe (5)
Allgemeinwissenschaftliches Wahlpflichtfach (2)	Allgemeinwissenschaftliches Wahlpflichtfach (2)	Wahrscheinlichkeitsrechnung und Statistik (5)

Grundsätzlich gilt die Regelung, dass alle Module des ersten (bzw. zweiten) Semesters spätestens im zweiten (bzw. dritten) Semester erstmals angetreten werden müssen[2] – ansonsten wird von Amts wegen Note 5 vergeben. Die erste Wiederholung einer mit 5 bewerteten Prüfung muss im unmittelbar folgenden Semester angetreten werden; für einen ggf. erforderlichen dritten Versuch darf dann ein Jahr gewartet werden. Ein nicht bestandener Drittversuch oder mehr als fünf notwendige Drittversuche im gesamten Studienverlauf führen zur Exmatrikulation.

Auf Grundlage einer speziellen Regelung darf das Modul Angewandte Mathematik ausschließlich dann besucht werden, wenn mindestens eines der Mathematik-Module des ersten Semesters bestanden wurde. Zu beachten ist dabei, dass das Modul Angewandte Mathematik ausschließlich im Sommersemester angeboten wird. In der Folge erhalten alle diejenigen Studierenden, die im ersten Semester weder Analysis noch Lineare Algebra bestanden haben, automatisch im zweiten Semester eine fristbedingte 5 für Angewandte Mathematik.

5 Hypothesen, Prüfung und Ergebnisse

Erste Analysen der Prüfungsdaten [BTH20] machen deutlich, dass in unserem Bachelorstudiengang Informatik etwa 20% der Immatrikulierten nicht ernsthaft studieren. Studierende, die über mehrere Semester eingeschrieben sind, ohne je eine einzige Leistung zu erbringen, bezeichnen wir als Geister (16,6%). Andere Studierende sind zwar im Studiengang eingeschrieben und erbringen dabei ggf. einige wenige Leistungen, wechseln jedoch nach wenigen Semestern in einen anderen Studiengang und lassen sich dabei die bisher erbrachten Leistungen vollständig anerkennen. Dies legt die These nahe, dass diese Studierenden in der Informatik parken, bis sie in ihrem eigentlichen Wunsch-Studiengang einen Platz bekommen (4,3% an Parkstudierenden). Als "Poor Performers" kategorisieren wir diejenigen Studierenden, die aufgrund schlechter Leistungen ihr Studium aufgeben (müssen).

Für die weiteren Untersuchungen bereinigen wir diese "unechten" Studierenden (also Geister und Parker) aus dem Datensatz und beschränken uns im Folgenden auf diejenigen Immatrikulierten, die wir als ernsthaft studierend erkannt haben. Die "echte" Abbruchquote liegt damit bei rund 48%, ist also immer noch verhältnismäßig hoch.

Bislang fließen in die Beratung von Studierenden in konkreten Problemsituationen auch Ratschläge ein, die aus einem Bauchgefühl der Lehrenden bzw. der Lernberatung heraus entstehen, oft basierend auf als typisch empfundenen Beobachtungen. Im Folgenden formulieren wir gängige Hypothesen bezüglich dieser vermuteten Einflüsse auf den Studienerfolg, überprüfen diese anhand der Datenlage aus dem Prüfungsverwaltungssystem und leiten aus den dabei gewonnenen Erkenntnissen Konsequenzen für die Beratung von Studierenden ab.

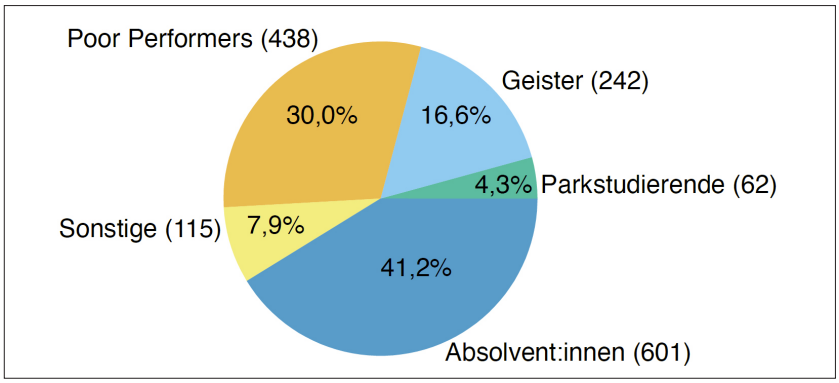

Abb. 1: Beobachtete Klassen von Studierenden aus den Immatrikulationsjahren 2006-2017 (in Summe 1458 Personen).

5.1 Hypothese 1:
Personen bestimmter Herkunftsschultypen sind besonders gefährdet

Unser Bildungssystem ermöglicht viele unterschiedliche Eingangspfade in ein Hochschulstudium. Doch nicht jeder dieser Pfade bereitet auf jeden Studiengang gleich gut vor. Eine statistische Analyse hat ergeben, dass für unseren Bachelorstudiengang Informatik die Studienerfolgsquoten der Studierenden unterschiedlicher Herkunftsschultypen zwischen rund 20% (FOS Sozialer Zweig) und ca. 60% (FOS Technischer Zweig) streuen [TBH21].

Hier empfiehlt es sich, den Studierenden der Herkunftsschultypen mit niedrigerer Erfolgsquote frühzeitig und proaktiv – also bereits zu Beginn des ersten Semesters – Unterstützungsangebote nahe zu bringen mit dem Ziel, Potenziale zu wecken *bevor* sich Überforderung und Misserfolge einstellen. Um jegliches Risiko für ein Bias in der Betreuung bzw. Bewertung durch die Lehrenden zu verhindern ist dabei für die Umsetzung zwingend erforderlich, dass die Selektion der entsprechenden Studierenden über den bzw.

2 so genannte Grundlagen- und Orientierungsprüfungen. Dazu zählen nicht die Allgemeinwissenschaftlichen Wahlpflichtfächer (AW).

die Datentreuhänder:in erfolgt und die Beratung ausschließlich der Lernberatung obliegt. Lehrende werden erst dann eingebunden, wenn der bzw. die Studierende das explizit anfrägt.

5.2 Hypothese 2:
Schulnoten sind enorm prädiktiv

Eine Analyse verschiedener Variablen zur Bildungsbiographie und dem Verlauf der Studieneingangsphase hat gezeigt, dass vor dem ersten Semester die Schulnoten etwas prädiktiv sind – aber in deutlich geringerem Maße als die Anzahl vorheriger Immatrikulationen in anderen Studiengängen sowie die Anzahl der in dieser Vorgeschichte erworbenen und dann über Anerkennung eingebrachten Credits [Bo21]. *Nach* dem ersten Semester ist die Anzahl der neu erworbenen Credits der aussagekräftigste Prädiktor für den Studienerfolg und dabei deutlich prädiktiver als die Durchschnittsnote der Hochschulzugangsberechtigung [Bo21].

Das bedeutet, dass auch Studierende mit weniger guten Schulnoten sehr wohl eine Chance auf Studienerfolg haben. Wichtig ist dabei jedoch ein möglichst guter Einstieg ins erste Semester. Hier können frühzeitige Unterstützungsangebote einen Beitrag leisten.

5.3 Hypothese 3:
Es ist grundsätzlich schädlicher, eine Prüfung zu schieben als sie nicht anzutreten

Jede:r Studierende kann für jede Prüfung des ersten Semesters entscheiden, ob sie bzw. er an dieser teilnimmt oder nicht (die Prüfung also „schiebt"). Studierende, die im ersten Semester um eine Beratung bitten, stehen oft vor der Frage, wie sie sich hinsichtlich der anstehenden Prüfungen konkret verhalten sollen. Das Aufschieben einer Prüfung ist auf den ersten Blick unschädlich. Diese Entscheidung hat allerdings ebenso wie das Nicht-Bestehen einer Prüfung eine Auswirkung auf die Arbeitslast in mindestens einem der Folgesemester. Dozierende neigen daher dazu, vom Schieben von Prüfungen abzuraten.

Abbildung 2 zeigt den relativen Anteil erfolgreicher Studierender (d. h. Absolvent:innen), abhängig von der Anzahl geschobener und nicht-bestandener Grundlagen- und Orientierungsprüfungen im ersten Semester. Von den Studierenden, die alle diese Prüfungsleistungen im ersten Semester erfolgreich erbracht haben (0 geschoben, 0 nicht bestanden) erreichen über 85% ihren Abschluss. Bei einer fehlenden Prüfungsleistung sind es dagegen noch um die 70%.

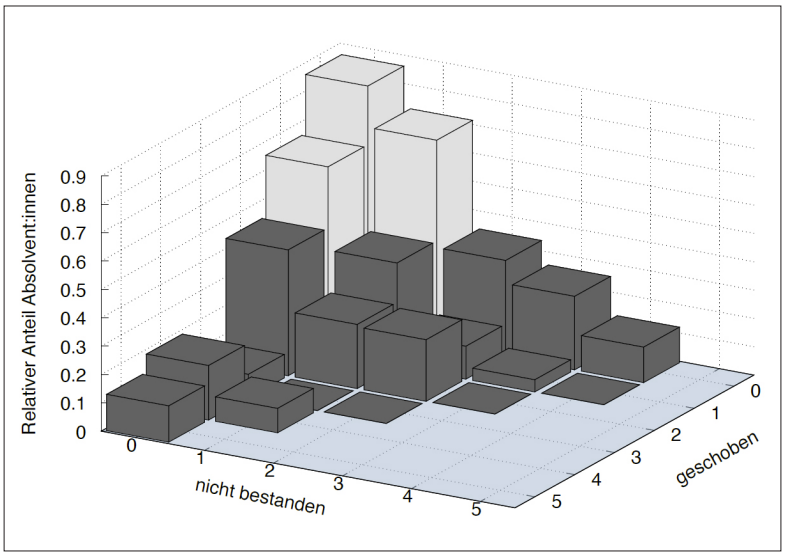

Abb. 2: Relativer Anteil erfolgreicher Studierender in Abhängigkeit der Anzahl an geschobenen und nicht-bestandenen Prüfungen im ersten Semester.

Insgesamt wird deutlich, dass der Grund, warum eine Prüfungsleistung nicht erbracht wurde (Nicht-Bestehen vs. Schieben), sich kaum auf den Studienerfolg auswirkt. Dies ist daran erkennbar, dass in beide Achsenrichtungen der Abfall der Säulen ähnlich verläuft. Dabei ist ein deutlicher Abfall der Erfolgsrate von einer zu zwei oder mehr nicht erfolgreich erbrachten Prüfungsleistungen erkennbar. Entsprechend ist es nicht ratsam, mehrere Prüfungen zu schieben. Der Datenpunkt bei fünf geschobenen Prüfungen deutet auf Studierende mit persönlichen Problemen oder ernsthaften Erkrankungen hin, die ihre Defizite später aufholen und ihr Studium zu einem erfolgreichen Abschluss bringen konnten.

Als Erkenntnis lässt sich ableiten, dass wir den Studierenden, die sich im ersten Semester überfordert fühlen, durchaus raten dürfen, einzelne Prüfungen zu schieben, um sich bei der Prüfungsvorbereitung nicht zu verzetteln. Offen bleibt dabei zunächst die Frage, auf welche Module dann vordringlich die Konzentration zu lenken ist.

5.4 Hypothese 4:

Softwareentwicklung I und Analysis sind die entscheidenden Hürden

Zur Überprüfung dieser Hypothese wurde ein Entscheidungsbaum-Modell erstellt [HKP12, TA19]. Basis sind Ergebnisse der Erstsemester-Prüfungen, repräsentiert durch kategoriale Variablen mit jeweils drei Merkmalsausprägungen *bestanden*, *nicht bestanden* sowie *geschoben*. Die Zielvariable ist die dichotome Variable Abschluss ja/nein. Aus der Analyse herausgenommen wurden die als Geister und Parker identifizierten Personen sowie diejenigen, die alle Prüfungsleistungen aus Vorstudien anerkannt bekommen haben, was die Differenz der Gesamtzahl zu Abbildung 1 erklärt. Das Ergebnis ist in Abbildung 3 dargestellt. Die Pfeile zu den beiden möglichen Werten *Kein Abschluss* bzw. *Abschluss* der Zielvariable sind beschriftet mit der Anzahl der korrekt (in Klammern: inkorrekt) klassifizierten Studierenden.

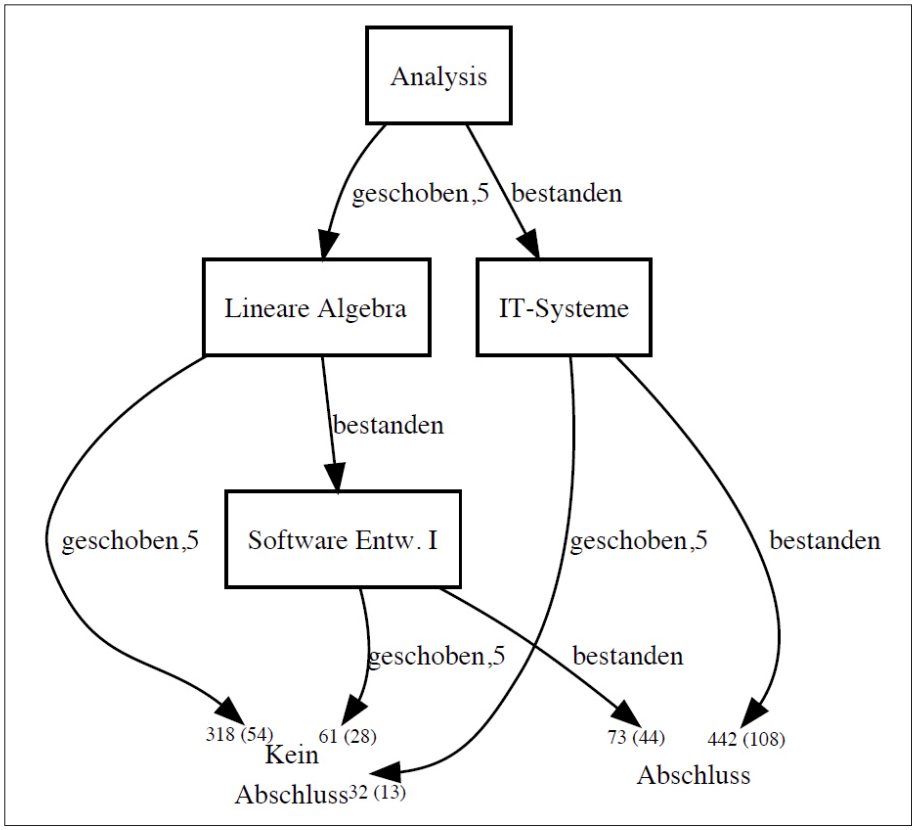

Abb. 3: Entscheidungsbaum für Prognose des Studienerfolgs auf Basis des Prüfungsverhaltens im ersten Semester.

Auch in diesem Modell zeigt sich kein Unterschied, ob eine Prüfung nicht angetreten (geschoben) oder nicht bestanden wurde. Die beiden Mathematik-Module erweisen sich als sehr prädiktiv, d. h. die Studierenden scheitern tendenziell eher an der Mathematik als am Modul Softwareentwicklung I. Das Modul IT-Systeme Grundlagen ist ebenfalls prädiktiv. Inhaltlich ist dieses Modul auch als Programmier-nah anzusehen, da dort intensiv auf Ebene von Maschinensprache gearbeitet wird.

Daraus lässt sich als Empfehlung an die Studierenden ableiten, sich bei Schwierigkeiten im ersten Semester auf eines der beiden Mathematik-Module sowie auf Softwareentwicklung I und IT-Systeme zu konzentrieren.

5.5 Hypothese 5:

Zu viele offene Prüfungsleistungen aus den ersten beiden Semestern sind schädlich

Ein immer wieder beobachtbares Verhaltensmuster ist, dass sich Studierende mit Altlasten aus dem Bereich der Grundlagen- und Orientierungsprüfungen im dritten Semester zu viel vornehmen. Dabei besteht die Gefahr, dass sich durch Nicht-Bestehen von Regelprüfungen des dritten Semesters unnötiger Weise zusätzlicher Druck aufbaut, in Form von noch mehr einzuhaltenden Fristen für die dann sofort im nächsten Semester anzutretenden Wiederholungsprüfungen dieser nicht bestandenen Drittsemester-Module.

Um herauszufinden, wie relevant diese Beobachtungen sind, stellen wir in Abbildung 4 die Anzahl der erfolgreichen und erfolglosen Studierenden einander gegenüber, in Bezug auf die Anzahl der offenen Grundlagen- und Orientierungsprüfungen („Altlasten") und der bereits angetretenen Regelprüfungen des dritten Semesters.

Hier lässt sich ein Phänomen beobachten, welches als „Murky Middle“ beschrieben wurde [EA], also eine Dreiteilung der Kohorten in

1. überwiegend erfolgreiche Studierende, ohne oder mit höchstens einer offenen Prüfung aus den ersten beiden Semestern (grüner Bereich in Abb. 4 c); grün gefärbt sind dabei Felder mit mindestens 14 erfolgreichen Studierenden). Diese Studierenden benötigen keine besondere Betreuung.

2. diejenigen, die den Studiengang absehbar bald verlassen werden; das sind im Wesentlichen diejenigen Studierenden, die im 3. Semester kaum reguläre Prüfungen antreten und mehrere Altlasten haben (roter Bereich in Abb. 4 c)).

3. einen mittleren Bereich, in dem die Anzahl der erfolgreichen und erfolglosen Studierenden in etwa ausgeglichen ist. Dieser Bereich ist in Abbildung 4 c) grau gefärbt.

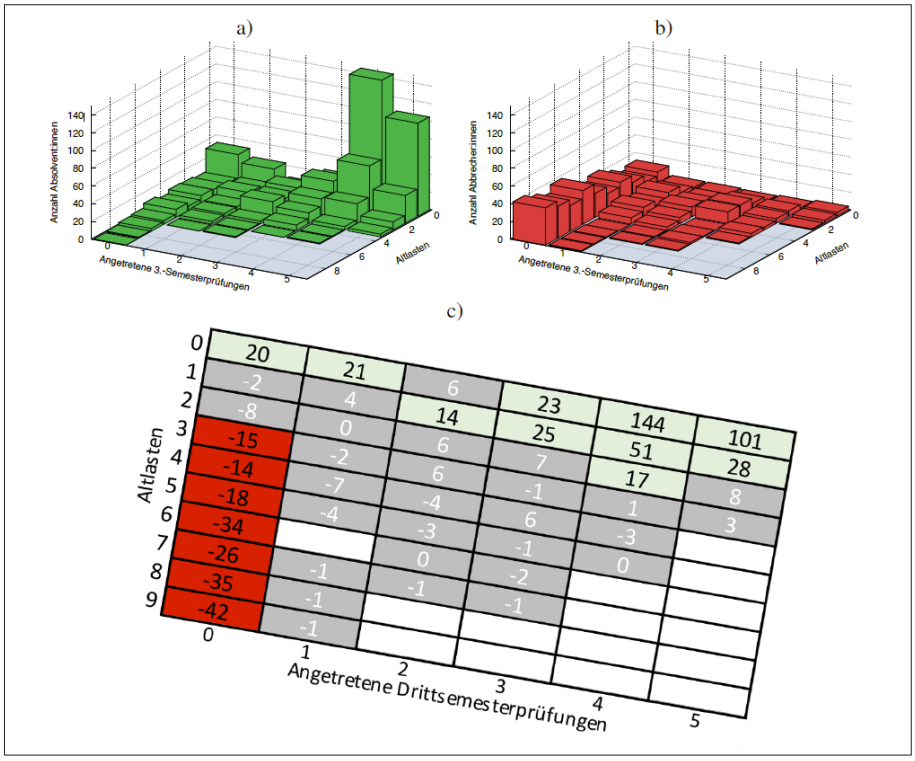

Abb. 4: Anzahl erfolgreicher (a) und erfolgloser Studierender (b) in Abhängigkeit davon, wie viele Prüfungen aus dem ersten und zweiten Semester sie im dritten Semester noch ablegen müssen und wie viele Prüfungen aus dem dritten Semester sie antreten. Teilgrafik (c) zeigt die Differenz als vereinfachte Heatmap. Farblich grau gekennzeichnet ist die „Murky Middle“

Gerade für Personen im mittleren Bereich ist eine gezielte Beratung erforderlich. Studierende diesem Bereich benötigen gezielte Unterstützung bei der Planung der wirklich abzulegenden Prüfungen, damit sie sich nicht verzetteln.

6 Zusammenfassung und Ausblick

Die Datenanalyse hat hinsichtlich der Korrektheit gängiger Hypothesen Klarheit geschaffen – teilweise wurden die Hypothesen widerlegt, teilweise bestätigt. Insgesamt lässt sich festhalten, dass Schulnoten weniger wichtig sind als gedacht, der Herkunftsschultyp jedoch eine große Rolle spielt für die Chance auf einen Studienerfolg. Des Weiteren ist es für den späteren Studienerfolg irrelevant, ob nicht-erbrachte Leistungen im ersten Semester durch Schieben oder doch Nicht-Bestehen bedingt sind. Ferner wurde offensichtlich, dass die entscheidende Hürde in unserem Bachelorstudiengang Informatik die Mathematik darstellt.

Eine positive Erkenntnis ist, dass ein nicht ganz so guter Start ins Studium nicht zwingend bedeutet, dass das Studium erfolglos aufgegeben werden muss. Sofern ernsthaftes Interesse vorhanden ist kann nach dem ersten Semester kein eindeutiger Indikator für das Nicht-Bestehen identifiziert werden. Eine weitere Analyse von Studienverlaufsdaten derjenigen Studierenden, die nach einem suboptimalen Start einen Abschluss erhalten haben, förderte entsprechend auch keine klaren Muster zutage, auf deren Basis sich Empfehlungen aussprechen lassen. Um dennoch von dieser Personengruppe zu lernen und daraus Empfehlungen für eine Ausdehnung bzw. zukünftig neu zu etablierende Beratung abzuleiten, führen wir derzeit qualitative Interviews durch, von denen wir uns weitere Einsichten erhoffen.

Mit Blick auf mögliche strukturelle Maßnahmen legen die Auswertungen der Prüfungsdaten nahe, das Modul Angewandte Mathematik auch im Wintersemester anzubieten. Das würde den Studierenden zum einen die Möglichkeit zu bieten, eine zwangsweise Fristüberschreitung zu vermeiden – und zum andern nicht unnötig dazu verleiten, weitere Probleme durch das zu frühe Antreten von Prüfungen des dritten Semesters aufzubauen. Auch die Einführung strengerer Regeln zum Vorrücken in höhere Semester würde die Problemgenerierung durch das zu frühe Antreten von Drittsemester-Prüfungen wenigstens teilweise unterbinden. (Die Erfahrung zeigt dass insbesondere die zentrale Studierendenvertretung massiv gegen die Einführung derartiger Regeln vorgeht – während die Studierendenvertretung der eigenen Fakultät durchaus deren Sinnhaftigkeit erkennt und der Einführung gegenüber offen ist.)

Literaturverzeichnis

[Be18] Berens, Johannes; Schneider, Kerstin; Görtz, Simon; Oster, Simon; Burghoff, Julian:, Early Detection of Students at Risk – Predicting Student Dropouts Using Administrative Student Data and Machine Learning Methods. CESifo Working Paper Series 7259, CESifo Group Munich, 2018.

[Bo21] Böttcher, Axel; Thurner, Veronika; Häfner, Tanja; Hertle, Jochen: A Data Science-based Approach for Identifying Counseling Needs in first-year Students. In: 2021 IEEE Global Engineering Education Conference (EDUCON). S. 425–434, 2021.

[BTH20] Böttcher, Axel; Thurner, Veronika; Häfner, Tanja: Applying Data Analysis to Identify Early Indicators for Potential Risk of Dropout in CS Students. In: 2020 IEEE Global Engineering Education Conference (EDUCON). S. 827–836, 2020.

[DI18] DIPF Deutsches Institut für Internationale Pädagogische Forschung: Bildung in Deutschland 2018 – Ein indikatorengestützter Bericht mit einer Analyse zu Wirkungen und Erträgen von Bildung. www.bildungsbericht.de/de/bildungsberichte-seit-2006/bildungsbericht-2018, 2018.

[EA] EAB Global, Inc.: , The Murky Middle Project. https://eab.com/technology/whitepaper/student-success/the-murky-middle-project/. Accessed: 2020-11-27.

[FY15] Fei, Mi; Yeung, Dit-Yan: Temporal Models for Predicting Student Dropout in Massive Open Online Courses. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW). S. 256–263, 2015.

[HKP12] Han, Jiawei; Kamber, Micheline; Pei, Jian: Data mining concepts and techniques, third edition. Morgan Kaufmann Publishers, 2012.

[HMT16] Hinkelmann, Mathias; Maucher, Johannes; Tobias, Seidl: Softwaregestützte Studienverlaufsanalyse zur frühzeitigen gezielten Studienberatung. die Hochschullehre, 2, 2016.

[HSW06] Holdt, Ulrike; Schneider, Heide; Wagner, Bernardo: Analyse von Studienverläufen und Studienabbrüchen in den Bachelorstudiengängen Informatik an der Leibniz Universität Hannover. In: HDI 2006 - Hochschuldidaktik der Informatik. 2. GI- Fachtagung. S. 115–126, 12 2006.

[KVW20] Kemper, L.; Vorhoff, G.; Wigger, B. U.: Predicting student dropout: A machine learning approach. European Journal of Higher Education, 10(1):28–47, 2020.

[TA19] Therneau, Terry; Atkinson, Beth: . rpart: Recursive Partitioning and Regression Trees, 2019. R package version 4.1-15.

[TBH21] Thurner, Veronika; Böttcher, Axel; Häfner, Tanja: A Detailed Analysis of Gender Differences in the Course of CS-Studies. In: 2021 IEEE Global Engineering Education Conference (EDUCON). S. 487–496, 2021.

Mehr Diversität durch Mono-Edukation

Juliane Siegeris[1]

Abstract:
Den Frauenstudiengang Informatik und Wirtschaft (kurz: FIW) gibt es seit 2009 an der HTW Berlin. Dieses Studienangebot ist eins von sechs mono-edukativen Studienangeboten in Deutschland. Ziel dieser Studiengänge ist es, die Zahl der weiblichen Absolventinnen in ausgewählten MINT–Fachrichtungen zu erhöhen. In dem vorliegenden Beitrag wird die Motivation für ein solches Angebot erläutert und Erfahrungen aus den zurückliegenden Jahren geteilt werden. Dabei wird insbesondere auf das Konzept und die Ansprache zur Gewinnung neuer Studentinnen sowie auf die Zusammensetzung der Studierendenschaft eingegangen. Es wird gezeigt, dass mit dem mono-edukativen Angebot eine neue Klientel von Studentinnen für die Informatik erreicht wird.

Keywords:
Diversity, Frauen und IT, Frauenstudiengänge, MINT

1 Mono-Edukative Studiengänge

Nach wie vor sind nur 15% der Beschäftigten in Informations- und Kommunikationstechnikberufen Frauen, vgl. [St19]. Ein ähnliches Bild ergibt sich auch bei der Anzahl der IT-Studentinnen. Obwohl ein Aufwärtstrend erkennbar ist, beträgt der Frauenanteil in den 1700 IT-nahen Studiengängen nur 23%, vgl. [BI18]. Demgegenüber steht ein erhöhter Bedarf an Fachkräften für die IT. Laut einer repräsentativen Befragung der Bitkom in Unternehmen aller Branchen konstatieren sieben von zehn aktuell einen Mangel an IT-Spezialisten, vgl. [BI20]. Um dem Fachkräftemangel zu begegnen, fordert der Bitkom ergänzend zu einer besseren Aus- und Weiterbildung, auch die Stärkung von Frauen in der IT.

In Deutschland gibt es verschiedenste Anstrengungen mit dem Ziel, den Anteil von Frauen in MINT-Studiengängen und infolgedessen auch in den entsprechenden Berufsfeldern zu steigern. Prominente Beispiele sind die bundesweite Netzwerk-Initiative „Komm, mach MINT."[2] oder das vom BMBF geförderte Vorhaben „FRUIT: Frauen in IT - zur Erhöhung des Frauenanteils im Studienbereich Informationstechnologie durch Maßnahmen flexibler, praxisorientierter und interdisziplinärer Studiengangsgestaltung"[3].

Mono-edukative Studiengänge werden als ein mögliches Instrument gesehen, um mehr Frauen in die IKT-Branche zu bekommen. In Deutschlands Hochschullandschaft gibt es derzeit drei mono-edukative Angebote an Fachhochschulen im MINT-Bereich, vgl. [De21].

In 2020 hat das gemeinnützige Centrum für Hochschulforschung (CHE) aktuelle Handlungsempfehlungen für mehr Frauen im Informatikstudium entwickelt, vgl. [Fr19]. Allerdings lautet die Handlungsempfehlung 4, S.10: Keine mono-edukativen Lehrveranstaltungen. Laut den Autoren, werden diese eher skeptisch beurteilt und sprechen nur wenige Frauen an. Es bestehe die Gefahr, dass Frauen sich dabei als Minderheit wahrnehmen, die speziell gefördert werden müsste.

Im vorliegenden Paper wird argumentiert, dass ganze Studiengänge als Angebot nur für Frauen durchaus eine Berechtigung haben. Für die Argumentation werden die Antworten aus einer seit 2015 regelmäßig durchgeführten Umfrage unter den Studienanfängerinnen verwendet. Erste Analyseergebnisse zeigen, dass mit dem mono-edukativen Angebot an der HTW Berlin in Kombination mit einer passenden Ansprache Studentinnen erreicht werden, die sonst nicht Informatik studiert hätten. Es ist jedoch wichtig, diese mono-edukativen Angebote so zu gestalten, dass die vermittelten Kompetenzen mit denen der gemischten IT-Studiengänge vergleichbar sind.

1 HTW Berlin, Wilhelminenhofstr. 75a, 12459 Berlin, Germany, siegeris@htw-berlin.de

2 www.komm-mach-mint.de

3 www.che.de/projekt/bmbf-forschungsprojekt-fruit-frauen-in-it

2 Vorstellung des Frauenstudiengangs Informatik und Wirtschaft

Der Studiengang Informatik und Wirtschaft (für Frauen) wurde 2009 an der HTW Berlin gegründet. Ziel war es, ein alternatives Informatikangebot zu schaffen, welches Frauen inhaltlich und programmatisch anspricht und ihnen den Eintritt in eine Informatiklaufbahn ermöglicht.

Der Studiengang Informatik und Wirtschaft ist ein Bachelorstudiengang mit einer Laufzeit von 6 Semestern und 180 ECTS.

Immatrikuliert wird nur einmal im Jahr - immer zum Wintersemester. Die Kapazität eines Zuges beträgt dabei 40 Studentinnen. Obwohl der Name des Studiengangs eine paritätische Aufteilung suggeriert, liegt der Schwerpunkt der Ausbildung mit etwa 80% des Fächeranteils auf der Informatik. Der Wirtschaftsanteil macht weniger als 20% aus und ist dafür gedacht, den Studentinnen die Sprache eines großen Anwendungsgebiets zu erschließen bzw. sie für den Fall einer Unternehmensgründung vorzubereiten.

Neben der Fachlichkeit sollen die Studentinnen ein Selbstverständnis als Informatikerin entwickeln. Das heißt, dass sie die eigenen Stärken kennen und selbsbewußt vertreten können.

Der Studiengang hat mittlerweile drei Akkreditierungsrunden durchlaufen. Das Curriculum wurde dabei immer stärker in Richtung Informatik und Praxis geschärft. Zu Beginn hatte der Studiengang einen wesentlich höheren Wirtschaftsanteil und zusätzlich Soft-Skill-Fächer wie Vertragsverhandlungen, Präsentationstechnik, Konflikt- und Karrieremanagement. Die Annahme war, dass dies die Studentinnen gut auf die männer-dominierte Arbeitskultur vorbereiten würde. Tatsächlich hat diese Ausrichtung bei Kritikern des mono-edukativen Studienangebots zur Einschätzung als ein „Informatik-light-Studiengang" geführt.

Um diesem Image entgegenzuwirken, setzt der Studiengang heute auf einen deutlich informatik-orientierteren Fächerkanon[4] und auf besonders viel Praxiserfahrung. Über letzteres sollen die sozialen Kompetenzen für die zukünftige Arbeit in Teams und das Selbstverständnis im Umgang mit aktuellen Methoden und Werkzeugen erreicht werden. Gleichzeitig deuten aktuelle Forschungsergebnisse darauf hin, dass ein hoher Praxisbezug beziehungsweise eine enge Verbindung zwischen Beruf und Studium die Attraktivität von IT-Studienfächern besonders für Frauen steigern können, vgl. [Fr18] S.28.

3 Curriculum und Praxisbezug

Die Abbildung 1 zeigt den aktuellen Ablauf des Studiums mit seiner ausgeprägten Praxisorientierung.

Abbildung 1: Ablauf des FIW-Studiums

Dabei dienen Semester eins und zwei (u. a. mit Lehrveranstaltungen, wie Programmierung I+II, Softwaretechnik, grundlegende Konzepte der Informatik, Rechnerarchitektur und Betriebssysteme, Rechnernetze) der Einführung in das Fachgebiet. Schon im dritten Semester absolvieren die Studentinnen ihr erstes Praxisprojekt (flankiert u. a. von Webtechnologien, Datenschutz und Datensicherheit, Datenbanken). Im vierten Semester gehen die Studentinnen für 18 Wochen in ein Vollzeitpraktikum. Im fünften Semester folgt ein weiteres Praxisprojekt (begleitende Lehrveranstaltungen sind u. a. Verteilte Systeme, Modellierung von Informationssystemen und zwei weitere Informatik-Wahlpflichtangebote). Das sechste Semester dient der Anfertigung der Bachelorarbeit.

Insgesamt werden den Studentinnen im Verlauf des Studiums bis zu elfmal Praxiseinblicke und -erfahrungen geboten. Das prominenteste Beispiel dafür sind die schon erwähnten Praxisprojekte.

4 Obwohl der Informatikanteil im Studiengang überwiegt, wurde der Name des Studiengangs bisher nicht geändert, da Frauen fachübergreifende Studiengänge (sogenannte Bindestrich-Studiengänge) vergleichsweise häufiger belegen, als reine IT-Studiengänge, vgl. [Fr18] S.23

Praxisprojekte:
Dabei arbeiten immer fünf bis sieben Studentinnen über ein ganzes Semester an der Aufgabenstellung eines externen Auftraggebers. Beschreibungen aller Projekte aus den letzten zehn Jahren, sowie die Anleitungen für die externen Beteiligten finden sich unter https://fiw.htw-berlin.de/studium/projekte/. Seit dem WS 2015/16 wird dabei auf eine konsequent agile Umsetzung der Projekte mit Scrum gesetzt, vgl. [SSB18].

In den ersten drei Jahren des Studiengangs waren die Teams entweder ausschließlich aus dem 3. oder dem 5. Semester besetzt. Das war eine zu große Herausforderung für die jüngeren Semester, da diese noch nicht über genügend Informatik-Know-how und Selbstbewusstsein verfügten. Seit 2013 werden die Teams paritätisch aus den beiden Semestern zusammengestellt. Das ermöglicht den Dritt-Semestlerinnen einen schnelleren Einstieg und fördert den Austausch zwischen den Zügen.

Arbeitswelt IT:
Im Rahmen dieser Lehrveranstaltung im 3. Semester werden mindestens sechs Exkursionen in verschiedene Berliner IT-Unternehmen durchgeführt. Im Laufe jeweils eines halben Tages erhalten die Studentinnen vor Ort (bzw. derzeit virtuell) Einblick in aktuelle Projekte und typische Aufgabenstellungen. Sie bekommen nebenher einen Einblick in die Unternehmenskultur, lernen „Role models" kennen und bauen Kontakte auf. Sowohl für die Unternehmen, als auch für die Studentinnen ist der Zeitpunkt im 3. Semester dafür sehr günstig, da im 4. Semester die Praktika anstehen.

Praktikum und Bachelorarbeit:
Eine weitere Möglichkeit ein Unternehmen und deren Projekte und Aufgabenstellungen kennenzulernen, bietet neben dem Praktikum auch die Bachelorarbeit. Hier werden die Studentinnen explizit ermutigt, sich erneut ein Unternehmen zu suchen und sich damit, einen weiteren Praxiseinblick vor dem eigentlichen Berufseinstieg zu ermöglichen.

Mentoring:
Begleitend zum Studium wird von FIW auch das fachbereichsweite Mentoring-Programm ProfIT[5] koordiniert. Dieses Programm bietet jedes Jahr zwanzig Studentinnen (aller Informatikstudiengänge der HTW) die Möglichkeit sich zehn Monate lang von einer Informatikerin aus der Praxis individuell coachen zu lassen und sich zu vernetzen.

4 Ansprache potentieller Studentinnen

In den ersten Jahren wurde sehr viel Aufwand in die Akquise potentieller Studentinnen gesteckt. Dafür wurden Flyer und Vorträge erstellt, um damit persönlich (durch Mitarbeiter*innen und Studentinnen des Studiengangs) in Schulen zu gehen und zu werben.

Mittlerweile beschränkt sich die Darstellung des Studienangebots auf die Kanäle der HTW: die Webseite[6] und die Beteiligung an internen Veranstaltungsformaten, wie „(Digitale) Einblicke", „Mädchen machen Technik"[7] und den „Girls' Day".

Relevant für die Ansprache von potentiellen Studentinnen sind die folgenden drei Slogans, die in allen Materialien des Studiengangs mittlerweile prominent vertreten sind:

1. Wir fangen bei Null an.
Dieser Slogan adressiert die Erwartungen, die bei Beginn des Studiums an die IT-Vorkenntnisse gestellt werden. Mit dieser Ansprache wird explizit auf die Befürchtung vieler Studentinnen eingegangen, nicht die gleichen Voraussetzungen mitzubringen, wie sie sie z. B. ihren früheren Mitschülern zugestehen. Damit sollen Studentinnen gewonnen werden, die sich IT zutrauen, obwohl sie bisher wenig Berührungspunkte damit hatten. Freude an Mathematik und logischem Denken wird jedoch vorausgesetzt.

2. Familienfreundliche Präsenz.
Mit Start des Studiengangs wurde durchgesetzt, dass Studentinnen die Kinder oder pflegebedürftige Angehörige haben, bei FIW eine Garantie für Präsenzzeiten zwischen 9:00 und 16:00 Uhr haben. Zumindest an der HTW ist FIW der einzige Studiengang, für den es eine solche Regelung gibt. Anfragen anderer Studierender zeigen uns jedoch, dass der Bedarf für solche Freiräume nicht auf Frauenstudiengänge und Frauen begrenzt ist. Damit ließe sich auch die Attraktivität jedes anderen Studienangebots erhöhen.

5 www.f4.htw-berlin.de/studieren/profit-mentoring

6 https://fiw.htw-berlin.de

7 https://events.htw-berlin.de/hochschule/maedchen-machen-technik

3. Fragen erwünscht.
In [ACM17] untersuchten die Autor*innen das Verhalten von Studierenden in gemischt-geschlechtlichen Computer Science-Kursen und fanden heraus, dass weibliche Studierende eher Hemmungen haben, Fragen im Unterrichtsraum zu stellen. Mit dem Slogan „Fragen erwünscht" soll deutlich gemacht werden, dass bei FIW eine offene Fragenkultur gelebt und Studentinnen explizit ermutigt werden, Nachfragen zu stellen.

5 Studierendenschaft

In diesem Abschnitt wird gezeigt, wie sich die Studierendenschaft des Studiengangs zusammensetzt und welche Beweggründe zur Wahl von FIW geführt haben.

Tabelle 1 zeigt die Bewerberinnen und Immatrikulationszahlen der letzten zehn Jahre.

Tabelle 1: Anzahl der Bewerberinnen und Immatrikulationen seit 2011

Jahr	2020	2019	2018	2017	2016	2015	2014	2013	2012	2011
Bewerb.	162	294	215	234	228	150	74	96	115	80
Immatrik.	42	41	42	51	43	42	58	41	42	54

Es wird deutlich, dass von Beginn an genügend Bewerberinnen für dieses spezielle Angebot vorhanden sind.[8]

Aus der Historie des Studiengangs und vielen Gesprächen mit Studentinnen wurde die These abgeleitet, dass mit dem Studiengang eine Klientel erreicht wird, die sich ohne dieses besondere Studienangebot und die entsprechende Ansprache nicht für ein Studium der Informatik entschieden hätte.

These.
Mit dem mono-edukativen Studiengang FIW werden Studentinnen erreicht, die sonst nicht Informatik studiert hätten.

Ein erstes Indiz dafür sind die Anfängerinnenzahlen anderer Informatikstudiengänge an der HTW. Der befürchtete Effekt, dass die Studentinnenzahlen abnehmen, ist nicht eingetreten. Dass lässt sich aus der Abbildung 2 ablesen. Hier sieht man die Zahl der weiblichen Studienanfängerinnen von drei weiteren Informatikstudiengängen der HTW - vor und nach Aufsetzen des Studiengangs FIW in 2009. Die Tendenz in den anderen Studiengängen ist unabhängig von FIW steigend. In Klammern ist jeweils die angestrebte Zuggröße angegeben.

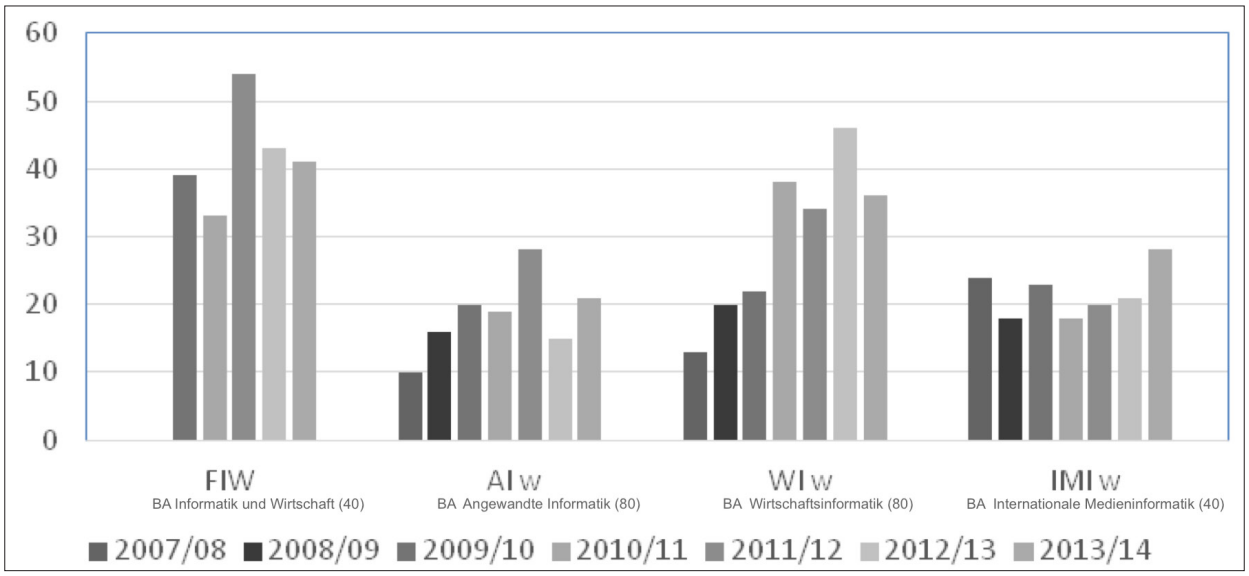

Abbildung 2: Anzahl der Studienanfängerinnen nach Studiengang, Quelle: QM-HTW

8 Der Sprung der Bewerberinnenzahlen im Jahr 2015 ist auf einen Wechsel auf ein Online-Anmeldesystems zurückzuführen. Seit diesem Zeitpunkt ist es leichter möglich, sich gleichzeitig für mehrere Studienplätze zu bewerben.

Als Nächstes werden die Gründe für die Wahl des Studiengangs untersucht. Um dieWahl der Studentinnen für FIW (bzw. speziell für den Aspekt der Mono-Edukation) zu untersuchen, werden alle Erstsemestlerinnen seit 2015 mittels eines Fragebogens zu ihrer Motivation und ihrem Hintergrund befragt. Dieser Fragebogen wird immer in der Einführungswoche auf Papier heraus gegeben. Die Rücklaufquote liegt im Schnitt bei etwas mehr als 90%.

Um die Relevanz der Mono-Edukation bei der Wahl des Studiengangs zu erfragen, werden u.a. folgende Aspekte abgefragt: die Motivation für die Bewerbung, Menge und Priorität der gewählten Studiengänge, Menge der Zusagen und die Gründe für die endgültige Entscheidung. Die folgenden Auswertungen zeigen die aggregierten Daten in Prozent für die Jahre 2015-2020.

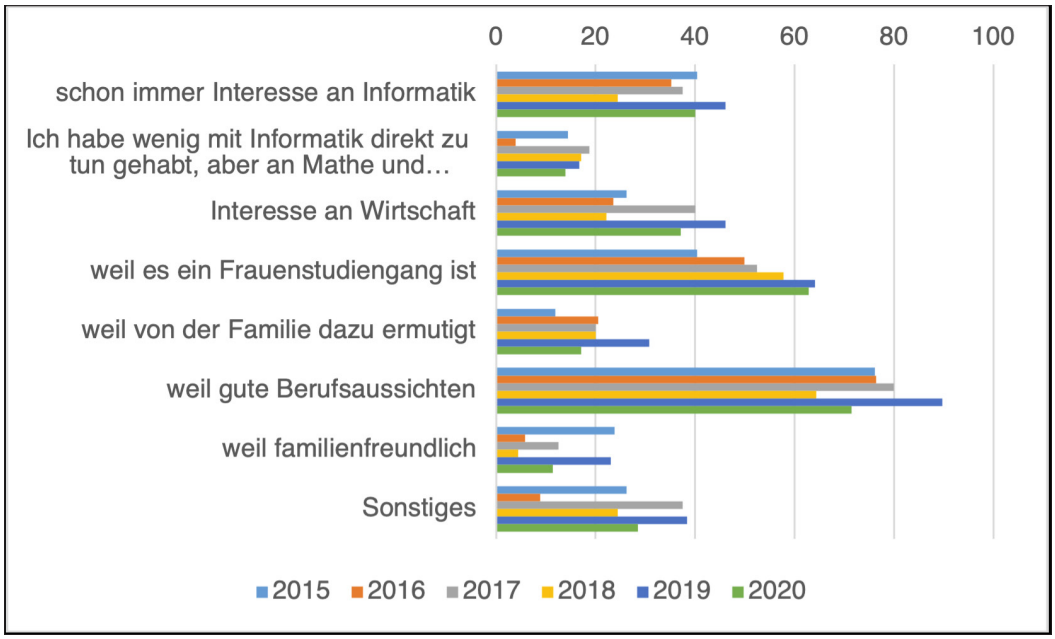

Abbildung 3: Warum haben Sie sich für FIW beworben?

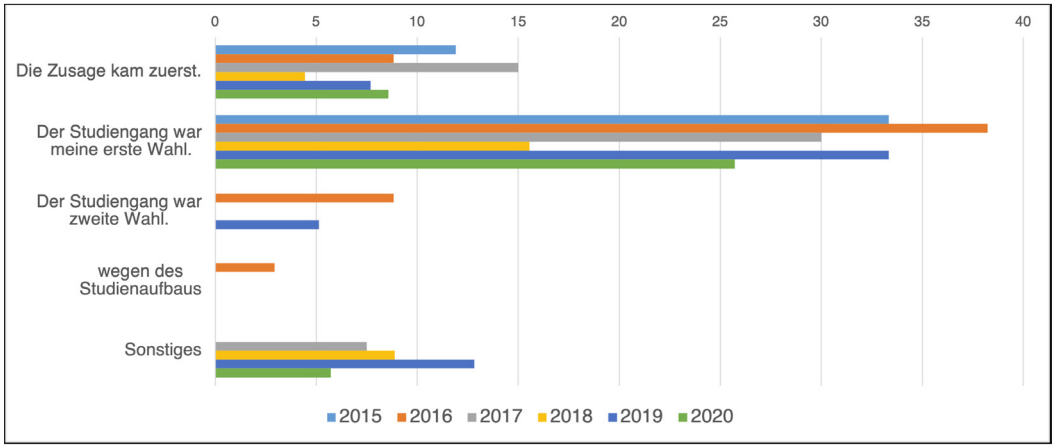

Abbildung 4: Warum haben Sie sich für FIW entschieden?

Die Resultate für die Frage nach den Gründen und dem Ausschlag für die Entscheidung finden sich in Abbildung 3 und 4. Bei beiden Fragen waren Mehrfachnennungen möglich.

Bei der Frage nach der Motivation (vgl. Abbildung 3) liegen die guten Berufsaussichten (im Mittel 76,4%) vor dem Aspekt Frauenstudiengang. Der Aspekt der Mono-Edukation wurde jedoch von mehr als der Hälfte (54,6%) der Studienanfängerinnen als Beweggrund genannt. Spätestens bei der Entscheidung (vgl. Abbildung 4) wird ersichtlich, dass ein knappes Drittel der Studentinnen (29,4% der Befragten) den Studiengang genau wegen des Aspekts Frauenstudiengang gewählt haben.

Ein ähnliches Ergebnis erbrachte auch die Frage: Haben Sie sich für weitere Studiengänge beworben?, vgl. Abbildung 5. Auch hier waren Mehrfachnennungen möglich. Daher ist nur die Nein-Aussage prozentual sinnvoll auswertbar. Mit „Nein" antworten im Schnitt etwas mehr als 27% der Befragten. Diese Gruppe hätte also ohne dieses Angebot gar nicht (nochmal) studiert. Aber auch unter den anderen Studentinnen, die sich nicht ausschließlich auf FIW fokussiert haben, gibt es einen großen Anteil an Studienwünschen, die nichts mit Informatik zu tun haben. Dazu zählen BWL, VWL als auch sonstige. Unter sonstige wurden (hier beispielhaft die Daten aus dem WS 2017/18) Pharmazie, Pädagogik, Jura, Architektur, Pädagogik, Archäologie, Wirtschaftsingenieurwesen, regenerative Energien und Medienwissenschaften genannt. Auch aus dieser Gruppe hätte ein Teil ohne den Studiengang FIW nicht zu einem Informatikstudium gefunden.

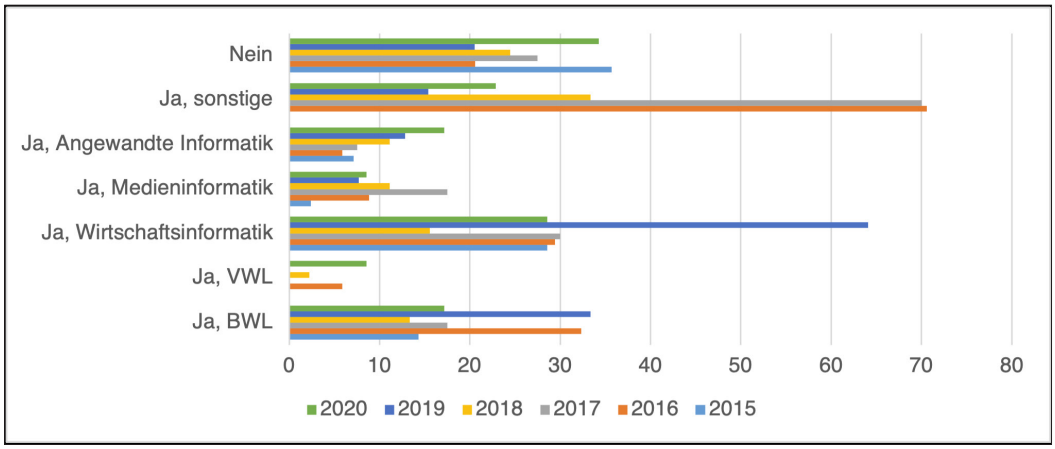

Abbildung 5: Haben Sie sich für weitere Studiengänge beworben?

Eine Begründung für die Wahl des mono-edukativen Angebots könnte aus dem Bedürfnis heraus entstehen, sich mit weniger IT-Vorkenntnissen in einem reinen Frauenstudiengang besser aufgehoben zu fühlen. Diese Frauen werden durch die Ansprache (vgl. Abschnitt 4 1. „Wir fangen bei Null an." und 3. „Fragen erwünscht.") gezielt adressiert. Um das zu erforschen, wird gefragt: „Haben Sie bereits mit Informatik zu tun gehabt?". Die Ergebnisse zu dieser Frage bietet Abbildung 6: Man findet bestätigt, dass im Durchschnitt 47,5% der Befragten hier direkt mit Nein antworten.

Ein weiteres Indiz für die These, dass durch FIW eine neue Klientel gewonnen wird, kann man in der Akquise von Frauen sehen, die bereits einen Abschluss oder Hochschulabschluss haben. Diese Frauen hätten ohne den Studiengangwahrscheinlich gar nicht mehr studiert. Um das zu untersuchen, wurde gefragt: „Was hätten Sie gemacht, wenn Sie keinen Studienplatz bekommen hätten?". Hier sagen im Durchschnitt immerhin 21,5% der Befragten: „meiner bisherigen Berufstätigkeit nachgegangen".

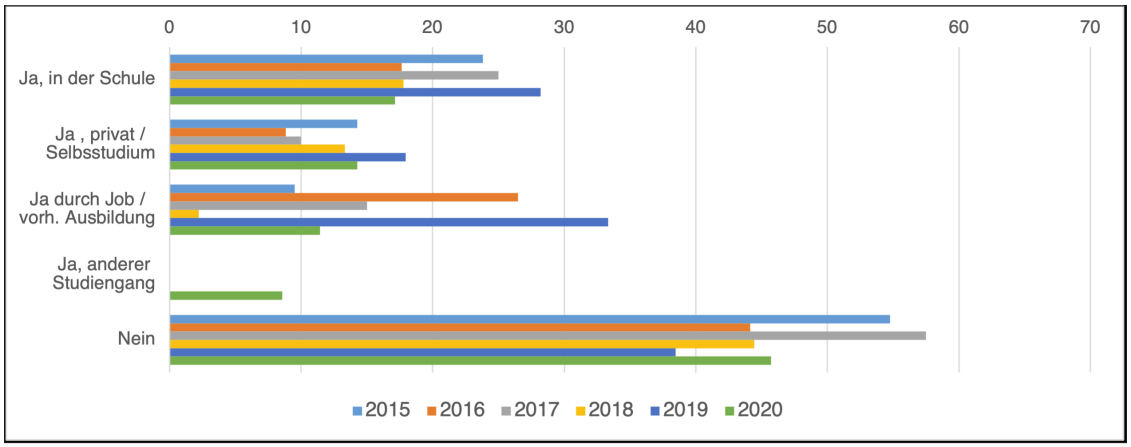

Abbildung 6: Eigeneinschätzung zu Vorkenntnissen in der IT

Die Attraktivität des FIW ist unter Quereinsteigerinnen, d. h. Personen, die schon einen ersten Ausbildungsweg angefangen oder abgeschlossen haben, besonders hoch. Das spiegelt sich in der Frage nach zurückliegenden Ausbildungen wieder, vgl. Abbildung 7.

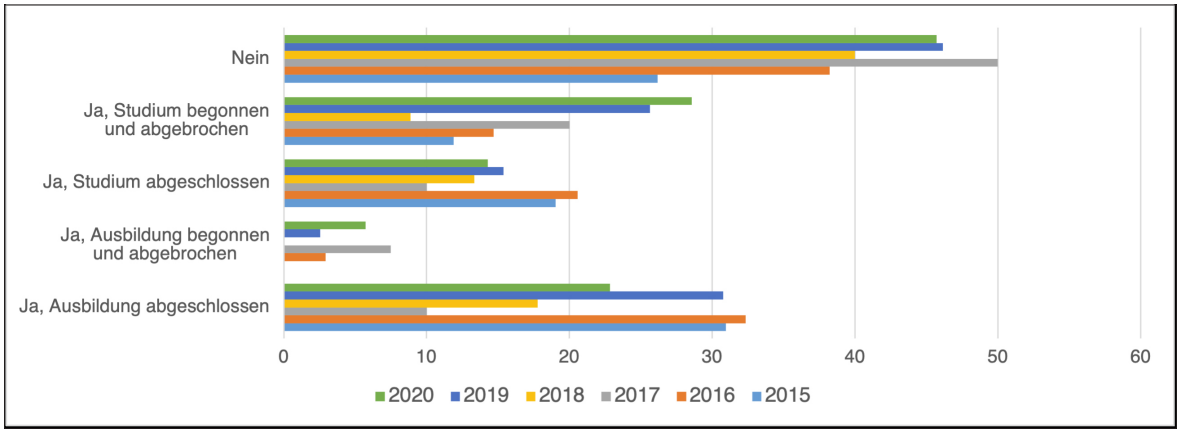

Abbildung 7: Haben Sie bereits eine Ausbildung oder ein Studium abgeschlossen?

Weniger als die Hälfte der Studentinnen (41% der Befragten) kommt direkt vom Abitur. Alle anderen haben schon eine Ausbildung oder ein Studium begonnen oder abgeschlossen[9].

Die Diversität der Lebenswege spiegelt sich auch im Alter der Studienanfängerinnen wieder, vgl. hierzu Abbildung 8. Die größte Gruppe ist die der unter 22-Jährigen (57% der Befragten). Die beiden Folgegruppen der 23-27 Jährigen und der 28-32 Jährigen liegen mit 18% und knapp 16% sehr dicht beieinander. Das Alter der Studentinnen liegt im Vergleich mit anderen Studiengängen des Fachbereichs im Durchschnitt um etwa 2 Jahre höher - bei 25 Jahren, während es zum Beispiel in der Angewandten Informatik und der Wirtschaftsinformatik bei 23,7 bzw. 23,6 liegt.[10]

Der Studiengang wirbt mit familienfreundlichen Präsenzzeiten, vgl. Abschnitt 4. Studentinnen gaben die mündliche Rückmeldung, dass sie explizit nach familienfreundlichen Studiengängen gesucht haben. Dabei haben sie nur diesen gefunden und sich deswegen für Informatik entschieden. Diese Priorität wurde bei der Frage nach der Motivation, vgl. Abbildung 3 erkennbar. Hier gaben 13,5% Familienfreundlichkeit als einen Beweggrund für die Entscheidung an.

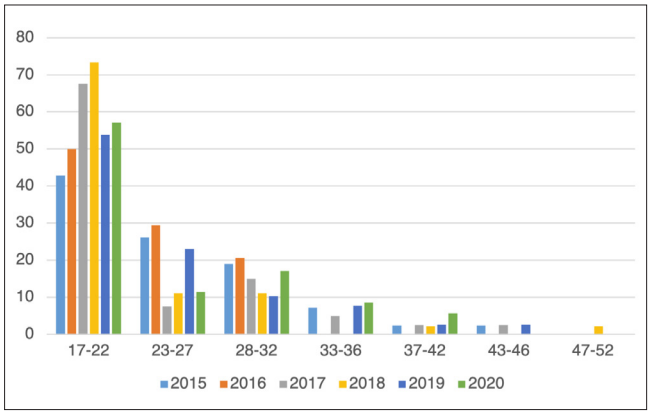

Abbildung 8: Alter der Studentinnen

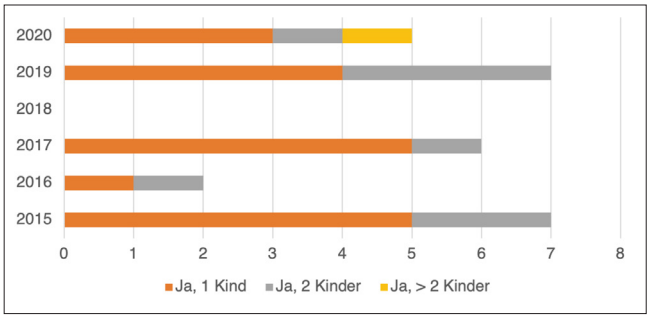

Abbildung 9: # Mütter zu Beginn des Studiums

Um das noch besser einschätzen zu können, wird zu Beginn des Studiums auch nach der Anzahl der Kinder gefragt, vgl. Abbildung 9. Der Anteil der Mütter liegt bei mehr als 11%.[11]

9 Hier ist leider kein Vergleich zum Anteil der Quereinsteigerinnen anderer IT-Studiengänge möglich, da keine Vergleichsdaten dazu vorliegen.

10 Diese Zahlen beruhen auf einer vergleichenden Hochschulstatistik über die Jahre 2012-2015, Quelle: QM-HTW

11 Ein Vergleich mit anderen Studiengängen ist an dieser Stelle nicht möglich, da diese Zahlen nicht erfasst werden.

Zuletzt wird ein Blick auf die Herkunft der Studentinnen geworfen. Es wird der Migrationshintergrund erfragt und geprüft, ob der Studiengang auch für internationale Studentinnen interessant ist. Um den Migrationshintergund zu erheben, wird gefragt: „Haben Sie einen Migrationshintergrund, d.h. sind Sie oder mindestens eines Ihrer Elternteile außerhalb Deutschlands geboren?". Darauf antworten mehr als die Hälfte der Studienanfängerinnen (51,5%) mit Ja.

Ein Vergleich mit anderen Studiengängen der HTW ist bei der Internationalitätsquote möglich. Diese beschreibt das Verhältnis der ausländischen Studierenden[12] zu Studierenden insgesamt. Hier liegt der Studiengang FIW mit 23,7% höher als andere IT-Studiengänge der HTW (AI: 18,3% , WI: 19,1%) und dem Fachbereichsschnitt (21,7%). Eine Ausnahme bildet hier der Studiengang Internationale Medieninformatik mit 29,7% (Quelle: QM-HTW, August 2020).

6 Exkurs - Studiumsverlauf

Alle bisher gezeigten Daten beziehen sich auf die Studienanfängerinnen. Wie in anderen Informatikstudiengängen brechen auch in FIW viele Studentinnen das Studium ab. Zum Vergleich innerhalb der HTW-Berlin wird dafür der Wert der Verbleibquote herangezogen. Diese beschreibt den Anteil der Studierenden im letzten Regelstudienzeitsemester im Vergleich zum Studienbeginn. Der Verbleib ist bei FIW mit 73,8% (Durchschnitt von WS 2009/10-WS 2018/19), vergleichbar mit anderen Bachelor-Informatik-Studiengängen an der HTW (AI: 68,8%, WI: 72,7% und IMI: 76,7%, Quelle: QM-HTW)

Aus der Beobachtung der Xing-Alumnae-Gruppe wird deutlich, dass geschätzt die Hälfte der FIW-Absolventinnen nach dem Studium direkt in die Wirtschaft geht. Die andere Hälfte schließt noch einen Master an. Die konsekutiven Masterstudiengänge Wirtschaftsinformatik und Angewandte Informatik der HTW Berlin sind dafür besonders nachgefragt.

7 Zusammenfassung und Ausblick

Die Erhebung zeigt erste Erkenntnisse aus einer Fragebogenerhebung der Jahre 2015-2020 ergänzt durch Hochschulstatistiken der HTW Berlin. Aus der Kombination der betrachteten Daten lassen sich zwei wesentliche Schlüsse ziehen.

1. Diversität: Die Gruppe von Studentinnen ist in sich sehr divers in den Dimensionen Alter, Herkunft, IT-Erfahrung, Bildungsweg und Elternschaft.

2. Wahl des Studiengangs: Gründe für die Wahl des Studiengangs liegen zuerst bei den guten Berufsaussichten. Die begleitenden Erhebungen deuten für einen wesentlichen Teil der Studierendenschaft auf eine Gruppe hin, die durch andere Informatik-Studiengänge nicht angesprochen werden. Dies wird durch die Tatsache gestützt, dass bei den anderen IT-Studiengängen der HTW mit Einführung von FIW kein Rückgang des Frauenanteils zu verzeichnen war.

Die aufgezeigten Ergebnisse liefern noch keinen empirisch gesicherten Beweis. Sie unterstützen jedoch die These, dass ein wesentlicher Teil der FIW-Studentinnen sich deswegen für ein Informatik-Studium entschieden hat, weil dieses Fach an der HTW mono-edukativ angeboten und entsprechend beworben wird.

Mit den Absolventinnen des Studiengangs kommen damit zusätzliche und sehr verschiedene Frauen in den IT-Arbeitsmarkt - was zu mehr Diversität in der IT-Branche führt.

Der vorgestellte Studiengang ist ein Bachelorstudiengang. Es ist nicht geplant einen monoedukativen Masterstudiengang zu ergänzen. Der Bachelorstudiengang ist als Einstieg in die IT gedacht und funktioniert in diesem Sinn sehr erfolgreich: Mit dem Angebot wird eine neue Klientel von Studentinnen für die Informatik erschlossen. Solange es dazu führt, dass mehr Frauen sich für MINT-Studiengänge entscheiden, sind mono-edukative Studiengänge damit ein Mittel der Wahl für mehr Diversität.

12 inklusive der Bildungsausländer*innen

Literatur

[ACM17] Alvarado, C.; Cao, Y.; Minnes, M.: Gender Differences in Students' Behaviors in CS Classes throughout the CS Major. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. Association for Computing Machinery, Seattle, Washington, USA, S. 27–32, 2017.

[BI18] BITKOM: Informatik-Hörsäle werden langsam weiblicher, 2018, url: www.bitkom.org/Presse/Presseinformation/Informatik-Hoersaelewerden-langsam-weiblicher.html, Stand: 21.03.2019.

[BI20] BITKOM: 86.000 offene Stellen für IT-Fachkräfte, 2020, url: www.bitkom.org/Presse/Presseinformation/86000-offene-Stellen-fuer-IT-Fachkraefte, Stand: 21.03.2021.

[De21] Degener, J.: Überschaubares Angebot - Frauenstudiengänge in Deutschland, 2021, url: www.studis-online.de/Studienfuehrer/frauenstudiengaenge.php, Stand: 12.08.2021.

[Fr18] Friedrich, J.-D.; Hachmeister, C.-D.; Nickel, S.; Peksen, S.; Roessler, I.; Ulrich, S.: Frauen in Informatik: Welchen Einfluss haben inhaltliche Gestaltung, Flexibilisierung und Anwendungsbezug der Studiengänge auf den Frauenanteil?, Gütersloh, 2018, url: www.che.de/download/che_ap_200_frauen_in_informatik-pdf.

[Fr19] Friedrich, J.-D.; Hachmeister, C.-D.; Nickel, S.; Peksen, S.; Roessler, I.; Ulrich, S.: Frauen in IT: Handlungsempfehlungen zur Gewinnung von Frauen für Informatik, 2019, url: www.che.de/wp- content/uploads/upload/CHE_AP_222_Handlungsempfehlung.pdf.

[SSB18] Siegeris, J.; Steinseifer, R.; Barke, H.: Agile Redesign of Student Projects in a Women-Only Degree Course. In: GenderIT '18, Association for Computing Machinery, Heilbronn, Germany, 2018.

[St19] Statistisches Bundesamt: Männerberufe, Frauenberufe? Klassische Rollenbilder bestimmen noch immer die Arbeitswelt. 2019, url: https:%20//www.destatis.de/DE/Presse/Pressemitteilungen/2019/11/PD19_%20N009_122.html.

Wie können wir Lehramtsstudierende auf einen inklusiven Informatikunterricht vorbereiten?

Kensuke Akao[1], Johannes Fischer[2]

Abstract:
Die aktuelle Bildungspolitik in Deutschland fordert die Umsetzung von Inklusion an Schulen. Die Ergebnisse einer Online-Lehrerumfrage in Nordrhein-Westfalen deuten jedoch darauf hin, dass es oft an Informatiklehrkräften fehlt, die in der Aus- oder Fortbildung genügend Kenntnisse zum inklusionsorientierten Fachunterricht erworben haben. Daher ist es die Aufgabe der universitären Fachdidaktiken, für eine entsprechende Ausbildung zu sorgen. Unsere Hochschule forschte in den letzten Jahren zur Weiterentwicklung der inklusionsorientierten Lehrerbildung. Aufgrund unserer vorangegangenen Ergebnisse in der Hochschulpraxis wissen wir jedoch, dass fast alle unsere Informatiklehramtsstudierenden keine Erfahrung mit dem Lernen in einer Inklusionsklasse aus ihrer Schulzeit haben. Damit die Studierenden eine inklusive Lehr-Lernsituation besser verstehen, kombinierten wir Ideen für den inklusiven Informatikunterricht mit Sensibilisierungsideen für den Förderbedarf inklusiver Kinder. In diesem Beitrag machen wir erste Vorschläge für die Entwicklung geeigneter Formate in der inklusionsorientierten Informatiklehramtsausbildung.

Keywords:
Informatikunterricht; Inklusion; inklusionsorientierte Lehrerbildung

1 Einleitung

Mit der Ratifikation der *Convention on the Rights of Persons with Disabilities (CRPD)* fordert die aktuelle Bildungspolitik in Deutschland die Umsetzung der Inklusion an Schulen [BD14]. Beispielsweise änderte sich in Nordrhein-Westfalen deshalb im Jahr 2014 das Schulgesetz § 2 (5) entsprechend, und Inklusion soll somit an Schulen implementiert werden. Daher sollen sich die Lehrkräfte auf die Umsetzung der Inklusion vorbereiten, indem sie nötige Kompetenzen erwerben. Laut Leonhardt [Le11] soll Wissen zum inklusiven Unterricht bereits in der universitären Lehrerausbildung erworben werden, um dann in Fort- und Weiterbildungen vertieft werden zu können. Da im Schulfach Informatik oft am Computer gearbeitet wird [HNR06], sind zudem Kenntnisse zum Einsatz assistiver Technologien (z. B. *Screen Reader*) unabdingbar [CG16]. Das nötige fachdidaktische und technologische Wissen muss also im Studium erworben werden.

Die Ergebnisse einer Online-Umfrage deuten darauf hin, dass die Behandlung des Themas „Inklusiver Informatikunterricht" an vielen Hochschulen in der Didaktik der Informatik (DDI) in den letzten Jahren schrittweise angestiegen ist, um diesen Lehrerbildungsbedarf zu decken [AF21]. Seit 2019 beteiligen wir uns an der Weiterentwicklung der inklusionsorientierten Lehrerbildung mit einem Informatik-Teilprojekt im Rahmen eines interdisziplinären Forschungsprojekts. Unsere Bemühungen zielen darauf ab, dass unsere Lehramtsstudierende durch die DDI-Veranstaltung praxisorientierte und anpassungsfähige Kompetenzen für die Umsetzung inklusionsorientierten Unterrichts erwerben. Dabei besteht ein wichtiges Lernziel darin, Studierenden den Umstand vor Augen zu führen, dass sie in Zukunft Kinder unterrichten müssen, deren Behinderung sie heute noch nicht kennen. Problematisch ist, dass fast alle unsere Studierenden noch keine realen inklusiven Lehr-Lernsituationen in der Schulpraxis kennen, weil sie in ihrer eigenen Schulzeit niemals das Lernen in einer Inklusionsklasse erlebten.

1 Technische Universität Dortmund, Fakultät für Informatik, Arbeitsgruppe Algorithmische Grundlagen und Vermittlung der Informatik, Otto-Hahn-Str. 14 44227 Dortmund
kensuke.akao@tu-dortmund.de

2 Technische Universität Dortmund, Fakultät für Informatik, Arbeitsgruppe Algorithmische Grundlagen und Vermittlung der Informatik, Otto-Hahn-Str. 14 44227 Dortmund
johannes.fischer@cs.tu-dortmund.de

Unser Ansatz ist, unsere Studierenden mithilfe von Behinderungssimulationen und Erfahrungsberichten einer von einer Behinderung betroffenen Person für den Förderbedarf im inklusiven Unterricht zu sensibilisieren. Dieser Beitrag wird die empirischen Erkenntnisse unserer Hochschulpraxis mit unseren ersten Ideen zur inklusionsorientierten Informatiklehrerausbildung beschreiben, die den Schwerpunkt auf die Sensibilisierung für den Förderbedarf legt.

2 Stand der inklusionsorientierten Informatiklehramtsausbildung

2.1 Aktuelle Bereitschaft zur Umsetzung der inklusiven Informatik

Zuerst fassen wir die Ergebnisse aus zwei kürzlichen Umfragen [AF21, AF20] zusammen, in denen ein Bild vom Stand der inklusionsorientierten Informatiklehramtsausbildung in Deutschland gezeichnet wird:

In der Studie 2020[3] gaben „19 der 34 Befragten [...] an, dass das Thema ‚Inklusiver Informatikunterricht' im Informatik-Lehramtsstudium an ihrer Hochschule *zumindest teilweise* bereits umgesetzt wird [...]. Zehn der 17 Befragten antworteten, dass ihre Studierenden im Bachelor- und Masterstudiengang DDI-Veranstaltungen mit dem Thema Inklusion belegen; weitere drei ‚nur im Bachelorstudiengang' und vier ‚nur im Masterstudiengang'. [...] Fünf Befragte denken aktuell über die Umsetzung des Themas nach [...]. Weitere sechs Befragte sind der Meinung, dass es noch keine Notwendigkeit für die Umsetzung des Themas in DDI-Veranstaltungen gibt [...] [AF21]".

Die Studie 2019[4] verdeutlicht die mangelnden Kenntnisse der Lehrkräfte über Inklusion. Insbesondere hatten ca. 46% der Befragten noch gar keinen Kenntniserwerb in diesem Bereich (n=188); nur sieben der 13 Befragten, die weniger als fünf Jahre in der Schulpraxis stehen (inkl. Referendariat), erwarben beim Lehramtsstudium Kenntnisse dazu, obwohl die damalige Bildungspolitik schon die Inklusion fördern sollte [AF20].

2.2 Wie wurde das Thema „Inklusiver Informatikunterricht" umgesetzt?

Im Anschluss analysieren wir die Ergebnisse der Studie 2020 [AF21] basierend auf den 17 detaillierteren Antworten zur Umsetzung in DDI-Veranstaltungen im Detail. Aus der Mehrfachauswahlfrage „Um welche Art von DDI-Veranstaltung handelt es sich bei dem Thema Inklusion?" wurde deutlich, dass das Thema bei verschiedenen Arten von Veranstaltungen umgesetzt wird, wie Tab. 1 zeigt.

Tab. 1: Art von DDI-Veranstaltung mit dem Thema Inklusion (n=17)

Art von DDI-Veranstaltung	Beide Studiengänge	Nur Bachelor	Nur Master	Summe
Vorlesung mit Übung + Seminar mit Übung + Praktikum	3	0	0	3
Vorlesung mit Übung + Seminar	2	1	0	3
Vorlesung mit Übung	1	1	1	3
Seminar + Praktikum	2	0	0	2
Vorlesung	0	1	1	2
Seminar	0	0	2	2
Vorlesung + Praktikum	1	0	0	1
Vorlesung + Vorlesung mit Übung + Übung + Seminar	1	0	0	1

3 Diese Umfrage wurde vom 1. Oktober bis 5. November 2020 im Internet als Online-Form durchgeführt. Der Aufruf zur anonymen Teilnehme wurde basierend auf dem Suchergebnis Mit dem Stichwort „Lehramt Informatik" im Hochschulkompass (www.hochschulkompass.de/home.html: Stand 21. September 2020) an 54 Verantwortliche für die Lehramtsstudiengänge Informatik gesendet. Wir erhielten 34 Rückmeldungen von Hochschulen aus 10 Bundesländern: Baden-Württemberg, Bayern, Berlin, Hessen, Niedersachsen, Nordrhein-Westfalen, Rheinland-Pfalz, Saarland, Sachsen und Schleswig-Holstein. Allerdings existieren mehr Rückmeldungen als Hochschulen aus den Ländern Nordrhein-Westfalen und Rheinland-Pfalz. Deshalb muss mit einer leicht verfälschten Auswertung gerechnet werden; den Gesamteindruck dürfte dies jedoch nicht trüben.

4 Diese Umfrage wurde vom 20. September bis 31. Oktober 2019 im Internet als Online-Form durchgeführt. Der Aufruf zur anonymen Teilnehme wurde via E-Mail an 2123 Schulen aus allen Regierungsbezirken in Nordrhein-Westfalen gesendet; wir erhielten insgesamt 219 gültige Rückmeldungen (186 vollständig beantwortete Fragebogen); Die Anzahl der Teilnehmende beträgt 104 Informatiklehrkräften (inkl. 30 Lehrkräfte, die auch Fächer mit informatischen Inhalten unterrichten, und 19 Lehrkräfte, die sich auch mit der Schulleitung beschäftigen), 32 Lehrkräften, die Fächer mit informatischen Inhalten unterrichten, (inkl. 5 Lehrkräfte, die sich auch mit der Schulleitung beschäftigen) und 83 Schulleiter/innen bzw. stellvertretende Schulleiter/innen ohne Aufgabe bezüglich der Informatik.

Abb. 1 zeigt die Ergebnisse der Frage „Welche inklusionsorientierten Lehr-Lerninhalte werden schwerpunktmäßig in den DDI-Veranstaltungen thematisiert?“ mit einer fünfstufigen Likert-Skala (1: trifft überhaupt nicht zu - 5: trifft voll zu) für jeden der sieben Lehr-Lerninhalte. Die Grundkenntnisse wurden öfter als andere Inhalte umgesetzt, jedoch wurden praxisbezogene Inhalte wie Fallbeispiele oder aktuelle Trends und Entwicklungen eher selten thematisiert. Weiterhin wird die Auswertung der Antworten zur Frage „Mit welchen informatischen Inhalten wird das Thema Inklusion in den DDI-Veranstaltungen kombiniert?“ über eine fünfstufige Auswahl (1: trifft überhaupt nicht zu - 5: trifft voll zu) in Abb. 2 wiedergegeben. Tendenziell wurde von den Befragten die Kombination des Themas Inklusion mit dem Inhaltsfeld „Informatiksysteme“ und „Informatik, Mensch und Gesellschaft“ am häufigsten angegeben.

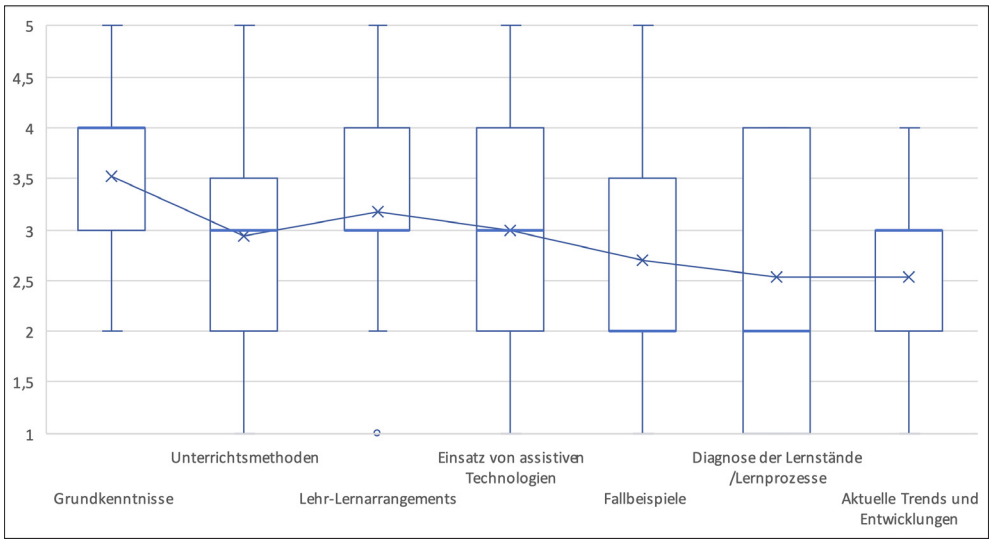

Abb. 1: In den DDI-Veranstaltungen thematisierte inklusionsorientierte Lehr-Lerninhalte (n=17)

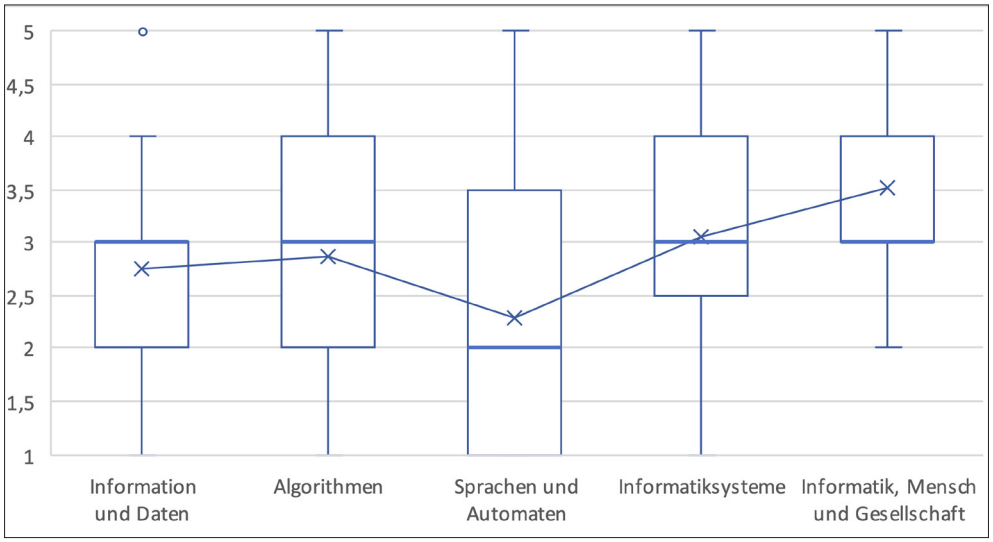

Abb. 2: Informatische Inhalte, die mit dem Thema Inklusion kombiniert werden (n=17)

2.3 Lernten Studierenden schon in einer Inklusionsklasse in ihrer Schulzeit?

Zum Zeitpunkt der ersten Konzeption unserer inklusionsorientierten Informatiklehrerausbildung war uns bewusst, dass erst fünf Jahre seit der Änderung des Schulgesetzes in Nordrhein-Westfalen § 2 (5) vergangen sind. Deshalb war unsere Hypothese, dass die Lehramtsstudierenden, die wir zurzeit ausbilden, mit hoher Wahrscheinlichkeit zu einer Zeit zur Schule gingen, als die Inklusion noch nicht umgesetzt worden war. Da seit dem Sommersemester 2019 insgesamt 23 Masterstudierende an unserer DDI-Veranstaltung teilnahmen oder aktuell teilnehmen, ist das Bild über die tatsächliche Situation in ihrer Schulzeit allmählich klarer geworden.

Wir fragten diese 23 Masterstudierende und 12 DDI-Studierende, die noch nicht an unserer Veranstaltung teilnahmen, über ihre Erfahrungen mit Inklusion. Sechs Studierende gaben an, dass an ihrer Schule Inklusion umgesetzt wurde, als sie noch Schüler*innen waren (n=35). Nur zwei der sechs Studierenden lernten jemals in ihrer Schulzeit in einer Inklusionsklasse, jedoch lernten sie nicht Informatik in einer Inklusionsklasse.

Da in Deutschland nur ca. 4.000 Studierende[5] auf einen Abschluss Lehramt mit Informatik hin studieren, ist die Anzahl dieser Studierenden pro Hochschule oft gering [Wi20, S. 19]. Deshalb war es schwierig, auf Basis der wenigen Rückmeldungen quantitative Auswertungen vorzunehmen, jedoch lässt sich basierend auf diesen Tendenzen die Gestaltung unserer Lehre für die Studierenden optimieren. Im nächsten Kapitel wird das Konzept unserer Hochschulpraxis - eine Kombination der konkreten Umsetzungsbeispiele für einen informatikspezifischen Inklusionsunterricht mit Aktivitäten zur Sensibilisierung für den Förderbedarf der inklusiven Kinder - vorgestellt.

3 Weiterentwicklung inklusionsorientierter Lehrerausbildung

3.1 Ideen für eine inklusionsorientierte Informatiklehrerausbildung

Wir setzen das Thema „Inklusiver Informatikunterricht" in der Übung zur Didaktik der Informatik im Masterstudiengang als ein halbtägiges Blockseminar um. Nun kommen wir zu den von uns umgesetzten Ideen für den inklusiven Informatikunterricht.

Wir verstehen die Aufarbeitung informatischer Lerninhalte (u. a. der in den Inhaltsbereichen der *GI-Bildungsstandards* [Ge08] genannten Inhalte) für inklusiven Unterricht als informatikspezifische Inklusion. Da wir im Rahmen der 1. Phase der Lehrerbildung hierbei eigentlich zu informatikspezifischen Teilen der inklusionsorientierten Lehrerausbildung beitragen, setzen wir voraus, dass theoretisches Wissen zur Inklusion im nicht-fachspezifischen Teil des Lehramtsstudiums (u. a. in den Kernmodulen der Bildungswissenschaften) schon ausreichend angesprochen worden ist. Aus diesem Grund legt unser Ansatz den Schwerpunkt darauf, was die Informatiklehrkräfte praktisch in einer Inklusionsklasse für ihre Schüler*innen tun sollen/können. Wichtig ist, dass wir unseren Studierenden nicht die Kompetenz für die Umsetzung eines bestimmten Unterrichtsvorhabens vermitteln, das zur Inklusion geeignet und schon fertig entwickelt ist, sondern die Richtung für die Anpassung der von ihnen entwickelten Unterrichtsvorhaben für die Inklusion festlegen, indem sie mithilfe praxisbezogener Beispiele von uns Inspirationen für die Entwicklung neuer Ideen bekommen. So etablieren wir durch die Lehre ihre Einstellung zur Inklusion und insbesondere die Bereitschaft, auf individuelle Bedürfnisse einzugehen.

Als Beispiel betrachteten wir die Blindheit, weil a) wir Kontakte zu Forschenden aus dem Bereich „Sonderpädagogik für die Blindheit und schwere Sehbehinderung" haben, b) zwei Sachverständige mit einer Sehbehinderung zu unserem Projekt gehören und c) es eine Fülle von Ideen für die Simulationen gibt[6]. Zudem gibt es schon erste methodische Ideen für diese Art von Behinderung (z. B. [CKH13]). Wir thematisieren in der DDI-Veranstaltung die informatikspezifische Inklusion anhand von Beispielen zur Kodierungstheorie und Programmierung.

Abb. 3 zeigt die behandelten Beispiele von Hilfsmitteln im Kontext Informatikunterricht. In der Kodierungstheorie nutzen wir ein spezielles „Schwellpapier", das sich durch Erhitzung in den vorher schwarz markierten Bereichen ausdehnt. Sie stellen Barcodes dar, die sowohl gesehen als auch gefühlt werden können. Die Kinder können somit etwas über die Kodierungstheorie lernen, indem sie die Barcodes wie ein Puzzle legen. Das Ergebnis wird zur Selbstkontrolle mithilfe eines Barcode-Scanners gescannt. Als zweites Hilfsmittel für den Programmierungsunterricht stellt „Code Jumper"[7] eine Möglichkeit zur Umsetzung der blockorientierten Programmierung für Menschen mit Blindheit oder schwerer Sehbehinderung dar. Beim Programmieren fügen Kinder die Programmierbefehls-Bausteine (Hardware-Komponente), deren Bedeutung man haptisch erkennen kann, mit Kabeln elektrisch aneinander; die Ausgabe erfolgt akustisch. Als letztes Hilfsmittel im Kontext DDI wird ein Arduino-Projekt mit einem Servomotor und Bananensteckern behandelt. Durch diese Anpassung wird das Problem gelöst, dass Menschen mit Blindheit oder Sehbehinderung nicht so einfach einen Stecker richtig einstecken können und optische Ausgaben wie blinkende LEDs nicht erkennen können. Dabei hat die Verwendung des Servomotors, der sich mit hohem Drehmoment bewegt, auch den Vorteil, dass man die Ausgabe optisch, haptisch und akustisch wahrnehmen kann.

---

5 Gesamtzahl der Bachelor- und Masterstudierenden im Informatiklehramt als erstes, zweites und drittes Fach.

6 Z. B. bietet der „Dialogue in the Dark" dies seit 1988 an (vgl. www.dialogue-se.com/what-we-do/dialoguein-the-dark, zuletzt geprüft am: 9.2.2021). Wir tauschten Meinungen bei der Planung dieses Seminars mit einem Mitarbeiter von Dialogue in the Dark aus, der selber von Blindheit betroffen ist.

7 Vgl. https://codejumper.com (zuletzt geprüft am: 9.2.2021)

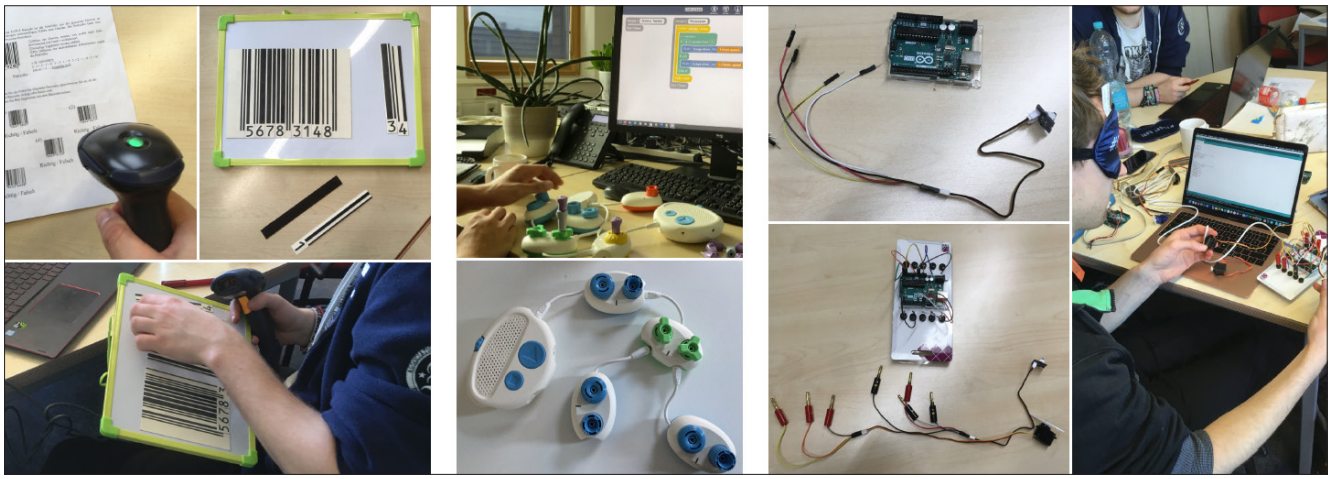

Abb. 3: Barcode aus Schwellpapier (links), Code Jumper (Mitte) und Arduino-Projekt (rechts)

Außerdem haben unsere Studierenden oft keine Erfahrung mit assistiver Technologie. Aus diesem Grund werden nicht nur informatikspezifische Unterrichtsmethoden oder Lehr-Lernarrangements, sondern auch allgemeine Hilfsmittel im Zusammenhang mit der Informationstechnik thematisiert. Als Beispiele für allgemeine Hilfsmittel werden ein „Knochenleitungskopfhörer" für den Einsatz von Screen Readern in lauten Klassenzimmern und ein „Digitalstift"[8] für das Ablesen nicht digitalisierter Lehrmaterialien behandelt (Abb. 4). Wenn man mithilfe der Knochenschallleitungstechnologie die Informationen des Screen Readers hört, kann man die Gespräche im Klassenzimmer und die Tonausgabe des Computers gleichzeitig vernehmen. Bei der Übung vergleichen die Studierenden die Hörweise zwischen dem normalen Kopfhörer und dem Knochenleitungskopfhörer, indem sie beim Hören z. B. laut sprechen, Ohrstöpsel in ihre Ohren stecken oder gleichzeitig beide Kopfhörer tragen. Da der Screen Reader nur digitalisierte Materialien vorlesen kann, wird auch noch der Digitalstift als ein weiteres Beispiel vorgestellt. Diese Idee aus Japan wird mittlerweile weltweit zur inklusionsorientierten Aufarbeitung von Lehrmaterialien in vielen Fächern eingesetzt [Ik20].

Abb. 4: Erprobung: Knochenschallleitung (links) und Digitalstift (rechts)

3.2 Sensibilisierung für den Förderbedarf der inklusiven Kinder

Unser Fokus liegt darauf, dass in der CRPD hauptsächlich die zwei Ansätze „Reasonable Accommodation"[9] und „Universal Design" zur Anpassung für die Inklusion angegeben werden (siehe CRPD § 24). *Universal Design for Learning (UDL)* für den Bildungsbereich beruht auf der Anwendung der Universal-Design-Idee, möglichst für alle Menschen hilfreiche Anpassungen zu schaffen [Ro01]. Laut dem *AAA-Modell*[10] [SSE01] beginnt die bedarfsgerechte Anpassung mit der Erfassung der einzelnen Bedürfnisse von Menschen mit Behinderungen; die Umsetzung der Barrierefreiheit und/oder zum Universal Design erfolgt erst danach schrittweise. Das bedeutet für die Hochschuldidaktik, dass die zukünftigen Lehrkräfte zunächst erst einmal für die Bedürfnisse der inklusiven Kinder sensibilisiert werden müssen.

8 Z. B. G-Speak (vgl. http://gridmark.co.jp/en/solution, zuletzt geprüft am: 9.2.2021)

9 Auf Deutsch etwa: „angemessene Vorkehrungen". Die *Reasonable Accommodation* bedeutet „notwendige und geeignete Änderungen und Anpassungen, die keine unverhältnismäßige oder unbillige Belastung darstellen und die, wenn sie in einem bestimmten Fall erforderlich sind, vorgenommen werden, um zu gewährleisten, dass Menschen mit Behinderungen gleichberechtigt mit anderen alle Menschenrechte und Grundfreiheiten genießen oder ausüben können (CRPD § 2)".

10 Advocacy-Accommodation-Accessibility

Aus diesem Grund legen wir im Seminar den Schwerpunkt auf die Sensibilisierung für den Förderbedarf. Aber wenn unsere Studierenden als Schüler*innen selbst niemals in einer Inklusionsklasse lernten, können ihre ehemaligen Lehrkräfte kein Vorbild für Lehr-Lernsituationen zur Umsetzung der Inklusion sein. Hier erkannten wir den Bedarf, eine praktische Einführung dazu zu geben, wie man den Förderbedarf inklusiver Kinder erfassen kann/soll. Daher werden im Seminar zunächst Behinderungssimulationen durchgeführt, bevor die Studierenden die im Abs. 3.1 vorgestellten konkreten Ideen für inklusiven Informatikunterricht kennenlernen.

Fotos bei der Behinderungssimulationen werden in Abb. 5 gezeigt. Eine Simulation ist die „Home Party“: dabei versuchen die Studierenden u. a., mit einer Augenbinde Kerzen anzuzünden oder zu löschen, die Geschmäcker von zufällig aus einer Tüte gezogenen Gummibärchen zu erraten oder Limonade in ein Glas zu gießen. Diese Erfahrungen können bald im Lernen zur Kodierungstheorie und Programmierung mit Code Jumper/Arduino verknüpft werden und veranschaulichen, wie ein Unterricht inklusiv gemacht werden kann/soll. Eine weitere Simulation ist „Fernsehen hören“. Die Studierenden vergleichen ein Animationsvideo mit und ohne Hörfassung für blinde und sehbehinderte Menschen, indem sie es mit einer Augenbinde anhören. Dann schauen sie auch das Video ohne Augenbinde und diskutieren zum Schluss über die Unterschiede der erhaltenen Informationen. Diese Simulation hilft beim Verstehen der Bedeutung und Weise des Ablesens besser, insbesondere die Erklärung der Inhalte im Material ohne Verwendung visueller Darstellung.

Abb. 5: Blindheitssimulationen „Home Party“ (links) und „Fernsehen hören“ (rechts)

Das Konzept wird mithilfe eines Forschungsmodells *Design-based research* (vgl. [Th03]) in jedem Semester erprobt, evaluiert und verbessert. Ein Verbesserungsvorschlag ist, dass wir ab der zweiten Iteration einen schwer sehbehinderten Informatikstudenten, der in seiner Schulzeit selbst in einer Inklusionsklasse lernte, in das Seminar einladen. Dabei erwarten wir, dass unsere Studierenden durch das Gespräch mit ihm ein Bild von inklusiver Lehr-Lernsituation erleben können, insbesondere mithilfe seines Erfahrungsberichts. Nach aktuellem Stand besteht das Seminar aus folgenden Inhalten:

1. Wiederholung von allgemeinen Grundkenntnissen zur Inklusion,

2. Sensibilisierung durch Behinderungssimulationen,

3. Inklusiver Informatikunterricht (s. 3.1),

4. Sensibilisierung durch den Austausch mit einer betroffenen Person.

4 Evaluation

4.1 Erprobung des Seminars

Das von uns entwickelte Blockseminar wurde im Wintersemester 2019/20 erstmals erprobt. In jenem Semester nahmen drei Masterstudierende daran teil. Da das Seminar eigentlich für eine Präsenzveranstaltung entwickelt wurde, trat im Sommersemester 2020 wegen des COVID-19-Lockdowns das Problem auf, dass weder vollständige Blindheitssimulationen online durchgeführt noch Hilfsmittel ausprobiert werden konnten. Aus diesem Grund musste das Vorhaben zunächst an das digitale Format angepasst werden.

Im Wintersemester 2020/21 wurde das Seminar als ein Tele-Learning Seminar durch eine Kombination von Online-Sitzungen und dem Paketversand der Hilfsmittel aufgebaut, damit Teilnehmende auch zu Hause praktische Ideen zum Thema Inklusion ausprobieren können. Dabei meldeten sich vier Masterstudierende für das Seminar an und zwei Studierende bearbeiteten alle Aufgaben vollständig.

4.2 Quantitative Analyse der Lernergebnisse

Nach dem Seminar wurden die Teilnehmenden nach einer Selbsteinschätzung der Lernergebnisse mithilfe eines Online-Fragebogens auf die Lernplattform Moodle gefragt (Wintersemester 2019/20: n=3; Wintersemester 2020/21: n=3). Bezüglich der Zufriedenheit mit diesem Seminar stellten wir die Frage „Würden Sie das Seminar auch anderen Lehramtsstudierenden empfehlen?", deren Beantwortung über eine sechsstufige Likert-Skala (1: gar nicht gerne – 6: sehr gerne) möglich war. Im Wintersemester 2019/20 wurde in allen drei Rückmeldungen „sehr gerne (6)" gewählt. Im Wintersemester 2020/21 gaben zwei Teilnehmende „sehr gerne (6)" an und eine weitere Person gab „eher gerne (4)" an.

Die Ergebnisse übriger drei quantitativen Befragungen mit einer sechsstufigen Likert-Skala (1: stimme gar nicht zu – 6: stimme vollkommen zu) wurden in Tab. 2 zusammengefasst. Bei der Frage nach der Nützlichkeit der gewonnen Erkenntnisse stimmten fünf Teilnehmenden zu (5) und eine weitere Person stimmte vollkommen zu (6), dass sie sich durch das Seminar besser als vorher auf inklusive Schulklassen vorbereitet fühlen. Drei Teilnehmenden spüren nach dem Seminar eher Bedenken (4), weitere zwei Teilnehmende keine Bedenken (2) und die letzte Person gar keine Bedenken (1). Zudem hatten zwei Teilnehmende im Wintersemester 2019/20 sowie alle drei Teilnehmende im Wintersemester 2020/21 nach dem Seminar weniger Bedenken als davor, aber die übrige Person im Wintersemester 2019/20 hatte eher mehr Bedenken. Außerdem stimmte eine Person die Frage „Motiviert das Seminar Sie zur Vertiefung Ihres Wissens über Inklusion, um sich in Zukunft in der Schulpraxis die Umsetzung der Inklusion aktiv anzugehen?" vollkommen zu (6), eine weitere Person zu (5) und die drei übrigen Teilnehmenden stimmten eher zu (4), während eine andere Person eher nicht zustimmte (3).

Tab. 2: Ergebnis des Feedbacks (1: stimme gar nicht zu – 6: stimme vollkommen zu)

Frage	Semester	Anzahl der Bewertung					
		1	2	3	4	5	6
Fühlen Sie sich durch das Seminar besser als vorher vorbereitet auf inklusive Schulklassen?	WS2019/20	0	0	0	0	3	0
	WS2020/21	0	0	0	0	2	1
Spüren Sie Bedenken, dass Sie in Zukunft in einer Inklusionsklasse unterrichten müssen?	WS2019/20	0	0	0	3	0	0
	WS2020/21	1	2	0	0	0	0
Motiviert das Seminar Sie zur Vertiefung Ihres Wissens über Inklusion, um sich in Zukunft in der Schulpraxis die Umsetzung der Inklusion aktiv anzugehen?	WS2019/20	0	0	0	2	0	1
	WS2020/21	0	0	1	1	1	0

4.3 Qualitative Analyse der Sensibilisierung mit den Blindheitssimulationen

Nach dem Seminar im Wintersemester 2019/20 wurde den drei Teilnehmenden die Aufgabe gegeben, einen ca. 1-2-seitigen Reflexionsbericht zu schreiben. Wir analysieren nun jede Person (A-C) qualitativ, was sie durch die Blindheitssimulationen lernten.

Die **Person A** ist darauf aufmerksam geworden, dass auf verschiedene Förderungsbedarfe mit unterschiedlichen Maßnahmen reagiert werden muss. Sie fand es nun wichtig, dass sie die Fähigkeit der Improvisation für die Anpassung inklusiver Lehr-Lernsituation auch beherrschen soll, weil sich der Unterricht nicht für alle Personen so präventiv vorbereiten lässt. Die **Person B** verstand,

dass sie nicht vollständig die Situation der betroffenen Kinder begreifen kann, weil ihr durch die Behinderungssimulation erstmals selbst bewusst geworden ist, welche Art von Hilfe sie in dieser Situation bekommen wollen würde. Deshalb findet sie es wichtig, dass sie die Anforderungen von dem Inklusionskind genau erfasst, damit eine Maßnahme das Kind auch wirklich unterstützt. Für die **Person C** war die Erfahrung der Blindheitssimulation, bei der sie mit verbundenen Augen die Schwierigkeiten erkannte, bei der Erprobung der Hilfsmittel für inklusiven Unterricht hilfreich. Insbesondere der Digitalstift war ein interessantes Hilfsmittel, weil sie die Verwendbarkeit dieser simplen Maßnahme für Menschen mit einer Sehbehinderung am eigenen Leib spürte. Sie berichtet auch, dass die Tonaufnahme für die Lehrkräfte tatsächlich nicht so aufwendig ist.

4.4 Qualitative Analyse der Sensibilisierung mit dem Erfahrungsbericht

Nach dem Seminar im Wintersemester 2020/21 gab ein Student, der vor allem mit dem Gespräch zur Sensibilisierung zufrieden war, schriftlich sein Feedback. Im Folgenden wird sein Kommentar zusammengefasst.

Er fand das Gespräch mit der sehbehinderten Person sehr wertvoll und positiv, weil in seinem bisherigen Studium wenig zur Inklusion angeboten worden ist und er persönlich wenige Berührungspunkte mit Inklusion hatte. Er bekam auch als Lehramtsstudent nur selten Informationen über den Wandel zur Inklusion in der Schulpraxis mit, deshalb konnte er beim Gespräch endlich ein Bild bekommen, auf welchem Stand öffentliche Schulen beim Thema Inklusion sind. Im Gespräch war für ihn besonders folgende Aussage eindrucksvoll: „Am schlimmsten ist es, wenn die Leute denken, sie müssten mir helfen und wüssten, was das Beste für mich ist, und sie dann über meinen Kopf hinweg Dinge für mich entscheiden und mich bevormunden, statt einfach mit mir zu reden. Denn ich weiß schließlich am besten, was das Beste für mich ist. Ich lebe mit dieser Behinderung schon mein ganzes Leben."Dabei lernte er, dass es viel sinnvoller ist, wenn man direkt mit dieser Person kommuniziert und sie fragt, was sie braucht, statt mithilfe theoretischer Arbeiten eine Lösung zu suchen.

Seine Aussagen erwecken den Eindruck, dass er die wesentliche Strategie für die Umsetzung von Inklusion (= Reasonable Accommodation) gut verstanden hat und das Erlernte seine Bedenken reduziert hat, Lernende mit Behinderungen unterrichten zu müssen. Er weiß nun, wie man mit ihnen in Dialog tritt, um sie ernst zu nehmen, sie zu respektieren und dann gemeinsam die besten Lösungen zu finden. Außerdem gewann er eine gute kritische Sichtweise darauf, dass manche aktuelle Herausforderungen zur Inklusion für vielfältige und individuell Lehr-Lernsituation versuchen, es in einen theoretischen Rahmen basierend auf typischen Fällen von Behinderung anzuwenden. Zum Schluss machte er den Verbesserungsvorschlag, dass die Teilnehmenden die Gelegenheit bekommen sollen, sich vorab auf das Gespräch vorbereiten zu können, indem sie ihre Meinungen, Unklarheiten und Vorstellungen über Inklusion reflektieren.

5 Fazit und Ausblick

Die Bildungspolitik in Deutschland zielt auf die Stärkung der Inklusion in den Schulen ab, deshalb sollten alle Informatiklehramtsstudierenden auf die Umsetzung eines inklusiven Unterrichts vorbereitet werden. Um diesen Lehrerbildungsbedarf zu decken, versuchen wir auch das Thema „inklusiver Informatikunterricht" zu behandeln. Jedoch stehen diese Bemühungen erst noch am Anfang. Fakt ist, dass fast alle unsere Informatiklehramtsstudierenden keine Lernerfahrungen in einer Inklusionsklasse in ihrer Schulzeit hatten. Außerdem mangelt es in beinahe allen Fällen unseren DDI-Studierenden an sonderpädagogischem Wissen. Deshalb nimmt die Wiederholung nicht-fachspezifischer Grundkenntnisse zwangsläufig einen großen Teil des Seminars ein, obwohl die Zeit im Seminar beschränkt ist. Das ist aktuell bei uns ein großes Problem.

Damit wir uns im Seminar auf informatikspezifische Inhalte konzentrieren können statt Grundkenntnisse aufzubauen, sehen wir den Bedarf, die universitäre Lehramtsausbildung für den inklusiven Unterricht im Sinne eines ganzheitlichen Curriculums (d. h. Bildungswissenschaft + Fachdidaktik) noch besser aufeinander abzustimmen. Dann kann sich die DDI verstärkt auf Hilfsmittel und Methoden zu informatikspezifischen Inhalten konzentrieren, wie wir es in Abschnitt 3.1 exemplarisch angedeutet haben. Da nur wenige Studierende die Abschlussart Informatiklehramt wählen, ist die Anzahl der Teilnehmenden zwar für die Evaluation nicht ideal, aber wir erwarten, dass ein solcher Ansatz auch anderen Lehramtsstudierenden helfen kann, sich auf einen zukünftigen inklusiven Informatikunterricht gut vorzubereiten.

Literaturverzeichnis

[AF20] Akao, K.; Fischer, J.: Wie läuft die Umsetzung inklusiven Informatikunterrichts tatsächlich? - Eine Lehrerumfrage zum inklusionsorientierten Unterricht. In (Thomas, M.; Weigend, M., Hrsg.): Mobil mit Informatik. BoD, Norderstedt, S. 9–18, 2020.

[AF21] Akao, K.; Fischer, J.: Zum Stand der Lehramtsausbildung für einen inklusiven Informatikunterricht. In (Humbert, L., Hrsg.): Informatik – Bildung von Lehrkräften in Allen Phasen. LectureNotes in Informatics(LNI)-Proceedings, Bonn, S. 291–294, 2021.

[BD14] Beck, C.; Deutsche Unesco-Kommission: Inklusion: Leitlinien für die Bildungspolitik. Dt. UNESCO-Kommission e.V, Bonn, 3. erw. aufl. Auflage, 2014.

[CG16] Capovilla, D.; Gebhardt, M.: Assistive Technologien für Menschen mit Sehschädigung im inklusiven Unterricht. Zeitschrift für Heilpädagogik, 1(67):4–15, 2016.

[CKH13] Capovilla, D.; Krugel, J.; Hubwieser, P.: Teaching Algorithmic Thinking Using Haptic Models for Visually Impaired Students. In: 2013 Learning and Teaching in Computing and Engineering, LaTiCE 2013, Macau, Macao, March 21-24, 2013. IEEE Computer Society, S. 167–171, 2013.

[Ge08] Gesellschaft für Informatik (GI) e.V.: Grundsätze und Standards für die Informatik in der Schule - Bildungsstandards Informatik für die Sekundarstufe I. LOG IN Verlag GmbH, Berlin, 2008.

[HNR06] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer, Berlin, 2006.

[Ik20] Ikuta, S.; Urushihata, C.; Saotome, N.; Abe, S.: School Activities for Disabled Students Using Self-Made Contents With Multimedia-Enabled Dot Codes. S. 1990–1999, 2020.

[Le11] Leonhardt, A.: Inklusion als fachspezifische Aufgabe der Gehörlosen- und Schwerhörigenpädagogik. Sprache Stimme Gehör, 35:222–223, 2011.

[Ro01] Rose, D.: Universal Design for Learning. Journal of Special Education Technology, 16(2):66–67, 2001.

[SSE01] Schwanke, T. D.; Smith, R. O.; Edyburn, D. L.: A3 Model Diagram Developed as Accessibility and Universal Design Instructional Tool. In: RESNA: Association for the Advancement of Rehabilitation Technology, Arlington,. Jgg. 21, S. 205–207, 2001.

[Th03] The Design-Based Research Collective: Design-Based Research: An Emerging Paradigm for Educational Inquiry. Educational Researcher, 32(1):5–8, 2003.

[Wi20] Wissenschaftsrat: Perspektiven Der Informatik in Deutschland. 2020. www.wissenschaftsrat.de/download/2020/8675-20.pdf.

The Place of Ethics in Computer Science Education

Gregor Große-Bölting[1], Lukas Scheppach[2], Andreas Mühling[3]

Abstract:
Ethical issues surrounding modern computing technologies are playing an increasingly important role in the public debate. Yet, ethics still either doesn't appear at all or only to a very small degree in computer science degree programs. This paper provides an argument for the value of ethics that goes beyond a pure responsibility perspective and describes the positive value of ethical debate for future computer scientists. It also provides a systematic analysis of the module handbooks of 67 German universities and shows that there is indeed a lack of ethics in computer science education. Finally, we present a principled design of a compulsory course for undergraduate students.

Keywords:
Ethics, diversity, social impact, bachelor, curriculum analysis, course development.

1 Introduction

Topics related to computer science that appear in media and thus are made aware to the general public today very often deal with ethics, reflecting how deeply our discipline has affected and shaped our modern world. Such topics, for example, deal with the environmental impact of cryptocurrencies [Ar21], or Non-fungible token (NFTs) [Ba21] or with the rejection of AI powered automatic facial recognition technology by cities in the US, set in motion by the Black Lives Matters movement [CFK19]. Another example are GPT3's capabilities that impress laymen and experts alike, but also engage philosophers to consider the ethical implications [Da21; MN20] and raise questions about the consequences for the human self-image [We20].

The ACM curricula for computer science have included ethics as an elective element since the 1970s and starting in the 1990s even had a (small) mandatory component. Since 2004, it is a required component of degree programs to be accredited [Qu06]. The current 2020 ACM guidelines [Fo20, pp. 76] as well consider ethics to be an important and permanent part of computer science education. As a result, ethics is widely taught as part of computer science in the USA, even though the exact implementation varies widely: Sometimes students have to take ethics courses outside the department, sometimes they are provided within computer science [Re20]. There is also the possibility to spread the compulsory part over several courses and thus to teach ethics in context, e.g. in Machine Learning courses to deal with problems of data bias and in Algorithmics courses to deal with the human part in the design of algorithms [Fi21]. Since the latter model is more difficult to prove that the ethics element has been delivered, it tends to be uncommon. If ethics is integrated within a single course, it is also referred to as a vertical, while if it is spread over several courses, it is referred to as a horizontal ethics offering [Qu06]. Systematic evaluations of the contents of these courses can be found in Saltz [Sa19] and Garrett [GBF20].

The GI as well has recommendations for anchoring ethics in a bachelor's program in computer science: while the 2005 recommendations [In05] consider ethics and responsible action as an aspect for strengthening self-competence in the sub-area of interdisciplinary key competencies, the 2016 recommendations [Zu16] list "Computer Science and Society" as a cognitive competence area. The content addresses legal topics, data protection, professional ethics and social responsibility. Since, as will be shown in the following, this is only part of the ethical and social issues in connection with computer science, this means that a large area of ethical reflection is not taken into account.

1 Kiel University, Department of Computer Science, Christian-Albrechts-Platz 4, 24118 Kiel, Germany ggb@informatik.uni-kiel.de

2 Kiel University, Department of Computer Science, Christian-Albrechts-Platz 4, 24118 Kiel, stu126679@mail.uni-kiel.de

3 Kiel University, Department of Computer Science, Christian-Albrechts-Platz 4, 24118 Kiel, andreas.muehling@informatik.uni-kiel.de

This paper offers a perspective on ethics within computer science courses in Germany in three different ways: First, we are presenting a theoretical argument for the value of ethics in CS. Second, we take a look at the current state of implementation of ethics in German universities and third, we are presenting the design of a new compulsory course on ethics to be implemented at Kiel university.

2 The Value of Ethics in Computer Science Education

Competencies in dealing with ethical implications are increasingly required from computer scientists. For example, conferences in the ML and AI sectors are now beginning to require that papers submitted include an ethical or social impact statement that provides an assessment of the consequences for society of the technical design presented in the paper [AR20]. For novice scientists unfamiliar with and untrained in ethical reasoning, this can be a challenge, in particular as the epistemic approach of our discipline manifested around computational thinking, formal reasoning and problem solving, is inappropriate for solving ethical or social problems, which require negotiation, interpretation, sometimes the toleration of ambiguities. Ethical education can make an important contribution at this point not only by enabling students to navigate their way around ethical considerations in their later professional life, but also by showing the limits of the discipline and making clear when engineers should seek the expertise of experts from other disciplines – such as trained ethicists [RSA21].

In addition, including ethics in CS curricula may also help with the lack of diversity still common in our discipline. One reason why computer science still has a problem with diversity is seen in the fact that women and minorities are particularly interested in realizing communal values through theirs course of study. However, since this is not established in the discipline of computer science and thus lacks appreciation, these groups often find themselves deterred and avoid or leave the field [Di10; Le19]. In addition, it has been observed that a social or ethical orientation decreases in the course of the study of computer science: students who had idealistic goals at the beginning lose them in the course of their studies [GSM20; Pe18]. To put it bluntly, students of computer science lose their ethical and social interest. The predominant epistemic approach for CS – outlined above – has been argued to be one of the driving factors for this [Ea14; RSA21].

Besides this, from the point of view of socio-cultural learning theories, such as Lave and Wenger's Community of Practice, ethics may also help in another way. Newcomers who want to gain a foothold in a discipline do so by learning the practices of that discipline. This includes not only what to do, but how and why [LW+91, p. 98]. Values play a prominent role in this, because a shared and non-explicit set of values must first be experienced and learned in order to become part of the discipline and develop an identity. In this context, ethics can serve as a means of self-assurance and reflection on values and thus make implicit knowledge explicit to newcomers [Lo05].

3 Ethics in German CS Curricula: An Analysis

In order to determine the extent to which ethics plays a role in computer science at universities in Germany, we systematically examined the curricula in this regard.

As the basis of the analysis, we used the most recent CHE ranking of 2020/2021[4]. All universities listed in the ranking and offering bachelor degrees for computer science were selected. Based on this data, the CS curricula and corresponding module catalogs of these institutions were searched for any modules that deal with ethical topics. Courses that are not primarily dealing with ethical topics, but have ethical components, were also included: We applied an inclusive understanding of what we consider ethical content in order not to leave out any eligible courses. The main criterion was whether the course reflects the subject of computer science itself or addresses ethical or social issues in terms of content or methodology. Courses that only include this to a lesser extent (e.g. soft skills) were nevertheless considered. A distinction was made between compulsory and elective modules.

The information available in the module descriptions – in particular the topics and the learning goals – was inductively coded using qualitative content analysis [Ma04] by one of the authors, thus forming a category system (see below). Another author then deductively coded the elective modules based on the existing category system. This approach determined whether further topics were dealt with that were not covered by the categories of the compulsory modules. We also recorded the weekly hours (SWS) of each module.

4 CHE Ranking, accessed May 7, 2021, www.che.de/ranking-deutschland

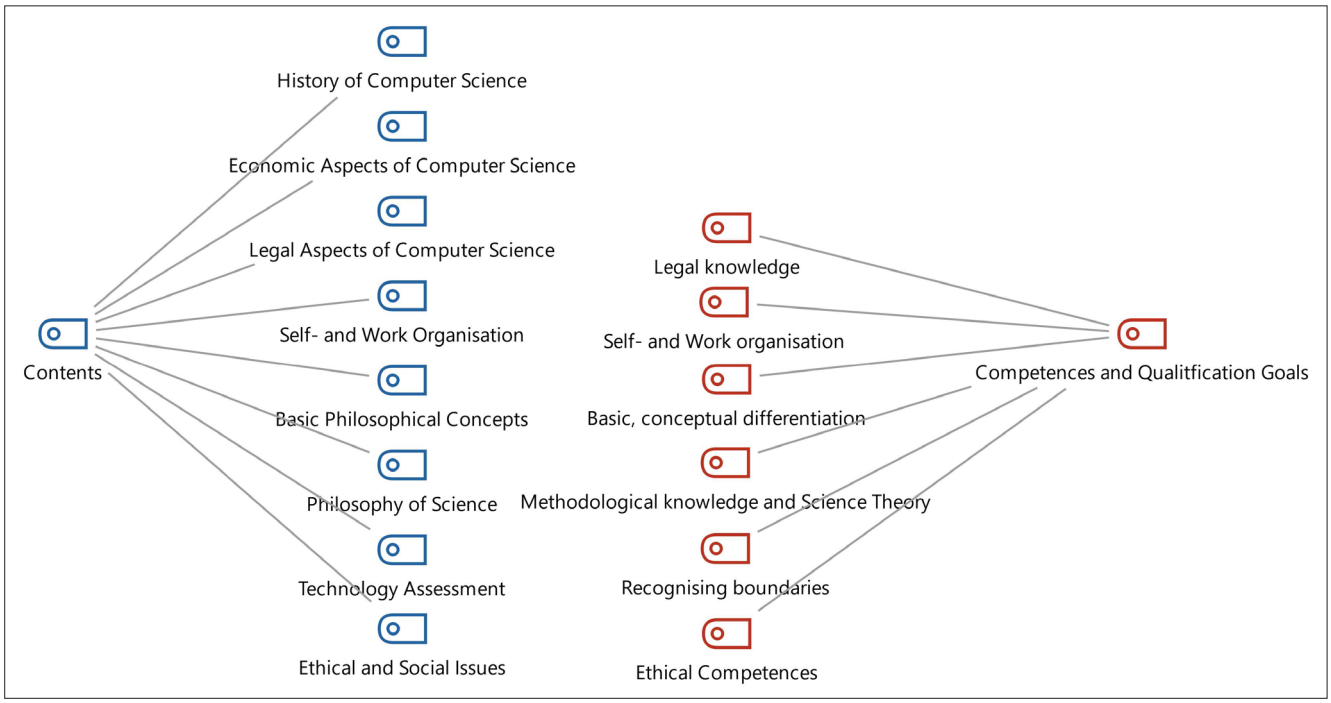

Fig. 1: Matching the contents and competencies (or qualification goals) found in the module descriptions.

3.1 Results

In total, we examined the material of 67 universities. Nine of the examined universities offer compulsory modules and 23 offer elective modules, the remaining institutions do not offer anything related to ethics. The compulsory modules have 3.38 SWS on average (SD 1.41), elective modules have an average of 2.95 SWS (SD 1.43). We were not able to determine the weekly hours for all modules.

The topics of the compulsory modules can be divided into the categories "History of Computer Science", "Economic Aspects of Computer Science", "Legal Aspects of Computer Science", "Self- and Work Organisation", "Basic Philosophical Concepts", "Philosophy of Science", "Technology Assessment" and "Ethical and Social Issues". The top three categories with 21, 16 and 10 appearances respectively are: "Ethical and societal issues", "Legal aspects of Computer Science" and "Self- and Work Organisation".

Regarding the learning goals or competencies that the compulsory modules state in their description, the categories "Ethical competences", "Recognizing boundaries", "Basic, conceptual differentiation", "Methodological knowledge and Science Theory", "Legal knowledge" and "Self- and Work organisation" were identified. "Ethical competences", "Self- and Work Organisation" and "Methodological Knowledge" with 19, 16 and 6 occurrences respectively were the top categories.

The examination of the elective modules did not lead to any new categories regarding the topics or learning goals.

Regarding module names, we found "Computer Science and Society" to be the most frequent, occurring three times in the compulsory modules and eight times in the elective modules. Two electives modules are named "Soft Skills", the remaining module names only appear once. In most module names, such as "Computer Science and Society", "Ethics and Law in Computer Science" or "Technology Assessment and Evaluation", an ethical component can be identified more or less directly. For other module names, such as "Soft Skills", "Key Competences I & II" or "Knowledge in Modern Society", a look at the module description is needed in order to determine that ethical topics in computer science are part of the module.

3.2 Discussion

As the results show there are hardly any modules, neither compulsory nor elective, that are concerned exclusively with ethical issues: Almost all modules deal with other interdisciplinary topics. They often mix ethical with legal issues and even self- and work organization. Considering that the modules are rather small with regard to the weekly hours and that not only ethical issues are dealt with during this time, it can be concluded that ethics currently only plays a subordinate role in computer science degrees in Germany. Even more so, mixing ethical topics, sometimes seemingly random, with other non-core ("soft") topics, fosters a devaluation of ethics as a relevant topic, especially if there is no epistemological clarification and framing [RSA21].

The frequent naming of the courses as "Computer Science and Society", as well as the alignment of the content, suggests a strong orientation of the module authors to the GI recommendations [Zu16]. A limitation of our analysis is the consideration of only module descriptions and manuals: This makes it difficult to gain a precise insight into the content taught. Further research in this area could circumvent this limitation by interviewing lecturers. It is owed to the explorative nature of this study that we initially approached the analysis with an inclusive view of the topic and thus found a wide range of content taught (e.g., soft skills). The category system we developed can be used in the future to conduct further analyses with a more precise focus.

4 A Compulsory Course on Ethics

As part of a restructuring of the Bachelor degree program, the Institute of Computer Science at Kiel University has opted to include ethics as a compulsory part of the curriculum. The concept of the new module is outlined below to serve as a basis for discussion.

The design of the module follows five major design goals: (1.) It deliberately focuses on ethics, not on legal issues, which are often co-covered in the same course context. However, it is important to point out that legality and legitimacy are two very different concepts. Engineers must be knowledgeable about law and the legal framework they operate in but it is ethical awareness that enables meaningful embedding of systems within a social context. (2.) It focuses on systems that are already part of everyday life and on challenges that engineers and scientists face in the here and now. It has been observed that in the context of AI or ML ethics it often happens that the discussion is about things of the distant future, such as speculations on the *Singularity* [Ku06], while the actual already prevailing problems of the present are given less consideration [Co20; Lo19; Mi18]. (3.) It provides students with tools for immediate use in projects. In order for students to experience their engagement with ethics as meaningful, there must be concrete points of connection to practice. Accordingly, problems from computer science should be taught and applicability of the methods should be a focus. (4.) It provides enough ethical knowledge that communication with ethics experts is possible. Regarding the difficult question of the amount of basic philosophical and ethical knowledge that computer scientists need to learn, Gambelin [Ga20] argues that knowledge that enables shared understanding and communication is sufficient. Thus, a shared vocabulary of basic terms and methods must be available without expecting to have in-depth knowledge. (5.) Finally, it strives to make participants aware that ethics is not in the hands of "ethical unicorns" [RSA21] who make decisions for themselves, but that as engineers they have to talk to stakeholders, or indeed to ethicists, when systems they develop affect the interests (of groups) of other people.

From these five design goals, three top-level content areas emerge quite naturally: Ethical foundations must be developed and the location of ethics in the discipline must be addressed. This is done in the first part, *Foundation*. The second part, *Context*, transfers the basic ethical concepts to specific problems that arise in the context of the development of computer science systems in particular. Finally, the third part, *Application*, tries to provide the students with the methodological tools that are necessary for an ethical development of systems.

The foundation part deals with the role of computer science ethics as a domain ethics. It discusses the epistemic means of computer science and how these differ from other sciences in order to narrow down the scope of problem solving. This part also clarifies fundamental lines of ethical reasoning: consequentialism, deontology, and virtue ethics, but also discourse ethics, justice theory, and care ethics. Key concepts – like agency, autonomy, freedom, identity, justice, privacy, responsibility – of computer science ethics, as derived from a comprehensive literature review by Stahl et al. [STM16] are introduced and their meaning is elaborated by the students. Basic techniques of philosophical argumentation are discussed as well, such as the goals and scope of ethical argumentation, thought experiments, etc., and how to read and write ethical texts.

The context part focuses on specific problems in the context of the development and use of computer systems. The individual sections are roughly based on the systematic literature review by Saltz et al. [Sa19] and the curriculum review by Garrett et al. [GBF20] respectively. One of the sections deals with responsibility, in particular with issues around the attribution of responsibility, the responsibility gap, and the problem of many hands that occur when a system acts "autonomously". Other topics include trans-

parency and security of information systems, especially transparency of algorithms and models (in the context of ML) and ethical considerations for the security of information systems, as well as issues around the collection and dissemination of data: The emergence and consequences of systematic biases in data, monopolization, and exploitation in the area of so-called ghost work [GS19]. This section will also discuss professional ethics, the ethical guidelines of the GI, ACM, and IEEE, among others, their scope and criticism (e.g. [Ha20]).

The application part focuses on the implementation of ethical tools and methods. Case studies, e.g. on autonomous weapon systems, self-driving cars and social robots are discussed. A large part of this part is the exploration of ethical design methods and the reflection on how existing design methods such as "agile" contain value concepts. Finally, specific requirements for the computer scientist as a scientist are explained and in particular ethical and social impact statements are elaborated. This should enable the students not only to evaluate the quality of impact statements, but also to write them on their own.

In accordance with the orientation of the module to impart both relevant knowledge from ethics and to specifically train students in practical skills, it will consist to equal parts of a lecture and a seminar. Within the seminar, students will have the opportunity to discuss ethical and social problems, to practice reading and interpreting philosophical texts, and to write their own texts – such as impact statements. In addition, they will have the opportunity to try out ethical design methods by means of exemplary project tasks.

To enable the students to apply their ethical knowledge even more and to ensure a higher level of practical relevance, an ethical component will also be incorporated in the software design project that all students have to do as part of their bachelor's degree. The project is offered in cooperation with a partner from industry and should be attended directly after the ethics module. In the course of the project one day is dedicated to an ethical reflection; the teacher of the ethics module is available to the students for questions and as a discussion partner. Finally, part of the exam performance of the ethics module is that the students develop an ethical concept for their project. Thus, ethics is not only integrated vertically but also horizontally, i.e. across at least two courses in the study program. Students can not only experience the integration of the subject matter into a practical context, but also see that the topic receives recognition and space from other teachers – a problem that often otherwise leads to a devaluation of the topic [Qu06].

5 Conclusion

The Peter Parker principle – "With great power comes great responsibility" – could already be reason enough to deal with ethics more decisively in computer science: Computer systems are part of everyday life and determine many aspects of our lives without us being aware at all times whether a decision is still made by a human or a machine. The fact that the engineers responsible for building and using these systems are nevertheless not subject to any ethical training, that these systems themselves are hardly regulated, should cause astonishment in a democratic society.

Nevertheless, as our analysis shows, ethics is not included as a compulsory part of German CS education in all but nine universities from our sample. Even then, it is often not awarded many weekly hours and grouped together with other "soft" topics in unspecific modules. One limitation of our analysis, however, is that horizontal approaches to teaching ethics are difficult to locate with the method presented above.

Yet, as we discussed, there is an increasing need for ethical trained computer scientists and this may not come naturally to many of them. Including ethics, however, may provide an opportunity for a more diverse group of students to feel welcome in our discipline.

Based on these considerations, a principles-driven proposal for the design of a compulsory ethics course in a Bachelor computer science degree was presented.

Further research would be necessary and helpful to substantiate the relationship between values and disciplinary identity, which is currently assumed in theory but supported only by circumstantial evidence. Overall, it would be desirable to better integrate the influence of reflection on science as part of the degree program and to research its impact. This would include an awareness not only of what computational thinking is, but also of what its limitations are and what problems might be better solved with other tools.

References

[AR20] Abuhamad, G.; Rheault, C.: Like a Researcher Stating Broader Impact For the Very First Time. CoRR abs/2011.13032/, 2020, arXiv: 2011.13032, url: https://arxiv.org/abs/2011.13032.

[Ar21] Aratani, L.: Electricity needed to mine bitcoin is more than used by 'entire countries', en, Feb. 27, 2021, url: www.theguardian.com/technology/2021/feb/27/bitcoin-mining-electricity-use-environmental-impact, visited on: 08/04/2021.

[Ba21] Barber, G.: NFTs Are Hot. So Is Their Effect on the Earth's Climate, en-US, June 3, 2021, url: https://www.wired.com/story/nfts-hot-effect-earthclimate/, visited on: 08/04/2021.

[CFK19] Conger, K.; Fausset, R.; Kovaleski, S. F.: San Francisco Bans Facial Recognition Technology, en-US, May 14, 2019, url: https://www.nytimes.com/2019/ 05/14/us/facial- recognition- ban- san- francisco.html, visited on: 08/04/2021.

[Co20] Coeckelbergh, M.: AI Ethics. The MIT Press, 2020.

[Da21] Dale, R.: GPT-3: What's it good for? Natural Language Engineering 27/1, pp. 113–118, 2021.

[Di10] Diekman, A. B.; Brown, E. R.; Johnston, A. M.; Clark, E. K.: Seeking Congruity Between Goals and Roles: A New Look at Why Women Opt Out of Science, Technology, Engineering, and Mathematics Careers. Psychological Science 21/8, PMID: 20631322, pp. 1051–1057, 2010.

[Ea14] Easterbrook, S.: From Computational Thinking to Systems Thinking: A conceptual toolkit for sustainability computing. In: Proceedings of the 2014 conference ICT for Sustainability. Atlantis Press, pp. 235–244, 2014.

[Fi21] Fiesler, C.; Friske, M.; Garrett, N.; Muzny, F.; Smith, J. J.; Zietz, J.: Integrating Ethics into Introductory Programming Classes. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education. SIGCSE '21, Association for Computing Machinery, Virtual Event, USA, pp. 1027–1033, 2021.

[Fo20] Force, C. T.: Computing Curricula 2020: Paradigms for Global Computing Education. Association for Computing Machinery, New York, NY, USA, 2020, isbn: 9781450390590.

[Ga20] Gambelin, O.: Brave: what it means to be an AI Ethicist. AI and Ethics/, 2020, issn: 2730-5961, url: https://doi.org/10.1007/s43681-020-00020-5.

[GBF20] Garrett, N.; Beard, N.; Fiesler, C.: More Than "If Time Allows": The Role of Ethics in AI Education. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. AIES '20, Association for Computing Machinery, New York, NY, USA, pp. 272–278, 2020.

[GS19] Gray, M. L.; Suri, S.: Ghost Work: How to Stop Silicon Valley from Building a New Global Underclass. Mariner Books, 2019.

[GSM20] Große-Bölting, G.; Schneider, Y.; Mühling, A.: Beginning Students' Conceptions of Computer Science: The Effect of the First Semester. In: 2020 International Conference on Learning and Teaching in Computing and Engineering (LaTICE). Ho-Chi-Minh-City; Vietnam, 2020.

[Ha20] Hagendorff, T.: The Ethics of AI Ethics: An Evaluation of Guidelines. en, Minds and Machines 30/1, pp. 99–120, Mar. 2020, issn: 1572-8641.

[In05] Informatik, G. f.: Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Dezember 2005), 2005.

[Ku06] Kurzweil, R.: The Singularity Is Near: When Humans Transcend Biology. Penguin (Non-Classics), London, 2006, isbn: 0143037889.

[Le19] Lewis, C.; Bruno, P.; Raygoza, J.; Wang, J.: Alignment of Goals and Perceptions of Computing Predicts Students' Sense of Belonging in Computing. In: Proceedings of the 2019 ACM Conference on International Computing Education Research. ICER '19, Association for Computing Machinery, Toronto ON, Canada, pp. 11–19, 2019.

[Lo05] Loui, M. C.: Ethics and the Development of Professional Identities of Engineering Students. Journal of Engineering Education 94/4, pp. 383–390, 2005.

[Lo19] Loh, J.: Roboterethik: Eine Einführung. Suhrkamp Verlag, Frankfurt am Main, 2019, isbn: 978-3-518-29877-0.

[LW+91] Lave, J.; Wenger, E., et al.: Situated learning: Legitimate peripheral participation. Cambridge university press, 1991.

[Ma04] Mayring, P.: Qualitative content analysis. A companion to qualitative research 1/2, pp. 159–176, 2004.

[Mi18] Misselhorn, C.: Grundfragen der Maschinenethik. Reclam, Philipp, jun. GmbH, Verlag, Stuttgart, 2018, isbn: 978-3-15-019583-3.

[MN20] McGuffie, K.; Newhouse, A.: The Radicalization Risks of GPT-3 and Advanced Neural Language Models./, 2020, arXiv: 2009.06807 [cs.CY].

[Pe18] Peters, A.-K.: Students' Experience of Participation in a Discipline – A Longitudinal Study of Computer Science and IT Engineering Students. ACM Trans. Comput. Educ. 19/1, Sept. 2018.

[Qu06] Quinn, M. J.: On teaching computer ethics within a computer science department. Science and Engineering Ethics 12/2, pp. 335–343, 2006.

[Re20] Reich, R.; Sahami, M.; Weinstein, J. M.; Cohen, H.: Teaching Computer Ethics: A Deeply Multidisciplinary Approach. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. SIGCSE '20, Association for Computing Machinery, Portland, OR, USA, pp. 296–302, 2020.

[RSA21] Raji, I. D.; Scheuerman, M. K.; Amironesei, R.: You Can't Sit With Us: Exclusionary Pedagogy in AI Ethics Education. In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. FAccT '21, Association for Computing Machinery, Virtual Event, Canada, pp. 515–525, 2021.

[Sa19] Saltz, J.; Skirpan, M.; Fiesler, C.; Gorelick, M.; Yeh, T.; Heckman, R.; Dewar, N.; Beard, N.: Integrating Ethics within Machine Learning Courses. ACM Trans. Comput. Educ. 19/4, Aug. 2019.

[STM16] Stahl, B. C.; Timmermans, J.; Mittelstadt, B. D.: The Ethics of Computing: A Survey of the Computing-Oriented Literature. ACM Computing Surveys 48/4, 55:1–55:38, Feb. 2016.

[We20] Weinberg, J.: Philosophers On GPT-3 (updated with replies by GPT-3). Daily-Nous.com/, 2020.

[Zu16] Zukunft, O.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016), 2016.

Aufbau informatischer Kompetenzen im Kontext KI bei Lehramtsstudierenden des Faches Politik

David Baberowski[1], Thiemo Leonhardt[1], Gregor Damnik[1], Susanne Rentsch[1], Nadine Bergner[1]

Abstract:
Informatische Bildung stellt nicht nur in allen Schularten und Fächern (und damit in der Lehrkräftebildung) eine Chance dar, um die wachsende Alltagsrelevanz von Informatik zu adressieren, sondern ist bereits heute unabdingbar, um aktuell gültigen Lehrplänen gerecht zu werden. So verweist beispielsweise der Lehrplan für Gemeinschaftskunde, Rechtserziehung und Wirtschaft in Sachsen für das Gymnasium der Jahrgangsstufe 11 mit dem Thema „Digitalisierung und sozialer Wandel" auf Künstliche Intelligenz (KI) und explizit auf die Bedeutung der informatischen Bildung. Um die nötigen informatischen Grundlagen zu vermitteln, wurde für Lehramtsstudierende des Faches Politik ein Workshop[2] erarbeitet, der die Grundlagen der Funktionsweise von KI anhand von überwachtem maschinellen Lernen in neuronalen Netzen vermittelt. Inhalt des Workshops ist es, mit Bezug auf gesellschaftliche Implikationen wie Datenschutz bei Trainingsdaten und algorithmic bias einen informierten Diskurs zu politischen Themen zu ermöglichen. Ziele des Workshops für Lehramtsstudierende mit dem Fach Politik sind: (1) Aufbau informatischer Kompetenzen in Bezug zum Thema KI, (2) Stärkung der Diskussionsfähigkeiten der Studierenden durch passende informatischen Kompetenzen und (3) Anregung der Studierenden zum Transfer auf passende Themenstellungen im Politikunterricht. Die Analyse der Diskussion zeigte das Bewusstsein der Alltagsrelevanz des Themas KI bei den Teilnehmenden, aber noch keine Anwendung der informatischen Inhalte des Workshops zur Stützung der Argumente in der Diskussion.

Keywords:
informatische Grundkompetenzen; Lehramtsstudium; KI; maschinelles Lernen; fächerverbindend

1 Einleitung

Informatische Bildung ist in verschiedenen fachdidaktischen Disziplinen außerhalb der Informatik bereits Gegenstand unterrichtlicher Praxis, wenn auch in sehr unterschiedlicher Ausprägung [Ba20]. Neben der Ausbildung der fachdidaktischen und medienpädagogischen Kompetenzen aller Lehramtsstudierenden, sind informatische Grundkompetenzen (IGK) nötig,umaktuelle Entwicklungen der fachdidaktischen Inhalte imKontext der Digitalisierung in allen Fächern diskutieren und analysieren zu können [Ga17] [Hu20].

Eine Möglichkeit informatische Bildung im Lehramtsstudium aller Schularten und Fächer zu verankern, sind allgemeinbildende Informatikveranstaltungen [LH19], welche im Bezug auf informatische Inhalte breit aufgestellt werden können und dies aufgrund der fächerübergreifenden Zielgruppe auch tun müssen. Der alternative, hier verfolgte Ansatz, ist ein Workshop Angebot, welches in Kooperation mit der entsprechenden Fachdidaktik durchgeführt wird. Durch die Integration in Fachveranstaltungen werden passgenaue Inhalte geboten und für das Fach relevantere Aspekte betrachtet, als dies in allgemeinbildenden Veranstaltungen möglich ist.

Im Fall des Lehramtsfachs Gemeinschaftskunde lassen sich über das Themenfeld Künstliche Intelligenz aufgrund der gesellschaftlichen Auswirkungen KI-basierter Informatiksysteme sinnvolle Berührungspunkte zwischen politischen und informatischen Kompetenzen konstruieren. Algorithmische Systeme, die in der Alltagswelt bereits an vielen Stellen gesellschaftliche Kommunikations- und Partizipationsprozesse beeinflussen, werfen für die Politikdidaktik zentrale Fragen auf, die nur auf Basis informatischer Grundkompetenzen reflektiert und diskutiert werden können: Wie wirken sich algorithmisch verstärkte Filterblasen und Echokammern auf Multiperspektivität und demokratischen Zusammenhalt aus? Welche Strategien zum Umgang mit medialen Informati-

1 TU Dresden, Didaktik der Informatik, Nöthnitzer Straße 46, 01187 Dresden, Deutschland
{david.baberowski | thiemo.leonhardt | gregor.damnik | susanne.rentsch1 | nadine.bergner} @tu-dresden.de

2 Das diesem Artikel zugrundeliegende Vorhaben wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung" von Bund und Ländern mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01JA2017B gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

onsangeboten werden im Kontext von Deep Fakes und Automated Journalism benötigt? Welche Auswirkungen haben Fälle von algorithmic bias auf Chancengleichheit und soziale Gerechtigkeit? Mit Blick auf die aktuelle und absehbar steigende gesellschaftliche Relevanz des Themas wird Künstliche Intelligenz im Rahmen der fachdidaktischen Ausbildung an der TU Dresden bereits als potenzielles inhaltliches Aufgabenfeld für die politischen Bildung identifiziert. Im sächsischen Lehrplan für das Unterrichtsfach Gemeinschaftskunde, Rechtserziehung, Wirtschaft an Gymnasien wird auf Künstliche Intelligenz konkret im Kontext des Lernbereichs „Digitalisierung und sozialer Wandel" für den Grundkurs der Jahrgangsstufe 11 hingewiesen. Vor diesem Hintergrund wird das Thema als geeignete Grundlage zur Entwicklung eines passgenauen Workshop-Angebots für die Lehramtsstudierenden im Fachbereich Gemeinschaftskunde ausgewählt.

Eine Herausforderung in derKonzeption eines entsprechenden Angebots ist die Identifikation der passenden informatischen Inhalte, die eine Integration des Themas in die Fachdidaktik der politischen Bildung ermöglichen. Des Weiteren ist die Auswahl eines geeigneten didaktischen Ansatzes zu diskutieren, um das Thema KI im Kontext der Anforderungen der politischen Bildung zu bearbeiten. Ausgehend von dieser Analyse ist in Zusammenarbeit mit der Fachdidaktik für politische Bildung an der TU Dresden ein Workshop zum Thema „KI und Politik" entwickelt worden, der als ein Seminartermin in die Veranstaltung „Aktuelle Tendenzen" im 9. Semester des Studiengangs Lehramt Gemeinschaftskunde integriert wurde. Die inhaltliche und didaktische Konzeption, die Materialien des Workshops sowie die aktuellen Auswertungsdaten der Pilotierung wurden im Open Science Framework registriert und können dort eingesehen werden.[3]

2 Fachdidaktische Begründung

Die Integration informatischer Ziele und Inhalte in einen politikdidaktischen Seminarrahmen setzt eine Einordnung politikdidaktischer Zielvorstellungen in den Kontext informatischer Kompetenzen voraus.Wie die Informatik selbst, ist auch die politische Bildung im Kern multiperspektivisch und interdisziplinär angelegt [Sc14, S. 229–230]. Im Fokus der politischen Bildung steht die Frage, wie junge Menschen in der Entwicklung politischer Mündigkeit unterstützt werden können. Mündigkeit meint hier die Fähigkeit, sich selbstbestimmt und eigenverantwortlich in einer Gesellschaft zu orientieren, politische Fragestellungen kompetent zu beurteilen und sich in öffentlichen Angelegenheiten zu engagieren [De04, S. 9]. Diese Ziele lassen sich in einer Informationsgesellschaft, die vom stetig wachsenden Einfluss der Digitalisierung geprägt ist, ohne den Erwerb informatischer Grundkompetenzen nicht verwirklichen. Um die Entwicklung von Mündigkeit und Selbstbestimmtheit zu fördern, strebt die politische Bildung insbesondere Kompetenzentwicklungen in den folgenden drei Bereichen an: Politische Urteilsfähigkeit, politische Handlungsfähigkeit und methodische Fähigkeiten [De04, S. 13]. Reflektierte Urteile zu den gesellschaftlichen Auswirkungen von Informatiksystemen zu entwickeln oder gesellschaftliche Partizipation und selbstbestimmtes Informieren im Kontext von Digitalisierungsprozessen einzuüben, setzen dabei unerlässlich immer auch grundlegende informatische Kompetenzen voraus.

Deshalb wurden für den Workshop (1) der Aufbau informatischer Kompetenzen im Kontext KI, (2) die Stärkung der Diskussionsfähigkeiten der Studierenden durch passende informatische Kompetenzen und (3) die Anregung der Studierenden zum Transfer auf passende Themenstellungen im Politikunterricht, als Ziele gesetzt. Die didaktische Betrachtung geschieht nach dem Ansatz der systemorientierten Didaktik [Ma03]. Informatiksysteme mit KI-Komponenten werden in dem hier betrachteten Fall des überwachten maschinellen Lernens in neuronalen Netzen als soziotechnische Informatiksysteme betrachtet und analysiert. Ziel ist die Verzahnung von technischer Entwicklung und gesellschaftlichen Veränderungen im Kontext dieses Informatiksystems begreifbar zu machen [Sc03, S. 39].

2.1 Systemorientierter Ansatz

Die Lehramtsstudierenden der Fachdidaktik der politischen Bildung sollen sich in das Thema KI so einarbeiten, dass sie den Prozess der Entwicklung von KI-Systemen durch maschinelles Lernen nachvollziehen und in ersten Schritten selbst durchführen können. Dadurch soll es den Studierenden möglich werden, die Erstellung von KI-Systemen als sozialverträgliche Technikgestaltung in einem interessengeleiteten Entscheidungsprozess zu erleben. Die Erkenntnis, eine zuvor nicht durchschaubaren Blackbox durch aktive Schritte selbst gestalten zu können, soll einen differenzierteren Blick auf das Thema KI bei den Studierenden erzeugen. Die anschließende Übertragung der erlernten informatischen Kompetenzen in den eigenen Politikunterricht ist das langfristige Ziel.

3 Die Projektdaten sind unter der Lizenz CC-By Attribution 4.0 International veröffentlicht. DOI 10.17605/OSF.IO/HR2ZM. Zugang zu den Projektdaten https://osf.io/hr2zm/.

Dafür wird nach dem Ansatz der systemorientierten Didaktik eine Dekonstruktion eines KI-Systems vorgenommen, um die Grundlage für eine Handlungskompetenz im Umgang mit diesem zu legen, die anschließend zur Vermittlung der Kompetenzen der politischen Bildung im eigenen Unterricht genutzt wird. Künstliche Neuronale Netzwerk im Kontext von maschinellem Lernen werden als soziotechnische Informatiksysteme begriffen, die sich aus den Komponenten Hardware, Software und Interessensgruppen zusammensetzt. Die Verzahnung von technischen und sozialen Aspekten ist dabei ein wesentliches Ziel. Die Dekonstruktion ist zwar themenabhängig nicht unbedingt für den Anfangsunterricht geeignet [SB02], bietet aber durch die Multiperspektivität einen vielversprechenden Ansatz zur Ausbildung der Kompetenzen der politischen Bildung. Die Schwierigkeit der Konzeption besteht demnach in der passenden fachlichen Tiefe der zu vermittelnden Komponenten aus der Dekonstruktion. Der hier entwickelte Ansatz setzt sich aus mehreren Komponenten zusammen (vgl. Tabelle 1), die im Folgenden näher erläutert werden.

Tab. 1: Neuronales Netz als soziotechnisches Informatiksystem

Neuronales Netz	
Hardware	Eingabegeräte, Prozessor, Ausgabegeräte
Software	Neuronales Netz, Trainingsdaten, Trainingsprozess/Trainingsfunktion
Interessensgruppen	Anwender:innen, Entwickler:innen

2.2 Komponenten der Dekonstruktion

Um die Besonderheiten von maschinellem Lernen gegenüber klassischem Implementieren von Algorithmen in Formvon Quellcode korrekt einordnen zu können, werden beide Ansätze gegeneinander abgegrenzt und im Folgenden als *trainierte* bzw. **formulierte** Software und Algorithmen bezeichnet. Während bei maschinellem Lernen das automatisierte generieren von Algorithmen *(Training)* im Mittelpunkt desWorkshops steht, wird sich beim klassischen Implementieren auf das Beschreiben von Handlungsanweisungen *(Formulieren)* fokussiert.

Maschinelles Lernen in neuronalen Netzen wird anhand von Softwarebeispielen in einen sozialen Kontext gesetzt, indem Datenschutz bei Trainingsdaten und algorithmic bias als direkte Auswirkungen auf das gesellschaftliche Leben thematisiert werden. Die Betrachtung des Informatiksystems aus der Perspektive Hardware erfolgt durch eine Reduzierung auf das EVA-Prinzip.

Aus der Perspektive Software wird die Entstehungsweise von Verarbeitungsregeln in Form von neuronalen Netzen betrachtet. Diese wird weiter zerlegt, um die Funktionsweise der in Schichten angeordneten Neuronen verständlich zumachen. Zusätzlich wird der Trainingsprozess von der Anwendung eines neuronalen Netzes getrennt betrachtet und von den Teilnehmenden erprobt. Das Unterteilen von Informatiksystemen und Algorithmen in die Bestandteile Eingabe, Verarbeitung und Ausgabe basiert auf den Empfehlungen für Bildungsstandards der GI [Br08] und dem Inhaltsbereich „Informatiksysteme“. Übertragen auf die Perspektive Software werden Gemeinsamkeiten *trainierter* und *formulierter* Algorithmen herausgestellt. Der Kontext, in den das neue Wissen eingebettet wird, hilft Erfahrungen und Konzepte aus dem Alltag einzuordnen. Das ist insbesondere deshalb sinnvoll, da trainierte Algorithmen (z.B. zur Gesichtserkennung und Segmentierung) häufig in Verbindung mit klassisch programmierten Anwendungen (z.B. Instagram) auftreten.

Durch die Einordnung von KI als disruptive Technologie kann schnell der Eindruck entstehen, dass *trainierte* und *formulierte* Software grundverschieden sind, obwohl es viele verbindende Aspekte gibt.Uman klassisch programmierte Software aus der Alltagserfahrung anknüpfen zu können, wird die Formulierung eindeutiger Verarbeitungsregeln betrachtet. Diese Regeln werden ohne eine Syntax in natürlicher Sprache vorgestellt, um den Einstieg zu erleichtern. Den Teilnehmenden werden die Schwierigkeiten bei der Entwicklung von generalisierten Lösungsverfahren demonstriert. So soll ein Verständnis dafür erlangt werden, dass dieses Vorgehen besonders dann an seine Grenzen stößt, wenn eine Lösungsstrategie im Allgemeinen auf menschlicher Intuition beruht. Dabei soll die Fehlvorstellung vermieden werden, diese Schwierigkeiten beziehen sich auf die Ausführung der Verarbeitungsregeln. Stattdessen wird herausgestellt, dass die Grenzen von *formulierten* Algorithmen in dem Entwurf dieser Regeln liegen.

Da künstliche neuronale Netze, bestehend aus verknüpften Neuronen, ein in Software implementiertes Modell sind, werden diese hier als Softwarekomponente der Dekonstruktion des Informatiksystems eingeordnet. Die üblicherweise in Schichten angeordneten Neuronen entscheiden aufgrund ihrer Verknüpfungen, ob ein Signal weitergeben wird. Dieses Verhalten kann für jedes Neuron angepasst werden. Der Vorteil dieser Struktur liegt in der automatisierbaren Manipulation des Verhaltens der Neuronen, womit der Übergang zum Trainingsprozess gegeben ist. Der Trainingsprozess tritt an die Stelle der klassischen Programmierung und ermöglicht es, Algorithmen mit Hilfe von Trainingsdaten zu finden, statt sie manuell zu formulieren. Dieser Prozess wird auf die drei Komponenten Trainingsdaten, neuronales Netz und Trainingsfunktion reduziert (siehe Abb. 1). Die Trainingsdaten sind dabei Beispieldaten, für die das Problem bereits gelöst wurde (z. B. ein Bild korrekt erkannt wurde). Um komplexe Vorgänge wie Backpro-

pagation zur Identifizierung der anzupassenden Neuronen nicht im Seminar thematisieren zu müssen, wird die Trainingsfunktion als Blackbox-Komponente in den Trainingsprozess eingeführt.

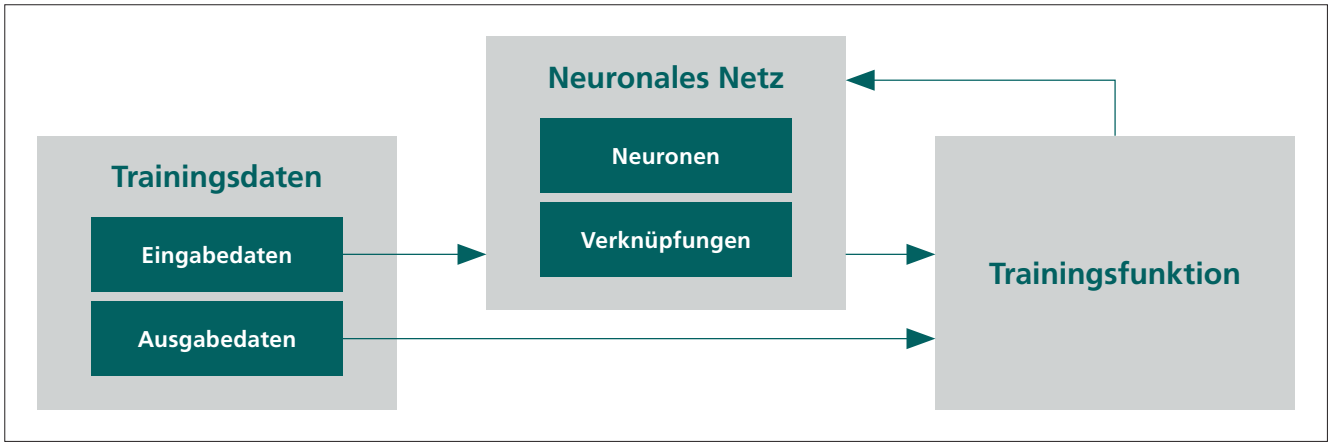

Abb. 1: Bestandteile des Trainingsprozess.

Der Trainingsprozess beginnt mit einem untrainierten Netz, in dem die Verknüpfungen und Neuronen mit meist zufälligen Startwerten angeordnet werden. Anschließend werden Trainingsdaten als Eingabedaten verwendet, um eine Ausgabe zu erzeugen. Die Trainingsfunktion vergleicht anschließend die Ausgabe des Netzes mit den Trainingsdaten und passt das Verhalten der Neuronen und Verknüpfungen im neuronalen Netz an. Anschließend wird der Vorgang mit weiteren Trainingsdaten wiederholt und das Verhalten der Neuronen wiederum angepasst, wodurch sich die Ausgabe des Netzes iterativ den Vorgaben der Trainingsdaten annähert. Anschließend wird das Training beendet. Die Verknüpfungen und das Verhalten der einzelnen Neuronen werden ab jetzt nicht mehr geändert. Stattdessen kann das trainierte neuronale Netz verwendet werden, um das Problem für neue Eingabedaten zu lösen. Das neuronale Netz repräsentiert nun einen Algorithmus, der in diesem Prozess gefunden wurden.

In diesem Kontext soll auch über *algorithmic bias* gesprochen werden. Im Workshop wird ein Algorithmus als *biased* bezeichnet, wenn er für bestimmte Eingabedaten von einem neutralen Verhalten abweicht, oder bestimmte Ausgabedaten überproportional produziert. Wann *algorithmic bias* vorliegt und was ein erwünschtes neutrale Verhalten des Systems wäre sind spannende kontextabhängige Fragestellungen für eine gesellschaftlich politische Diskussion. *Algorithmic bias* ist meist unintendiert und kann verschiedene Ursachen haben. Der Workshop behandelt den *training data bias* (vgl. [DL17]).

3 Workshop-Konzept

Das Workshop-Konzept soll es den Teilnehmenden ermöglichen im Unterricht Bezug auf das Thema KI und ML-Algorithmen zu nehmen. Die Vermittlung der ML Grundprinzipien profitiert besonders von der Behandlung der Arbeitsschritte des Datenlabelings, sowie der Evaluation und Anwendung des trainierten Modells (vgl. [Hi19]). Da in politischen Diskussionen oft die Frage diskutiert wird, welche Arten von Algorithmen bestimmte Aufgaben übernehmen sollten,werden Chancen und Risiken von *formulierten* und maschinell *trainierten* Algorithmen gegenübergestellt.

Der Workshop wurde in ein virtuell durchgeführtes Seminar der Didaktik der Politik als ein regulärer, 90-minütiger Seminartermin integriert und besteht aus einer Vorbereitungphase, der Workshopdurchführung und einer Nachbereitungsphase (siehe Abb. 2). Die Vorbereitungphase des Workshops besteht aus einer Vorbefragung zur Zuversicht zur Vermittlungskompetenz und einer anschließenden Selbstlernphase zum EVA-Prinzip. Dazu wird den Teilnehmenden ein Lernvideo zugänglich gemacht, in welchem das EVA-Prinzip anhand von Alltagsbeispielen thematisiert wird. Darauf aufbauend sollen die Komponenten der KI-Webanwendungen CNNMRF Api[4] und TalkingHeads[5] von den Teilnehmenden den Bestandteilen Eingabe, Verarbeitung und Ausgabe zugeordnet werden. Die Anwendungen wurden ausgewählt, da hier direkt eine politische Relevanz (DeepFakes) zu erkennen ist.

4 https://deepai.org/machine-learning-model/CNNMRF

5 https://talkingheads.rosebud.ai

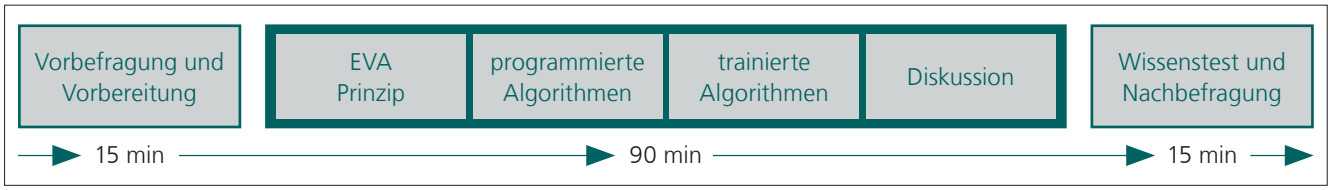

Abb. 2: Ablauf der Phasen des Workshops

Die Inhalte der Selbstlernphase in der Vorbereitung werden zu Beginn des Workshops in Form einer Live-Umfrage mit anschließender Diskussion aufgegriffen. Dies dient der kurzen Wiederholung der Inhalte aus der Vorbereitungsphase, um Verständnisprobleme oder Versäumnisse der Bearbeitung direkt per anonymen Live-Feedback aufzugreifen. Ein aktives Einbinden der Teilnehmenden sollte dazu beitragen, eine offene Atmosphäre im Distanzlernen zu erzielen und gleichzeitig die wichtigsten Begriffe zu aktivieren. Als zweite Komponente werden die Grundlagen von programmierten Algorithmen erarbeitet. Die Teilnehmenden werden in Dreiergruppen eingeteilt und erhalten die Aufgabe, in eigenen Videokonferenzräumen für verschiedene grafische Formen eindeutige Handlungsanweisungen zu formulieren, um diese zu zeichnen. Sind die Handlungsanweisungen formuliert, wird die Gruppenarbeit beendet. Die formulierten Anweisungen werden im nächsten Schritt mit den anderen Gruppen über ein virtuelles Whiteboard ausprobiert. Dazu führt jeweils eine Gruppe die Anweisungen der anderen Gruppe aus. Gemeinsam wird anschließend die ursprüngliche Form mit der gezeichneten verglichen (vgl. Abb. 3 links). Das Ziel dieser Aufgabe ist es, zu vermitteln, wie schwer es ist, eindeutige Regeln für eine automatisierte Ausführung zu formulieren.

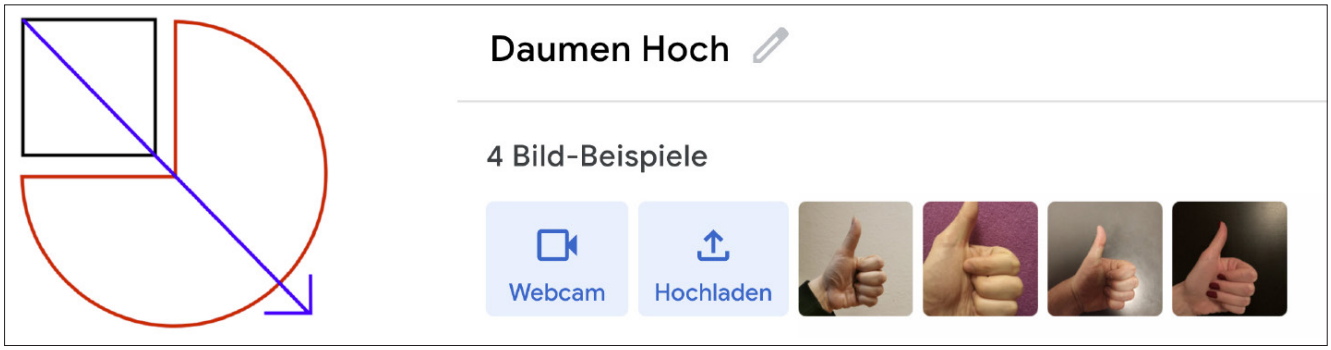

Abb. 3: links: vorgegebene Form, rechts: Beispieldaten der Handgestenerkennung

Am Beispiel der Erkennung von Ziffern in Bildern werden die Grenzen von *formulierten* Algorithmen plausibel gemacht, indem zunächst über Lösungsstrategien zur Klassifizierung gesprochen wird. Anschließend wird das Problem auf handschriftliche Ziffern erweitert, wodurch klar wird, dass ein *formulierter* Ansatz viele verschachtelte Ausnahmen vorsehen müsste, um die Aufgabe zu lösen. Im Gegensatz dazu wird der Trainingsprozess eines neuronalen Netzes durch maschinelles Lernen Schritt für Schritt nachvollzogen und als alternative Lösung für diese Problemklasse vorgestellt. Begriffe wie Neuronen und Trainingsdaten werden jeweils an den Stellen erklärt, an denen sie zum ersten Mal auftreten und auf dem Niveau eingeführt, das in der didaktischen Reduktion begründet wurde.

Für die praktische Demonstration des Trainingsprozesses wird das webbasierten Tools Teachable Machine[6] verwendet. Dieses wurde aufgrund der intuitiven Benutzung und des visuellen Ansatzes ausgewählt, der Assoziationen zu alltagsbezogenen Einsatzszenarien von neuronalen Netzen ermöglicht, um dadurch einen Transfer in den Politikunterricht zu vereinfachen. Der berechtigen Kritik der fehlenden ganzheitlichen Betrachtung des Deep Learning Workflows bei Teachable Machine [SGB21], trägt das Workshopkonzept Rechnung. Erstens muss aufgrund der Integration in den Workshop das Tool keinen vollständigen Prozess abbilden, da die fehlenden Prozessschritte durch andere Workshopelemente abgebildet werden können. Zweitens ist eine vollständige, aber rein technische Betrachtung des Prozesses aufgrund der Zielgruppe der Lehramtsstudierenden der politischen Bildung und der Zielsetzung nicht zielführend. Das Tool Teachable Machine macht wesentliche Bestandteile eines neuronalen Netzwerks in einer reduzierten Oberfläche zugänglich und manipulierbar. In der Anwendung des Tools werden die Teilnehmenden mit dem Userinterface und den grundlegenden Funktionen der Webanwendung vertraut gemacht. Nach dieser Einführung beginnt das selbstständige Trainieren eines neuronalen Netzes zur Erkennung von Handgesten (vgl. Abb. 3 rechts). Die Aufgabe ist so strukturiert, dass ein *algorithmic bias* zunächst sehr wahrscheinlich ist. Dies wird erreicht, indem mit einem vorgegebenen Trainingsset die Posen Daumen-hoch und OK besser vertreten sind als die Pose Peace. Erst durch eine Erweiterung der Trainingsdaten, z.B. durch das Hinzufügen weiterer Peace-Posen über die Webcam, wird eine falsche Klassifizierung vermieden.

6 https://teachablemachine.withgoogle.com

Da die Teilnehmenden ihre Netze untereinander austauschen sollen, muss ein markantes Feature im Mittelpunkt der Bilder der Webcams stehen, damit die Trainingsergebnisse auch mit der Bandbreite an verschiedenen Hintergründen im Homeoffice-Setting umgehen können. Die Aufgabe mit den Handgesten erfüllt diese Anforderung zuverlässig. In der Nachbesprechung der Aufgabe werden beobachtete Probleme auf das Beispiel der Filterung von Bildern in Sozialen Netzen übertragen. Dadurch wird das System einer Bildklassifikation im Sinne des Systemorientierten Ansatzes aus den Perspektiven der Interessensgruppen der Entwickler:innen und Anwender:innen betrachtet.

Den Abschluss des Workshops bildet eine offene Diskussion aller Teilnehmenden untereinander. Eingeleitet wird die Diskussion mit der Fragestellung der Alltagsrelevanz von KI, sie geht dann über zu der zu diskutierenden Kernfragestellung, welche Rolle das Thema im Politikunterricht einnehmen sollte.

4 Analyse der Ergebnisse

Als zentrales Instrument der Evaluation wird eine Analyse der Gruppendiskussion genutzt, in der die Teilnehmenden informatische Kompetenzen bei der Beurteilung der gesellschaftlichen Auswirkungen von Informatiksystemen mit KI heranziehen sollen. Damit soll ein Transfer auf Themenstellungen im Politikunterricht nachvollziehbar werden, der sich zum Beispiel durch neue, im Workshop nicht behandelte Themen zeigen kann. Die Gruppendiskussion wird mit dem Einverständnis der Teilnehmenden während der Durchführung aufgezeichnet und anschließend transkribiert.

In der Diskussion wird die Anwendung der erlernten informatischen Inhalte als Argumente in der Gruppendiskussion im Kontext der politischen Bildung evaluiert. In der Diskussion wird analysiert, ob die Teilnehmenden 1.) Stellen korrekt nennen, an denen bereits heute schon KI eingesetzt wird, 2.) fachlich nachvollziehbar begründen, weshalb das Thema KI für das Ziel der politischen Urteilsfähigkeit relevant ist und 3.) erläutern, welche informatischen Aspekte des Themas KI im Politikunterricht relevant sind. Um die Teilnehmenden zu einer besseren Mitarbeit zu motivieren, werden die letzten beiden Fragestellungen der Diskussion in ein Szenario zur Bearbeitung des Lehrplans im Fach Politik eingebettet. Die konkreten Fragestellungen für die Diskussion lauten:

- Nenne Szenarien, in denen bereits heute KI eingesetzt wird oder Anwendungen bei denen du dir in Zukunft einen Einsatz von KI vorstellen kannst.
- Warum ist das Thema KI für das Ziel der politischen Urteilsfähigkeit relevant?
- Welche Aspekte sollten in den Unterricht integriert werden?

Eine erste Pilotierung fand am 18.01.2021 mit acht Teilnehmenden statt. In der Auswertung der Pilotierung wurden in der Gruppendiskussion als aktuelle und zukünftige Einsatzgebiete von KI Verkehrsleitsysteme, autonomes Fahren und der Einsatz im Gesundheitssystem z.B. bei der Unterstützung von Diagnostik genannt. Keines der von den Teilnehmenden genannten Einsatzgebiete wurde imWorkshop als Beispiel präsentiert. Die Relevanz für den Politikunterricht wurde nicht durch informatische Argumente belegt, sondern mit Beispielen für Schnittmengen von informatischer und politischer Bildung im Kontext KI, wodurch sich diese nicht vom dritten Teil der Diskussion (Aspekte, die sich für eine Integration in den Politikunterricht eignen) trennen lässt. Genannte Aspekte waren Deepfakes und FakeNews, Filterblasen in sozialen Netzwerken und Social-Creditsysteme. Allerdings wurde dabei noch nicht auf die informatischen Grundlagen Bezug genommen. Die Transkription der Diskussion ist den Erhebungsdaten beigefügt.

5 Diskussion

Ziele des Workshops sind der Aufbau informatischer Kompetenz im Kontext KI, die Stärkung der Diskussionsfähigkeit der Studierenden durch passende informatische Kompetenzen und die Anregung der Studierenden zum Transfer auf passende Themenstellungen im Politikunterricht.

Die Pilotierung zeigt, dass KI-Systeme im Alltag sowie in politischen Themen identifiziert werden können. Dennoch ist unklar, ob Punkte wie Social-Credit-Systeme wirklich als Anwendung für KI-Systeme zur Bewältigung großer ungeordneter Datenmengen erkannt oder ob lediglich bekannte Begriffe aus der Schnittmenge Politik und Digitalisierung genannt wurden. Abgesehen von Deepfakes, die im Selbstlernteil erwähnt wurden, waren keine der in der Diskussion genannten Beispiele bereits im Workshop enthalten. Für alle diese Beispiele lässt sich auch eine mittel- bis langfristige gesellschaftliche Relevanz feststellen, wie sie in den Kompetenzen zur politischen Urteilsfähigkeit von der Gesellschaft für Politikdidaktik und politische Jugend- und Erwachsenenbildung

(GPJE) in [De04] gefordert wird. Trotz der fehlenden informatischen Dimension der Argumente erscheint es plausibel, dass die Teilnehmenden die informatischen Inhalte in politischen Spannungsfeldern identifizieren können.

Kritisch ist anzumerken, dass für 90 Minuten sehr viele informatische Inhalte im Workshop enthalten sind. EVA-Prinzip, der Algorithmusbegriff, Neuronale Netze und die Grundprinzipien von maschinellem Lernen wurden in eine Sachstruktur eingebettet, die gleichzeitig an Alltagserfahrungen anknüpft und dennoch logisch aufeinander aufbauend zum Thema KI hinführt. Für Teilnehmende ohne informatische Vorkenntnisse ist des Workshop trotz Reduktion der Themen sehr anspruchsvoll und sollte in zukünftigen Erprobungen mindestens auf die doppelte Zeit ausgeweitet werden.

Um zukünftig die gelernten Inhalte im Workshop und die Wirkung auf die teilnehmenden Personen besser beurteilen zu können, wird ein Wissenstest der vermittelten informatischen Inhalte sowie eine Pre-Post-Befragung zur Zuversicht zur Vermittlungskompetenz in Bezug auf maschinelles Lernen in neuronalen Netzen im Unterricht konzipiert. Der Wissenstest dient der Überprüfung des aktuellen Wissensstandes bezüglich der dekonstruierten Komponenten des KI-Informatiksystems und damit als Unterstützung bei der Einordnung von Äußerungen im Rahmen der Gruppendiskussion. Für die erfolgreiche Integration von informatischen Themen in andere Fächer ist eine positive Selbstwirksamkeitserwartung der (angehenden) Lehrkräfte bezüglich der Vermittlung von facheigenen Inhalten, die auf Aspekten der Informatik aufbauen, ausschlaggebend. Daher ist neben dem Bewusstsein für die Relevanz des Themas auch die Zuversicht auf die eigenen Vermittlungskompetenzen für die spätere Umsetzung im Unterricht nötig. Geplant sind weitere Durchführungen des Workshops sowie die Begleitung der Teilnehmenden in praktischen Unterrichtserprobungen mit direkten informatischen Berührungspunkten.

Literatur

[Ba20] Barkmin, M.; Bergner, N.; Bröll, L.; Huwer, J.; Menne, A.; Seegerer, S.: Informatik für alle?! – Informatische Bildung als Baustein in der Lehrkräftebildung. In (Beißwenger, M.; Bulizek, B.; Gryl, I.; Schacht, F., Hrsg.): Digitale Innovationen und Kompetenzen in der Lehramtsausbildung. Universitätsverlag Rhein-Ruhr, Duisburg, Nov. 2020.

[Br08] Brinda, T.; Fothe, M.; Friedrich, S.; Koerber, B.; Puhlmann, H.; Röhner, G.; Schulte, C.: Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I, Techn. Ber., 2008.

[De04] Detjen, J.; Kuhn, H.-W.; Massing, P.; Richter, D.; Sander, W.; Weißeno, G.; für Politikdidaktik und Politische Jugend- und Erwachsenenbildung, G., Hrsg.: Nationale Bildungsstandards für den Fachunterricht in der Politischen Bildung an Schulen: ein Entwurf. Wochenschau-Verl, Schwalbach/Ts, 2004, isbn: 978-3-89974-112-4.

[DL17] Danks, D.; London, A. J.: Algorithmic Bias in Autonomous Systems. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, S. 4691–4697, Aug. 2017, isbn: 978-0-9992411-0-3, url: www.ijcai.org/proceedings/2017/654, Stand: 30.07. 2021.

[Ga17] Gallenbacher, J.: Allgemeinbildung in der digitalen, gestalteten Lebenswelt. Gesellschaft für Informatik, Bonn, 2017, isbn: 978-3-88579-668-8.

[Hi19] Hitron, T.; Orlev, Y.;Wald, I.; Shamir, A.; Erel, H.; Zuckerman, O.: Can Children Understand Machine Learning Concepts? The Effect of Uncovering Black Boxes. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, S. 1–11, Mai 2019, isbn: 978-1-4503-5970-2, url: https://doi.org/10.1145/3290605.3300645, Stand: 21.06.2021.

[Hu20] Humbert, L.; Best, A.; Micheuz, P.; Hellmig, L.: Informatik – Kompetenzentwicklung bei Kindern./, 2020, issn: 1432-122X, url: http://dl.gi.de/handle/20.500.12116/33218, Stand: 07.02.2021.

[LH19] Losch, D.; Humbert, L.: Informatische Bildung für alle Lehramtsstudierenden: Reformprozess einer allgemeinbildenden Informatikveranstaltung in der universitären Lehrerbildung./, 2019, issn: 1617-5468, Stand: 13. 10. 2020.

[Ma03] Magenheim, J.: Informatik Lernlabor - Systemorientierte Didaktik in der Praxis. In (Hubwieser, P., Hrsg.): Informatische Fachkonzepte im Unterricht, INFOS 2003, 10. GI-Fachtagung Informatik und Schule. Gesellschaft für Informatik e.V., Bonn, S. 12–30, 2003.

[SB02] Schulte, C.; Block, U.: Das Sieben-Schritte-Schema zur Dekonstruktion objektorientierter Software. In (Schubert, S. E.; Magenheim, J.; Hubwieser, P.; Brinda, T., Hrsg.): Forschungsbeiträge zur „Didaktik der Informatik" - Theorie, Praxis, Evaluation. Gesellschaft für Informatik e.V., Bonn, S. 3–12, 2002.

[Sc03] Schulte, C.: Lehr- Lernprozesse im Informatik-Anfangsunterricht. Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II, Diss., Paderborn: Universität Paderborn, Okt. 2003.

[Sc14] Schattschneider, J.: Politik- und Informatikunterricht. In (C., D.; Tischner, C. K., Hrsg.): Handbuch Fächerübergreifender Unterricht in der politischen Bildung. Wochenschau Verlag, Schwalbach, S. 225–263, 2014.

[SGB21] Schultze, S.; Gruenefeld, U.; Boll, S.: Demystifying Deep Learning: Developing and Evaluating a User-Centered Learning App for Beginners to Gain Practical Experience. i-com 19/3, S. 201–213, 26. Jan. 2021, issn: 2196-6826, 1618-162X.

Mediendesigninformatik – Erfahrungen mit einem interdisziplinären Bachelor-Studiengang

Volker Ahlers[1], Jürgen Dunkel[1], Christian Gädtke[2], Arne Koschel[1], Jonas Schild[1], Timo Schnitt[2]

Abstract:
Der Bachelor-Studiengang Mediendesigninformatik der Hochschule Hannover ist ein Informatikstudiengang mit dem speziellen Anwendungsgebiet Mediendesign. In Abgrenzung von Studiengängen der Medieninformatik liegt der Anwendungsfokus auf der kreativen Gestaltung etwa von 3D-Modellierungen, Animationen und Computerspielen. Absolvent*innen des Studiengangs sollen an der Schnittstelle zwischen Informatik und Mediendesign agieren können, zum Beispiel bei der Erstellung von Benutzungsschnittstellen und VR/AR-Anwendungen. Der Artikel stellt das Curriculum des interdisziplinären Studiengangs vor und reflektiert nach dem Abschluss der ersten beiden Studierendenkohorten die Erfahrungen, indem die ursprünglichen Ziele den Zahlen der Hochschulstatistik und den Ergebnissen zweier Studierendenbefragungen gegenübergestellt werden.

Keywords:
Curriculumsentwicklung; Interdiziplinäre Studiengänge; Angewandte Informatik; Mediendesign; Mediendesigninformatik; Programmierausbildung; Mediendesignausbildung; Praxisprojekte

1 Motivation

Softwareentwicklung besteht längst nicht mehr allein aus dem Entwurf von Architekturen und dem Programmieren effizienter Algorithmen. Unter anderem aufgrund der fortschreitenden Durchdringung immer weiterer Lebensbereiche durch digitale Technologien und die Etablierung neuartiger Nutzerschnittstellen (z. B. mobil, web-basiert, Augmented Reality) kommt Design- und Usability-Aspekten eine immer größere Rolle zu. 3D-Modellierungen, Animationen und Game Engines finden inzwischen auch außerhalb der Spieleindustrie vielfältige Anwendungen [Ab20; Da21]. Um dem Bedarf an entsprechend ausgebildeten Fachkräften zu begegnen, wurde an der Hochschule Hannover zum Wintersemester 2015/16 der Studiengang Mediendesigninformatik (B. Sc.) eingeführt. Nach dem Abschluss der ersten beiden Studierendenkohorten werden im Folgenden Erfahrungen mit dem Studiengang reflektiert. Zunächst werden in Abschnitt 2 die Ziele des Studiengangs beschrieben. In Abschnitt 3 wird das Curriculum unter besonderer Berücksichtigung der Design- und Programmierausbildung sowie der Rolle von Projekten vorgestellt. Abschnitt 4 stellt die Ziele den Zahlen der Hochschulstatistik und den Ergebnissen zweier Studierendenbefragungen gegenüber. Abschnitt 5 fasst Erfahrungen und Herausforderungen zusammen.

2 Entstehungsgeschichte und Ziele

An der Hochschule Hannover gibt es einen seit vielen Jahren etablierten und gut nachgefragten Bachelor-Studiengang (B. Sc.) Angewandte Informatik.[3] Im Laufe der Jahre haben sich dennoch einige Herausforderungen gezeigt: Im Zuge des Hochschulpakts 2020 (HP2020) und durch eine Absenkung des curricularen Normwerts (CNW) hat sich die Aufnahmekapazität zwischen 2004 (Umstellung auf Bachelor/Master) und 2012 von 58 auf 113 nahezu verdoppelt, wodurch sich die angestrebte Gruppengröße sowie die typische Lehr- und Lernatmosphäre eines Fachhochschul- bzw. HAW-Studiengangs immer weniger realisieren ließen. Der Anteil weiblicher Studierender blieb trotz vielfältiger Maßnahmen (Werbung in Schulen, Roboter-Ferienkurse speziell für Schülerinnen, Niedersachsen-Technikum) auf konstant niedrigem Niveau zwischen 5% und 10%. Schließlich bewegten sich zwischenzeitlich die

1 Hochschule Hannover, Fakultät IV, Abteilung Informatik, Ricklinger Stadtweg 120, 30459 Hannover, {volker.ahlers,juergen.dunkel,arne.koschel,jonas.schild}@hs-hannover.de

2 Hochschule Hannover, Fakultät III, Abteilung Design und Medien, Expo Plaza 2, 30539 Hannover, {christian.gaedtke,timo.schnitt}@hs-hannover.de

3 https://f4.hs-hannover.de/studium/bachelor-studiengaenge/angewandte-informatik-bin

(steigende) Aufnahmekapazität und die (sinkende) Zahl der Bewerbungen aufeinander zu, so dass die Gefahr bestand, nicht alle Studienplätze besetzen zu können.

Aufgrund dieser Beobachtungen und durch die Aufforderung, Vorschläge zur Verstetigung von HP2020-Studienplätzen einzureichen, wurden ab dem Jahr 2012 Überlegungen angeregt, einen neuen Studiengang einzuführen und eine begrenzte Zahl von (HP2020-)Studienplätzen des bestehenden Studiengangs in diesen zu überführen. Unabhängig von der fachlichen Ausrichtung wurden die folgenden organisatorischen Ziele angestrebt:

- Realisierung kleinerer Gruppengrößen mit der vorhandenen oder einer geringfügig erhöhten Lehrkapazität.
- Erhöhung des Anteils weiblicher Studierender.
- Verbreiterung des Studienangebots zur Ansprache neuer Zielgruppen.

Hinsichtlich der fachlichen Ausrichtung stand zunächst fest, dass keine Konkurrenz zu an der Hochschule Hannover bereits vorhandenen Studiengängen angestrebt werden soll, wie z.B. der Wirtschaftsinformatik und mehrerer Studiengänge im Bereich Ingenieurinformatik. Nach ersten Überlegungen in Richtung Bioinformatik und Scientific Computing wurde die Medieninformatik in den Blick genommen. Da in der näheren Umgebung mehrere Medieninformatikstudiengänge existieren und es an der Fakultät III der Hochschule Hannover eine sehr aktive Abteilung Design und Medien gibt, wurde Kontakt zu den Lehrenden des Studiengangs Mediendesign aufgenommen, um einen Studiengang mit stärkerem Fokus auf dem Design zu konzipieren.

Der Studiengang Mediendesign hat seinen Fokus auf der digitalen Gestaltung von 3D-Animationen, u. a. für Anwendungen in Animationsfilmen und zunehmend Computerspielen.[4] Gerade bei letzteren zeigte sich, dass Mediendesignstudierende häufig an technische Grenzen im Bereich Programmierung stoßen. Umgekehrt fehlen Informatikstudierenden häufig gestalterische Fähigkeiten, sofern sie nicht entsprechende Erfahrungen aus einer vorherigen Ausbildung mitbringen. Ein Kernziel des neuen Studiengangs war daher, beide Bereiche – Programmierung und Software-Entwicklung auf der einen sowie 3D-Modellierung und Animation auf der anderen Seite – zusammenzubringen.

Die zunächst vorgeschlagene Studiengangsbezeichnung *Designinformatik* erschien zu umfassend, da viele auch an der Hochschule Hannover gelehrte Teilgebiete wie Produkt-, Grafik- und Modedesign nicht im Fokus des neuen Studiengangs stehen. Eine naheliegende Alternative war dann die endgültige Bezeichnung *Mediendesigninformatik* mit der Kurzform *MDI*. Der Studiengang nahm zum Wintersemester 2015/16 mit 35 Studienplätzen seinen Lehrbetrieb auf und wurde im Frühjahr 2016 durch die ASIIN akkreditiert.

3 Curriculum

Ziel des Studiengangs Mediendesigninformatik (MDI) ist es, Informatiker*innen mit vertieften Kenntnissen im Bereich Mediendesign auszubilden. Es sollten daher alle grundlegenden Themen der Informatik abgedeckt werden, wenn auch unweigerlich mit stellenweise reduzierter Tiefe. Grundsätzlich sollte es Absolvent*innen des Studiengangs auch möglich sein, in der klassischen Software-Entwicklung zu arbeiten. Die grobe Leitlinie bei der Planung des Studiengangs lautete „⅔ Informatik, ⅓ Design".

An zahlreichen Hochschulen existieren Studiengänge der Medieninformatik, die zumeist Schwerpunkte in den Bereichen Medientechnik und Mensch-Maschine-Interaktion haben [Ge; He09; Wo19]. In einer 2017 durchgeführten Umfrage mit Rückmeldungen zu 43 Studiengängen wurden die Schwerpunkte HCI und Web-/App-Entwicklung 22-mal bzw. 19-mal genannt, Mediengestaltung hingegen 13-mal [Wo18]. Gleichwohl existieren auch stärker auf Mediendesign ausgerichtete Medieninformatik-Studiengänge, etwa an der Universität Ulm. Ein Kernelement in der Konzeption des Studiengangs Mediendesigninformatik ist demgegenüber, dass neben der Informatik die kreative Mediengestaltung im Fokus steht, welche von Designer*innen an einer Designfakultät gelehrt und im Austausch mit Designstudierenden angewendet wird.

Die Überlegungen mündeten in ein Curriculum, dessen aktuelle Ausprägung Abb. 1 zeigt. Eine wichtige Entscheidung war, welche Module speziell für den neuen Studiengang angeboten werden und welche gemeinsam mit anderen Studiengängen belegt werden sollen. Zum einen bestand aus Kapazitätsgründen die Notwendigkeit, Synergieeffekte mit bestehenden Lehrveranstaltungen zu nutzen. Zum anderen sollen die MDI-Studierenden in den Austausch mit Studierenden der Angewandten Informatik und des Me-

4 https://f3.hs-hannover.de/studium/bachelor-studiengaenge/mediendesign-bme

diendesigns kommen, sich gleichzeitig aber auch als eigene Gruppe wahrnehmen können. An der Fakultät IV wurden die Module Mobile Computing und Usability sowie die Praxis- oder Auslandsphase speziell für den MDI-Studiengang neu eingeführt. Darüber hinaus wurde entschieden, die Module in den Bereichen Computergrafik und Programmieren mit eigenen Inhalten für MDI-Studierende anzubieten. Eine Besonderheit stellt das Startprojekt dar, welches den Studierenden ermöglicht, bereits im 1. Semester Erfahrungen in der Projektarbeit zu sammeln [DG16]. Dieses Modul wird ebenfalls mit speziellen Themen und eigenen Gruppen für MDI-Studierende angeboten.

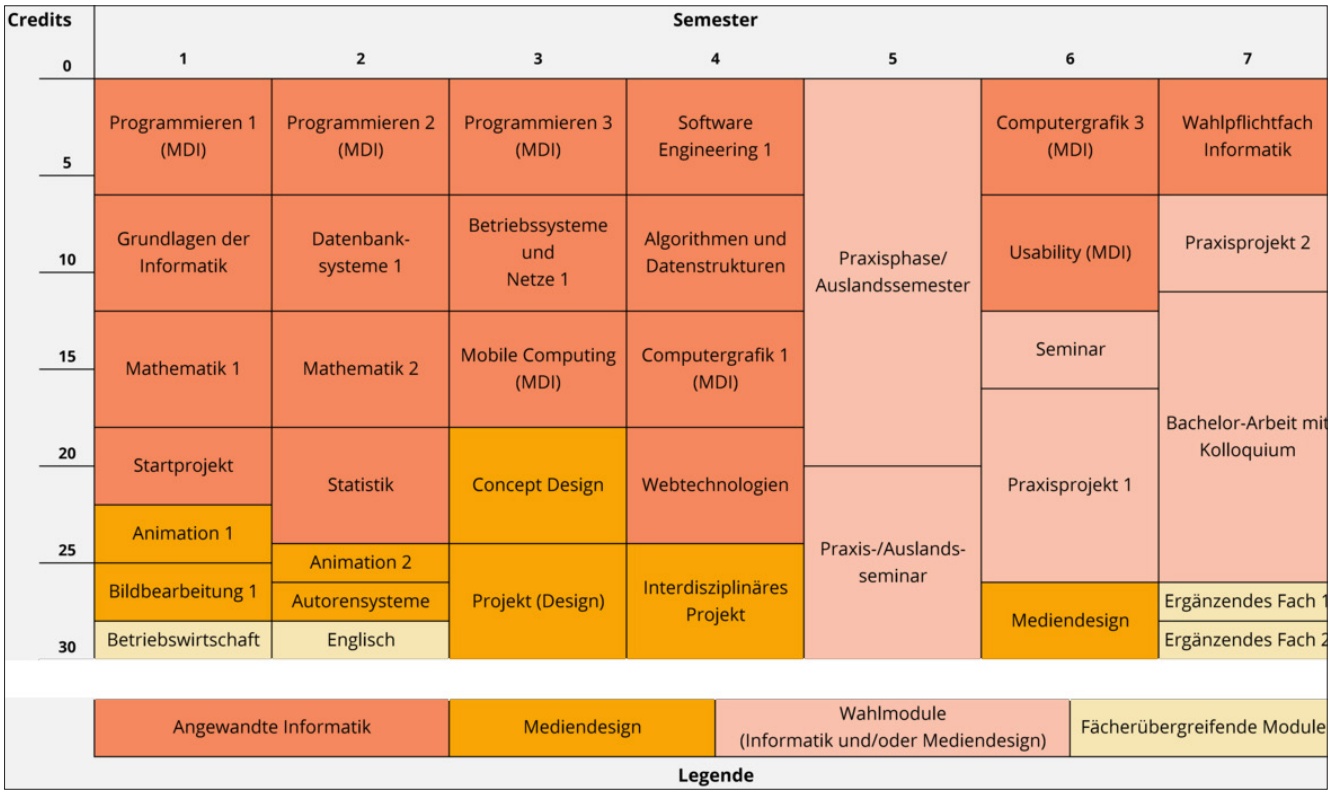

Abb. 1: Curriculum des Studiengangs Mediendesigninformatik (MDI).[5] Mit „(MDI)" bezeichnete rot dargestellte Module werden speziell für den Studiengang MDI angeboten, die übrigen Module gemeinsam für die Studiengänge Angewandte Informatik und MDI. Orange dargestellte Module werden teilweise speziell für MDI angeboten, teilweise gemeinsam für Mediendesign und MDI.

Der Studiengang wurde als Studiengang vom Typ 2 („Informatik-Studiengänge mit einem speziellen Anwendungsbereich") gemäß den Empfehlungen der Gesellschaft für Informatik konzipiert [Zu16]. Der Studiengang ist dementsprechend organisatorisch in der Abteilung Informatik der Fakultät IV angesiedelt, welche auch den Prüfungsausschuss stellt, der gemeinsam für die Studiengänge Angewandte Informatik (B. Sc./M. Sc.) und MDI zuständig ist. Prüfungs- und Praxisphasenordnungen müssen dennoch von den Fakultätsräten beider beteiligter Fakultäten beschlossen werden. Absprachen zwischen den Lehrenden beider Fakultäten werden in einem regelmäßigen Jour Fixe getroffen.

3.1 Ausbildung im Bereich Softwareentwicklung

Zentraler Schwerpunkt im Studiengang Mediendesigninformatik (MDI) ist die Softwareentwicklung in all ihren Facetten. Die Absolvent*innen sollen komplexe Anwendungssysteme entwickeln können, die mit innovativen Bedienkonzepten den ständig wachsenden ergonomischen und ästhetischen Anforderungen genügen.

Aus diesem Grund sollen den MDI-Studierenden die selben Kenntnisse und Fähigkeiten wie Studierenden an klassischen Informatikstudiengängen vermittelt werden. Deshalb wurden alle Kernmodule des Studiengangs 'Angewandte Informatik' aus dem Bereich Softwareentwicklung übernommen. Eine besondere Bedeutung kam jedoch der Auswahl der gelehrten Programmiersprachen zu. Hier sollten die besonderen Rahmenbedingungen des MDI-Studiengangs berücksichtigt werden.

5 https://f4.hs-hannover.de/studium/bachelor-studiengaenge/mediendesigninformatik-mdi

- Die MDI-Studierenden verfügen zu Studienbeginn meist über weniger Programmiererfahrungen als Bewerber*innen für klassische, eher technisch ausgerichtete Informatikstudiengänge. Diese Annahme bestätigte sich durchweg in den vergangenen Jahren: Viele MDI- Studienanfänger*innen interessieren sich gerade für die Synergie von Technik und Gestaltung. Sie sind daher nicht so stark technisch ausgerichtet und besitzen oft noch gar keine Programmiererfahrung. Daher sollte mit einer besonders einfach zu erlernenden Programmiersprache das Studium begonnen werden. Letztendlich wurde aus verschiedenen Kandidaten Python ausgewählt. Python weist einen schlanken Sprachkern mit einer einfachen Syntax auf und hat sich inzwischen an vielen Hochschulen als erste Programmiersprache etabliert und bewährt. Die Wahl fiel insbesondere auf Python, weil die Sprache bereits in vielen Lehrveranstaltungen im Bereich Mediendesign verwendet wurde, bspw. als Skriptsprache in der dort eingesetzten 3D-Animationssoftware Maya.

- Weiterhin musste die Kompatibilität des MDI-Studiengangs mit den anderen an der Hochschule Hannover angebotenen Informatik-Studiengängen gewährleistet werden. Aus organisatorischen und kapazitiven Gründen werden viele Lehrveranstaltungen den Studierenden der beiden Bachelor-Studiengänge angeboten. Deshalb setzt sich die Programmierausbildung aktuell im 2. und 3. Semester mit den Programmiersprachen Java (1,5 Lehrveranstaltungen) und C/C++ (0,5 Lehrveranstaltung) fort. Das Java-Ökosystem wird in den meisten Modulen als technische Basis genutzt, bspw. in den Bereichen Software Engineering, Mobile Computing und Datenbanksysteme. Die Sprache C/C++ wird in den Modulen Betriebssysteme und Computergrafik eingesetzt.

Bisher konnten die folgenden Erfahrungen gewonnen werden: Insgesamt hat sich bewährt, mit Python als erster Programmiersprache zu beginnen. Insbesondere absolute Programmieranfänger*innen profitieren von dem kleinen Sprachumfang und der einfachen Syntax, die eine einfache Formulierung von Algorithmen ermöglicht. Vorteilhaft erweist sich auch, dass fortgeschrittene Konzepte wie Objektorientierung und Exception-Handling erst im zweiten Teil den Vorlesung eingeführt werden müssen, wenn grundlegende Programmiertechniken bereits erlernt sind.

Hauptproblem und Kritikpunkt der derzeitigen Programmierausbildung ist allerdings, dass aktuell zu viele Programmiersprachen unterrichtet werden, die darüber hinaus konzeptionell sehr ähnlich sind und deswegen zu keinem großen Erkenntnisgewinn beitragen. Neben kurzen Lehreinheiten zu Sprachen wie PHP und JavaScript bzw. TypeScript im Rahmen des Moduls Webtechnologien kommt zu den genannten Sprachen auch noch C# hinzu, das Basis für die Spieleentwicklung mit Unity ist. Aufgrund der sehr starken Ähnlichkeit mit Java wird die Verwendung von C# aber weder von den Studierenden noch den Lehrenden als Problem wahrgenommen. Damit sich die Studierenden auf wenige Programmiersprachen konzentrieren können, soll die grundlegende Programmierausbildung in Zukunft ausschließlich mit Python und Java erfolgen und auf C/C++ verzichtet werden. Dadurch wird das Curriculum geschärft, ohne wesentliche Inhalte aufzugeben. Der gewonnene Freiraum soll durch ein Softwareentwicklungsprojekt genutzt werden, in dem die Studierenden ihre Programmierkenntnisse in kleinen Gruppen anwenden können.

3.2 Mediendesignausbildung

Die Designausbildung findet im Studiengang Mediendesign der Fakultät III statt, der sich in drei Schwerpunkte aufteilt (Animation, Game Design und Film). In den ersten drei Semestern werden die Designgrundlagen gelegt. In den Modulen Animation 1 und 2 werden die Studierenden der Studiengänge Mediendesign und MDI in Gruppen gemischt. Hierdurch soll u. a. gewährleistet werden, dass sich die MDI- und Mediendesignstudierenden kennenlernen und in gemeinsamen Gruppenaufgaben gegenseitig unterstützen. Die Studierenden lernen grundlegende Techniken des 3D-Modellierens, Texturierens, Animierens und Renderns in einer zeitgemäßen 3D-Software anwenden.

Im Modul Bildbearbeitung lernen die Studierenden grundlegende Techniken und Prinzipien der digitalen Bildbearbeitung und wenden diese auf einfache aus den Bereichen Bildretusche und -manipulation an. Weiterhin erlernen sie aktuelle Layout-Techniken zum Gestalten von Präsentationen und erfahren eine Einführung in die Druckvorstufe. Das Modul Game Design vermittelt den praktischen Umgang mit Autorensystemen zur Umsetzung interaktiver Inhalte, inkl. der Überführung konzeptioneller Ansätze in die technische Realisierung, des Einsatzes von Skripten zur Interaktion und der Auswahl und Aufbereitung medialer Assets.

In den Modulen Concept Design und Projekt Design lernen die Studierenden grundlegende Techniken und Prinzipien der visuellen konzeptionellen Arbeit und setzen diese in einem selbst gewählten Medienprojekt aus einem der Bereiche Animation oder Game Design in Produktionsabläufe um. Die Studierenden beherrschen anschließend u. a. die Planung und Durchführung eines Medienprojekts, das Erstellen von Konzeptvisualisierungen, die Definition des Looks sowie die technische Umsetzung (Animation, Compositing, Schnitt, digitale Endfertigung).

Ab dem 4. Semester arbeiten die Studierenden in Mediendesign-Entwurfsprojekten; der Schwerpunkt (3D-Animation oder Game-Design) kann pro Semester frei gewählt werden. Innerhalb dieser Projekte sollen praxisorientierte Aufgaben gemeinsam mit den Designstudierenden gelöst werden. Angestrebt und wünschenswert sind Projekte mit Unternehmen oder öffentlichen Einrichtungen. Im Modul Mediendesign werden schließlich weiterführende Themen wie zum Beispiel Content Design für webbasierte Systeme behandelt.

3.3 Praxisprojekte

Der Studiengang enthält im ersten Studienabschnitt im 3. und 4. Semester mehrere Projektveranstaltungen (Concept Design, Design Projekt, Interdisziplinäres Projekt), die im Rahmen der Mediendesignausbildung von den Lehrenden der Fakultät III und teilweise gemeinsam mit Studierenden des Studiengangs Mediendesign durchgeführt werden.

Im 6. und 7. Studiensemester findet jeweils ein Praxisprojekt statt, welches im 6. Semester mit 10 Credits und im 7. Semester mit 5 Credits bemessen wird; im 7. Semester ist so der Bachelor-Arbeit der überwiegende Fokus eingeräumt. Die Praxisprojekte werden zu gleichen Teilen von Lehrenden der Informatik und des Mediendesigns betreut. Entsprechend der beiden Fachrichtungen haben sich zwei unterschiedliche Durchführungsformen herausgebildet, zwischen denen die Studierenden zu Beginn des 6. Semesters wählen können:

- In der Abteilung Informatik der Fakultät IV werden die beiden Praxisprojekte als zusammengehöriges Projekt über zwei Semester durchgeführt, für welches eine größere Gruppe von 8 bis 15 Studierenden mit einem vorgegebenen Thema im Bereich Softwareentwicklung (z. B. Virtual-Reality-Trainingssimulation, Multiplayer-Computerspiele) betraut wird. Dabei wird insbesondere die Aufgabe gestellt, die Konzeption, Implementierung und Evaluation der User Experience in einer größeren Gruppe und in agilen, iterativen Prozessen zu organisieren. In der ersten Projekthälfte soll ein funktional möglichst weit getriebener Prototyp entstehen, der in der zweiten Hälfte durch Tests und Feinschliff in einen produktreifen Zustand gebracht wird.

- In der Abteilung Design und Medien der Fakultät III finden die beiden Praxisprojekte als einzelne Projekte mit ähnlichen Organisationsformen wie im Grundstudium statt. Die Studierenden haben hier überwiegend die Möglichkeit, eigene Themen zu entwickeln und in selbst organisierten Konstellationen umzusetzen. Es stehen aber auch konkrete Zielthemen von Seiten der Lehrenden zur Wahl.

Durch diese beiden Durchführungsformen wird den Studierenden ein vielfältiges Angebot unterbreitet. Die Nachfrage nach beiden Angeboten bei der Wahl war in den ersten drei Jahren gleichmäßig oder leicht in Richtung Mediendesign orientiert. In persönlichen Gesprächen wurde als Grund für eine Wahlrichtung Mediendesign oft die größere persönliche und fachliche Freiheit angeführt. Für die Informatik spräche eine größere fachliche und organisatorische Herausforderung und ein interessantes, forschungsnahes Thema.

Gerade hinsichtlich der Strukturiertheit bzw. des Vorwissens im Projektmanagement wurden Unterschiede zu den Studierenden des Studiengangs Angewandte Informatik deutlich, welche verschiedene Vorgehensmodelle in der Pflichtveranstaltung Software Engineering 2 kennenlernen. Diese Lehrveranstaltung wird in der Mediendesigninformatik als Wahlmodul angeboten. Entsprechend konnten Studierende, die ein solches Wahlmodul gewählt hatten oder jene, die im Rahmen ihrer Praxisphase in agilen Prozessen gearbeitet hatten, einen deutlichen Vorsprung vorweisen. In Zukunft wird daher für die Projekte beider Abteilungen im Rahmen eines Vorkurses der Abschnitt zu Vorgehensmodellen und Softwareentwicklungsprozessen als freiwilliges, zusätzliches Lehrangebot organisiert.

Hinsichtlich der Inhalte werden die beiden Praxisprojekte auch als Ankerpunkte für die Verknüpfung zwischen Forschung und Lehre gesehen. So wurden mehrere Projekte beider Abteilungen mit aktuellen Drittmittelprojekten oder externen Firmenkollaborationen verbunden. Den Studierenden wurden so Einblicke in aktuelle Forschungsthemen auf internationalem Niveau geboten, beispielsweise zu Multi-user Virtual Reality Training oder Photogrammetrie bzw. Digitalisierung großer Umgebungen.

Beispielprojekt Multi-User Virtual Training Experiences (MuViTEx):
Im Studienjahr 2018/19 fand ein Praxisprojekt zur prototypischen Erprobung von Mehrbenutzertrainingssystemen für Notfallsanitäter mit neun Teilnehmenden statt. Dieses wurde eng verknüpft mit aktuellen Inhalten aus dem BMBF-/EU-geförderten Projekt EPICSAVE (FKZ 01PD15004, [Sc21]).[6] So arbeiteten sich die Studierenden in der ersten Projekthälfte in die Ergebnisse der Anforderungsanalyse von Notfallsanitätern aus dem Forschungsprojekt sowie in die Themen Mehrbenutzerinteraktion, Virtual Reality und Serious Game Design ein. Den Studierenden wurde ein von wissenschaftlichen Mitarbeitern des Projekts EPICSAVE entwickeltes, Unity-basiertes Framework für vernetzte Trainingswelten samt VR-Hardware (Head-mounted Displays, Controller) zur Verfügung gestellt.

6 ttps://www.epicsave.de

Im ersten Projektsemester wurde ein entsprechender, funktionaler Prototyp (Alpha-Status) fertig gestellt, auch wenn besonders die selbstständige Organisation in einer größeren, auch leistungsmäßig inhomogenen Gruppe eine Herausforderung für die Studierenden darstellte. Auf eigenes Betreiben organisierten die Studierenden zwischen den Semestern intensive Arbeitswochen, in denen sie sich gemeinsam im Team ausschließlich dem Projekt widmeten. So konnte ein funktional ausgefeilter Beta-Prototyp im November präsentiert werden, der bis Ende Januar 2019 noch hinsichtlich Fehleranfälligkeit und Benutzbarkeit optimiert wurde. Das Ergebnis, ein Multiplayer Virtual Reality Serious Game eines medizinischen Trainings mit Storytelling, multimodalem Dialogsystem, 3D-Inventar, Minigames und Missionssystem (siehe Abb. 2), wurde auf der anschließenden Postermesse von den anderen Studierenden der Abteilung ausgezeichnet.

Bemerkenswert war aus Lehrendensicht, dass die Studierenden die organisatorischen Fehler im ersten Projektsemester erkannten und eigene Wege fanden, diese Schwachstelle im darauffolgenden Semester zu beseitigen. Besonders hilfreich war laut Evaluation eine disziplinierte Anwesenheit an regelmäßigen Terminen, insbesondere in intensiven Hackathons. Aus Forschendensicht war die mit dem zweisemestrigen Verlauf einhergehende fachliche Vertiefung ebenso zuträglich, da einige im Projekt entwickelten zusätzlichen Module des VR-Frameworks (insb. ein Skripting-System für Missionen und Aufgaben) anschließend von einer beteiligten Studentin als Hilfskraft wieder in das Forschungsprojekt überführt werden konnten, was für ein Bachelor-Projekt ein eher seltenes Resultat ist.

Abb. 2: Multiplayer VR Serious Game des an der Fakultät IV durchgeführten zweisemestrigen Praxisprojekts MuViTEx mit Patientin, Intubations-Minigame und Dialogsystem (oben) sowie realer bzw. virtueller Übergabe und gescriptetem Missionssystem (unten).

Der Trainingsprozess beginnt mit einem untrainierten Netz, in dem die Verknüpfungen und Neuronen mit meist zufälligen Startwerten angeordnet werden. Anschließend werden Trainingsdaten als Eingabedaten verwendet, um eine Ausgabe zu erzeugen.

3.4 Bachelor-Arbeiten

Die Bachelor-Arbeit kann wahlweise in der Informatik oder im Mediendesign geschrieben werden, teilweise unter gemeinsamer Betreuung durch Lehrende aus beiden Bereichen. Vielfach werden im Rahmen der Praxisphase erschlossene Kontakte genutzt, um mit Firmen gemeinsam betreute Abschlussarbeiten zu ermöglichen. Einige Bachelor-Arbeiten werden auch im Rahmen von Drittmittelprojekten durchgeführt und von Promovierenden co-betreut. Beispielhaft seien sechs Themen der letzten Jahre genannt:

- Entwicklung einer Renderfarm-Software für den 3D-Animationsbereich, die es erlaubt, Aufträge an eine aus 50 Rechnern bestehende Renderfarm zu leiten.
- Entwicklung eines Materialscanners, mit dem mittels Photogrammetrie eine Materialbibliothek für Design-Studiengänge erstellt wird (siehe Abb. 3 links).
- Beteiligung an einem Forschungsprojekt zur Entwicklung eines 3D-Scanners, mit dessen Hilfe sich Kulturgüter in Museen digitalisieren lassen (siehe Abb. 3 rechts).
- Explorative Untersuchung visueller Effekte zur Steigerung der Präsenzwahrnehmung in virtuellen Realitäten.
- Vergleichende Implementierung von Navigationsverfahren wie Teleportation oder auf Redirected Walking basierender Portale zur Verminderung der Simulator Sickness.

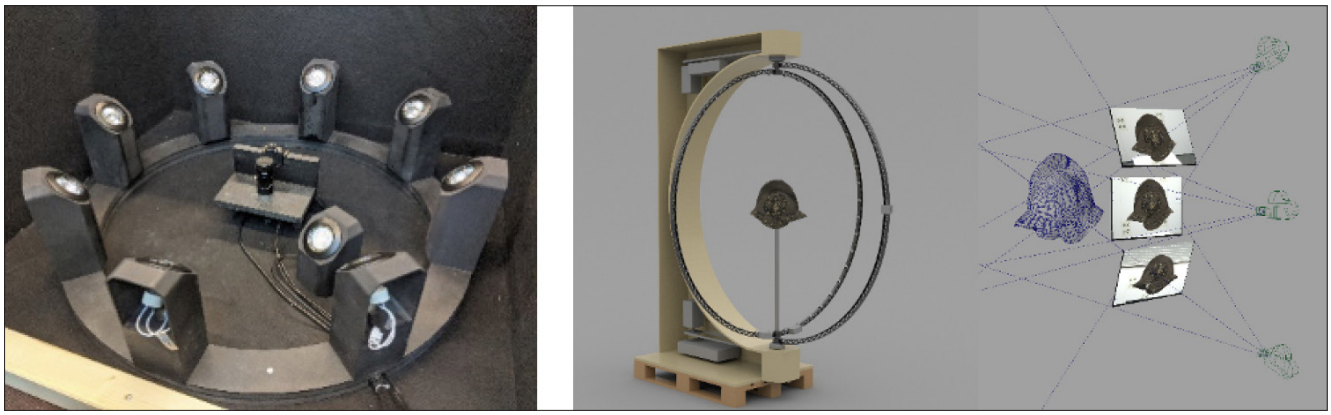

Abb. 3: An der Fakultät III entwickelter Materialscanner zur Erzeugung digitaler 3D-Materialien mittels Photogrammetrie (links) und 3D-Scanner zur Erzeugung digitaler 3D-Objekte (rechts).

4 Evaluation

Statistische Daten aus dem hochschulinternen Lehrbericht für die Lehreinheit Informatik zeigen, dass viele der ursprünglichen Ziele der Einführung des Studiengangs Mediendesigninformatik (MDI) erreicht wurden. Der Studiengang ist sehr gut nachgefragt, die Zahl der Bewerbungen war in den vergangenen Jahren fünf- bis sechsmal so hoch wie die Zahl der Studienplätze. Die Zahl der Bewerbungen pro Studienplatz für den bestehenden Studiengang Angewandte Informatik hat sich durch die Einführung des MDI-Studiengangs nicht wesentlich verringert, was darauf hindeutet, dass tatsächlich neue Zielgruppen erschlossen wurden. Dies wird auch dadurch deutlich, dass der Anteil weiblicher Studierender im Studiengang MDI mit derzeit 38% deutlich höher ist als in der Angewandten Informatik mit 11%. Auch in den Verteilungen der Hochschulzugangsberechtigungen (Allgemeine Hochschulreife vs. Fachhochschulreife) sowie deren Durchschnittsnoten sind Unterschiede zu erkennen, die auf unterschiedliche Zielgruppen hinweisen. Der Studiengang Mediendesign wiederum spricht aufgrund der klaren Designausrichtung und der künstlerischen Aufnahmeprüfung grundsätzlich eine andere Zielgruppe an als der primär auf die Informatik ausgerichtete Studiengang MDI. Eine Besonderheit des MDI-Studiengangs ist die Praxisphase, die es im Studiengang Angewandte Informatik aufgrund der kürzeren Studiendauer (6 Semester) nicht gibt; diese kann wahlweise als Auslandsphase an einer Partnerhochschule absolviert werden. Entsprechend haben in den Jahren 2017 bis 2021 viermal so viele MDI-Studierende ein Auslandssemester wahrgenommen wie Studierende der Angewandten Informatik.

Über die reinen Zahlen hinaus werden die MDI-Studierenden von den meisten Lehrenden der Fakultät IV auch als (im Mittel) „anders" wahrgenommen als die Studierenden der Angewandten Informatik. Positiv gesehen leisten sie damit einen wesentlichen Beitrag zur Erhöhung der Diversität der Informatikstudierenden. Kritisch gesehen wird jedoch auch, dass sie sich mit einigen eher technischen Bereichen der Kerninformatik oftmals schwer tun. Die ursprüngliche Sorge, dass der Studiengang viele Studierende anziehen könnte, die in erster Linie Medien- und Designinhalte mit ein wenig Informatik erwarten, hat sich von Ausnahmen abgesehen allerdings nicht bestätigt.

Es wurden zwei interne Studierendenbefragungen zur Evaluation des MDI-Studiengangs durchgeführt: 2019 mit den ersten Absolvent*innen bzw. Studierenden kurz vor dem Abschluss (15 Rückmeldungen) und 2020 mit Teilnehmenden des Praxisphasenseminars im 6. Semester (25 Rückmeldungen), jeweils bei einer Kohortengröße von 35 Studierenden.[7] Die generelle Zufriedenheit mit dem Studiengang war sehr hoch: Auf die Frage, ob sie den Studiengang weiterempfehlen würden, antworteten 92% (2019) bzw. 100% (2020) mit „ja". Der Informatikanteil am Studium wird von der Mehrheit der Befragten jedoch als größer als 2/3 wahrgenommen. Auch bezüglich der Verzahnung der Inhalte beider Fakultäten wird Verbesserungsbedarf gesehen. Ihre berufliche Zukunft sehen 50% an der Schnittstelle zwischen Informatik und Mediendesign, 25% eher in der Informatik und 21% eher im Mediendesign; 4% wussten dies noch nicht (Zahlen aus 2020). In zukünftigen Absolvent*innenbefragungen sollen insbesondere die Erfahrungen beim Einstieg in den Beruf und die Tätigkeitsfelder genauer analysiert werden.

7 Der Fragebogen kann auf Anfrage an den Erstautor zur Verfügung gestellt werden.

5 Zusammenfassung

Der Bachelor-Studiengang Mediendesigninformatik verbindet ein Studium der Informatik mit dem speziellen Anwendungsgebiet Mediendesign. Der Fokus liegt auf der kreativen Mediengestaltung, die in der Designfakultät der Hochschule Hannover gelehrt wird. Viele der ursprünglichen Ziele des Studiengangs konnten erreicht werden, u. a. die Erschließung neuer Zielgruppen und die Erhöhung des Anteils weiblicher Studierender. Der Studiengang ist stark nachgefragt und erhält ein positives Feedback seitens der Studierenden.

Ein interdisziplinärer Studiengang mit Beteiligung zweier Fakultäten stellt jedoch auch eine besondere Herausforderung dar. Vergleichsweise unkompliziert verläuft dank frühzeitiger schriftlicher Festlegungen die Aufteilung von Haushaltsmitteln und Lehrverpflichtungen sowie die Zuständigkeit von Gremien. Reibungspunkte entstehen vor allem in organisatorischen Alltagsfragen, insbesondere bei der Erstellung von Stunden- und Prüfungsplänen, die Wechselwirkungen mit den zwei verzahnten Studiengängen Angewandte Informatik und Mediendesign sowie die Raumplanung zweier Fakultäten berücksichtigen müssen, aber auch beim Festlegen von Zeiträumen und Fristen für Lehrveranstaltungen und Prüfungen. Nicht zuletzt treten auch immer wieder die unterschiedlichen Mentalitäten und Vorstellungen von „Designer*innen" und „Informatiker*innen" zutage, die die Zusammenarbeit einerseits spannend machen, andererseits aber auch zu Verständigungsschwierigkeiten führen können. Eine gewisse Herausforderung ergibt sich auch durch die räumliche Trennung. Während die Fakultät IV am Hauptcampus der Hochschule Hannover angesiedelt ist, befindet sich die Fakultät III am ehemaligen Expo-Gelände am Stadtrand von Hannover. Durch die Entfernung der beiden Standorte (12 km bzw. 35 min Fahrzeit im ÖPNV) erfahren die Studierenden unterschiedliche Studienstrukturen. Dies erlaubt es den Studierenden, an bestimmten Tagen in der Woche kreativ arbeiten zu können und den Informatikanteil einmal ruhen zu lassen. Der Austausch mit den Mediendesignstudierenden schafft somit Freiraum etwa für die Planung gemeinsamer Projekte. Ein offensichtlicher Nachteil ist dagegen der lange Anfahrtsweg, der auch einen spontanen Austausch zwischen den Lehrenden erschwert.

Literatur

[Ab20] Abdallah, A.; Primas, M.; Turcin, I.; Traussnigg, U.: The potential of game development platforms for digital twins and virtual labs. In (Lalic, B. et al., Hrsg.): Advances in Production Management Systems. Towards Smart and Digital Manufacturing (APMS 2020). Springer International Publishing, Cham, S. 117–121, 2020.

[Da21] Davis, N.: Mapping what's next for in-car navigation experiences, 2021, url: https://blogs.unity3d.com/2021/03/24/mapping-whats-next-for-incar-navigation-experiences/, Stand: 15.05.2021.

[DG16] Dennert-Möller, E.; Garmann, R.: Das „Startprojekt" – Entwicklung überfachlicher Kompetenzen von Anfang an. In (Schwill, A.; Lucke, U., Hrsg.): Hochschuldidaktik der Informatik (HDI 2016). Universitätsverlag Potsdam, S. 11–23, 2016.

[Ge] Gesellschaft für Informatik e.V., Fachgruppe Medieninformatik: Entwicklung einer Rahmenempfehlung für Studiengänge der Medieninformatik, url: https://fg-mi.gi.de/publikationen/curriculum/, Stand: 15. 05. 2021.

[He09] Herczeg, M.: Medieninformatik in Forschung, Lehre und Praxis. In (Kain, S.; Struve, D.; Wandke, H., Hrsg.): Workshop-Proceedings der Tagung Mensch & Computer 2009. Logos, Berlin, S. 317–328, 2009.

[Sc21] Schild, J.; Luiz, T.; Lerner, D.; Herkersdorf, M.; Neuberger, M.;Wegner, K.: epicsave – Enhanced ParamedIC vocational training with Serious games And Virtual Environments – Optimierung der Berufsausbildung von Notfallsanitätern durch Training mittels Serious Games und Virtuellen Umgebungen: Schlussbericht, Hochschule Bonn-Rhein-Sieg, St. Augustin, 2021, url: www.tib.eu/de/suchen/id/TIBKAT%3A1752317777, Stand: 15. 05. 2021.

[Wo18] Wolters, C.; Heinecke, A. M.; Kindsmüller, M. C.; Noss, C.; Rakow, T. C.; Rumpler, M.: Medieninformatik 2018: MI-Kernkompetenzen und -Färbungen. In (Dachselt, R.; Weber, G., Hrsg.): Mensch und Computer 2018 – Workshopband. Gesellschaft für Informatik, Bonn, S. 933–940, 2018.

[Wo19] Wolters, C.; Kindsmüller, M. C.; Heinecke, A. M.; Rakow, T. C.; Dahm, M.; Jent, S.; Rumpler, M.: Medieninformatik 2019: Kompetenzorientierte Lehr-Lernszenarien in der Medieninformatik. In: Mensch und Computer 2019 – Workshopband. Gesellschaft für Informatik, Bonn, S. 512–517, 2019.

[Zu16] Zukunft, O. et al.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen, Gesellschaft für Informatik, Bonn, 2016, url: https://dl.gi.de/handle/20.500.12116/2351, Stand: 15. 05. 2021.

Separating Algorithmic Thinking and Programming

Maurice Chandoo[1]

Abstract:
We describe an approach to teaching algorithmic thinking and programming and the first experiences that we made with it in practice. The idea is to present computational problems as a certain kind of game that the learner can play in order for them to develop a concrete idea of what constitutes an algorithm. The purpose of this is to emphasize that one can think of algorithms independently of a particular programming language. For the programming part a mini language called machine programs and a method to construct such programs from traces is described.

Keywords:
programming education; computational thinking; notional machines; execution traces

1 Introduction

The ubiquity of computers and their immense computing power in our age has contributed to the perception that programming is a generally useful ability. Consequently, it has become commonplace to teach students in other fields than CS how to program. In contrast to CS students, however, they usually only have one course which does not only need to convey the fundamentals of programming but also a minimal understanding of algorithmic thinking.

We describe an approach to teaching algorithmic thinking and programming that can be implemented as a short learning unit. It emphasizes the separation of algorithm design and implementation: design the algorithm first without thinking about a programming language and then implement it in a principled manner using its execution traces. The purpose of our approach is to diverge from the language-centric perception of programming that is often implicitly conveyed ("what commands do I need to write in this programming language to solve the given task") to a more algorithmically-oriented view: "what do I need to compute and how can this be done with the given primitives". Additionally, it should convey the message that designing and communicating algorithms can be done independently of a programming language [La18]. Due to the formal nature of our approach we believe that it might be particularly suitable for mathematically inclined students.

The general idea for teaching algorithmic thinking in our approach is to present computational problems as games to the learner. The learner can test and demonstrate his or her winning strategy (algorithm) by playing the game. We describe a general mechanism to translate computational problems into games. The basis for this translation is a formal definition of *model of computation* from automata theory described in [Sc67]. For example, Turing machines, push-down automata and finite state machines are examples of models of computation in this sense. The definition does not only describe concepts from theoretical computer science but can also express more playful settings such as Kara the ladybug [RNH00] or technical ones such as a simple CPU.

Models of computation are a subclass of notional machines. A notional machine is described as "the general properties of the machine that one is learning to control" by du Boulay who introduced the concept [dB86]. Sorva has given an extensive survey on the relevance of notional machines in programming education and different research threads in this area [So13]. In particular, forming a mental representation of a notional machine is one of the difficulties that novice programmers face. We aim to minimize this difficulty by starting with a very simple model of computation and then generalizing it to a more complex one.

The idea of teaching algorithmic thinking by letting students play games is fairly natural and has been explored and implemented before. For example, a way of using games for specific computational problems as means to understand and develop algorithms in groups has been described by Futschek and Moschitz [FM10]. By algorithmic thinking we mean a set of abilities related to con-

1 FernUniversität in Hagen, Lehrgebiet für Softwaretechnik und Theorie der Programmierung, Universitätsstraße 11, 58084 Hagen, Deutschland maurice.chandoo@fernuni-hagen.de

structing and understanding algorithms as defined in [Fu06]. We see it as a subset of the broader, more popular term computational thinking [Wi06].

In general, we feel that the complexity of translating a mental representation of an algorithm into a program seems to be underestimated. A learner might know an algorithm for a given problem but does not know how to translate it into a given programming language other than by trial and error. In our approach the learner is taught how a formal representation of an algorithm can be constructed from its traces. The traces can be either written manually or generated from playing the game that is used to present the computational problem.

In [Ch19b] we present a programming method that deals with implementing complex algorithms. The method presented here is a simplified version thereof, which is adapted to the context of models of computations and which does not require prior programming experience. Similar approaches are described in [HLR19] and [DvK10]. All have in common that one starts with executing an algorithm by hand for a concrete input and then generalizing the steps to build a program incrementally. In these trace-based approaches, the user must already possess a mental representation of the algorithm that they want to implement. A related approach called direct-manipulation programming is tested in [ADF19]. The importance of traces and tracing activities in early programming courses has been affirmed in [HJ13].

The paper is structured as follows. The first part contains a formal description of models of computation and how computational problems can be framed as what we call machinecomputer game. Moreover, a sequence of training tasks which build on each other is given. The second part describes machine programs – a formalism used to describe algorithms – and how to construct them from traces. Finally, we briefly report our experiences with this approach in an experimental run with two 16-year-old high school students and what next steps are planned for evaluation and further development of this approach.

2 Models of Computation

A model of computation can be seen as an abstract description of a machine: it consists of buttons (operations) and indicator lamps (predicates), and it resides in a particular state (machine state). The indicator lamps (partially) describe its current state. When a button is pressed the machine changes its state depending only on its current state and which button has been pressed. For example, a Turing machine with tape alphabet {0,1,□} has 5 buttons and 3 indicator lamps. For each character x from the alphabet it has an indicator lamp which is on iff the current cell contains x and a button which writes x to the current cell. It also has buttons to move the head one cell to the left and right. Its state consists of the tape contents and the position of the head.

Formally, a model of computation consists of a set of machine states S, a set of operations (total functions from S to S) and a set of predicates (total functions from S to {0,1}). A counter machine with k registers can hold a non-negative integer in each register; a machine state is a sequence of k such numbers ($S = \mathbb{N}_0^k$). Each register can be incremented or decremented by one (decrementing a register that contains 0 has no effect). This means the machine has $2k$ operations. For each register it has a predicate which holds iff that register contains 0.

A simple task on a counter machine with 3 registers is to copy the number of register 1 to register 2. More specifically, the machine is initialized with an unknown state and a human computer has to operate the machine such that register 1 and 2 contain the value that register 1 had in the beginning. This can be presented as a game to the learner as shown in Figure 1. The difficulty is that the indicator lamps (predicates) only reveal partial information about the machine state, i.e. whether a register contains 0 or not.

A trivial strategy for the computer is to determine the number in register 1 by counting (decrement until 0) and then set register 1 and 2 to this value by repeatedly applying the '+1'-operations. This strategy can be generalized to solve arbitrary tasks: determine the initial machine state, solve the problem without the machine, enter the result into the machine. The issue is that no algorithm is executed on the machine itself. To prevent the learner from resorting to this strategy an additional requirement must be made that it should be possible to teach their strategy to a person who cannot count (e.g. a preschool child). More generally, the computer is only allowed to remember a constant amount of information independent of the input.

The learner can train the first steps of algorithmic thinking by trying to find algorithms for simple models of computation and tasks. If the learner consistently wins the game without resorting to the trivial winning strategy described above then this indicates that they found a correct algorithm. The feedback for the learner is relatively quick (provided the inputs are not too large) and easy to obtain since there is no need to convert the mental representation of the algorithm into a program or an informal description; by playing the game they can check whether their algorithm produces the desired output for a given input.

Any computational problem which can be framed as the task of reaching a particular target machine state from an unknown initial machine state in some model of computation can be presented as machine-computer game. For example, in a gummy bear factory there are two containers: one with sugar and one with food dye. Beneath these two containers is a mixer. A gummy bear consists of three pieces of sugar and two pieces of food dye. There is a button which drops a piece of sugar in the mixer and another one which drops a piece of food dye in the mixer, provided the respective container is not empty. There are also two indicator lamps which show whether the sugar or food dye container is empty. The task is to put as many piece of sugar and food dye as possible into the mixer in the correct ratio (3:2) to produce gummy bears. For example, if there were 60 pieces of sugar and 45 pieces of food dye in the beginning then there should be 60 pieces of sugar and 40 pieces of food dye in the mixer in the end. The challenge is that one does not know how many pieces of sugar and food dye are in the containers in the beginning. Moreover, the trivial strategy involving counting cannot be applied here (because it is not possible to revert the operations of dropping sugar or food dye in the mixer) thus making it a good introductory task to set the learner on the right path.

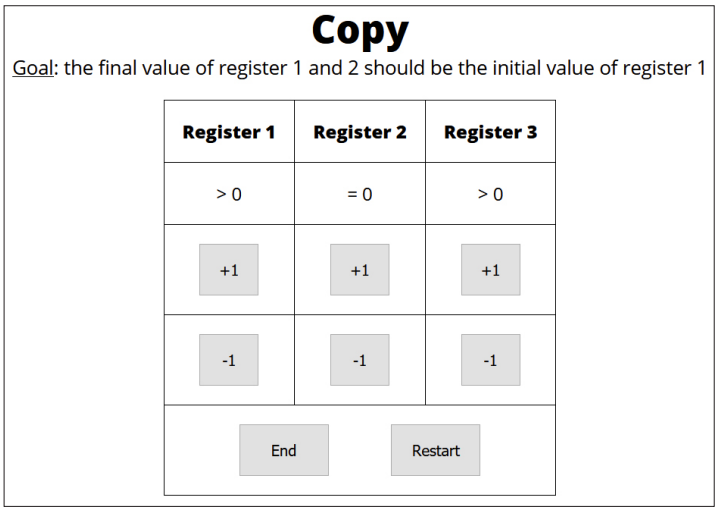

Fig. 1: Machine-computer game interface for the task copy on a 3-counter machine

A stack machine with k registers over an alphabet Σ can hold a string over Σ (including the empty string ε) in each register; a machine state is a sequence of k such strings. For each register it has an operation to remove the last character or append a given character from Σ; removing the last character from an empty string has no effect. For each register and each character x from Σ it has a predicate which holds iff x is the last character of the string in that register.

In the following, sequences of training tasks for the counter and stack machine are given. We write Ri and Ri' for the initial and final value of the i-th register, respectively. For a string x let |x| denote its length. The right column contains tasks for counter machines and the left one for stack machines over the alphabet {A, B, C}. The numbers in parentheses after the task names indicate the available number of registers.

- Reverse (2): R2′ = the reverse of R1
- Move (3): R2′ = R1
- Copy (3): R1′ = R2′ = R1
- Concat (3): R3′ = R1 concatenated with R2
- Repeat (4): R3′ = R1 concatenated with itself |R2| times
- Substr (4): R4′ = substring of R1 starting from the (|R2|+1)-th character with length |R3|
- Binary (4): R2′ = x times A where x is the value of the binary number in R1 (A=1, B=0, last character is LSB)

- Move (2): R2′ = R1
- Copy (3): R1′ = R2′ = R1
- Add (3): R3′ = R1 + R2
- Mult (4): R3′ = R1 · R2
- Div (4): R1 · R3′ + R4′ = R2
- Parity (1): R1′ = R1 mod 2
- Prime (6): R2′ = 1 if R1 is prime, 0 otherwise

For the task Substr the initial machine state must be chosen such that it describes a valid substring, i.e. |R2| + |R3| ≤ |R1|. The game for these tasks can be played at [Ch19a].

We recommend starting with a task like the gummy bear factory that has a concrete scenario before introducing the abstract counter machine. The task Parity is interesting because one cannot infer what the computer does simply by observing them play

the game. It could be either that they count the number and determine its parity (illegal) or they keep track of the parity. This becomes visible when the algorithm has to be formalized. The task PRIME is a bit more difficult and can be given as optional task. The stack machine can be introduced as a generalization of the counter machine: a counter machine is a stack machine with unary alphabet.

3 Machine Programs

The complexity of general-purpose programming languages can make it difficult for a learner to grasp the basics of programming as they struggle with language-specific details. Mini languages such as Karel the Robot or the drawing turtle try to solve these difficulties by reducing the language to a bare minimum and are suggested for programming courses for non-CS students by Brusilovsky et al [BSS04].

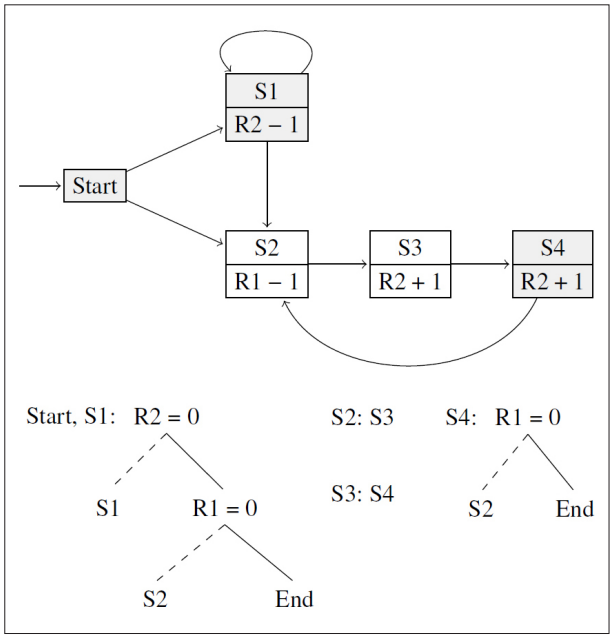

Fig. 2: Machine program for a counter machine computing R2′ = 2 · R1

Program State	Operation	R1	R2	Predicate Sequence
Start		2	1	R2 = 0 ⇢ S1
S1	R2 − 1		0	R2 = 0 → R1 = 0 ⇢ S2
S2	R1 − 1	1		S3
S3	R2 + 1		1	S4
S4	R2 + 1		2	R1 = 0 ⇢ S2
S2	R1 − 1	0		S3
S3	R2 + 1		3	S4
S4	R2 + 1		4	R1 = 0 → End
End				

Tab. 1: Execution trace of program from Fig. 2

Machine programs constitute a mini language and are a generalized form of finite state automata for arbitrary models of computation. A machine program consists of a set of (program) states of which each is associated with an operation that is executed when that state is reached and of a special state *Start*. After a state has been visited, the program needs to decide with what state to continue or whether to halt. This decision process is encoded by a binary decision tree (BDT). Each inner node of such a tree is a predicate. If the predicate holds, the right (non-dashed) subtree is taken, otherwise the left (dashed line). The leaves are program states or *End* to indicate end of execution.

Figure 2 shows such a program and Table 1 is the trace it produces for the input (2, 1). An empty cell in a register column indicates that the value has not changed. For instance, the first row begins with the start state and register contents (2, 1). Then the BDT of Start is checked. The predicate at the root node is R2 = 0 and therefore the program must check whether this holds. Since R2 contains 1 this is false and thus the left (dashed) branch is taken, which leads to the leave with S1. Therefore the program continues execution with the program state S1 in the second row.

While Figure 2 also shows the control flow graph (CFG), the set of states along with their operations and BDTs are already a complete description of the program. A gray state indicates that the program's execution can end there.

While it is possible to describe algorithms for models of computation as a sequence of instructions (see [Sc67]) or with conventional control structures, we opted for this particular representation due to the tight correspondence between machine programs and traces which is the basis for the programming method described below.

A fundamental principle of programming is to construct complex programs from simpler ones using an abstraction mechanism such as functions. Subprogram calls are one such mechanism that can be easily integrated into machine programs. Suppose P is a machine program for the model of computation M. P can be treated as if it were an operation of M because it takes a machine state as input and returns one (assuming it always halts). Thus, if one writes a new program P' for M, a program state s of P' can be associated with P meaning: execute P when s is visited and then continue with the BDT of s. There should not be any cycles in the subprogram calls such as P' calling P and vice versa.

Programming Method.
The last column in Table 1 describes the path that was taken in the BDT of the program state in that row. For example, in the second row the machine state is (2, 0) after executing the operation R2 − 1. Thus, in the BDT of S1 the predicate R2 = 0 is checked first (true) and then R1 = 0 (false), which leads to S2. This trace table can be used to derive a partial version of the program in Figure 2.

The program states and their operations can be directly read from the table. The BDT of a program state can be reconstructed by combining the paths from the predicate sequences. For example, for S4 we combine the paths 'R1 = 0 ⇢ S2' and 'R1 = 0 → End'. Whenever an inner node in a reconstructed BDT has only one child, we add ; as other child which represents a program state with undefined behavior. Applying this procedure leads to the partial program shown in Figure 3. The program state ∅ has *End* as BDT and an arbitrary operation can be assigned to it (represented by in the CFG). Further traces can be added and incorporated until the program is complete (no more undefined behavior).

This method has a consistency guarantee. Suppose that we want to reconstruct a program P from some of its traces T. Let P' be the partial program derived from T as described above. Then P' will either behave identically to P or terminate in the undefined state ∅ In our example this means the partial program in Figure 3 behaves identically to the complete one for all inputs (x, y) such that $x \geq 1$ and $y = 1$.

The previous example only served to show the correspondence between machine programs and their traces. But since the goal is to construct a program from scratch this begs the question of how to obtain trace tables without a program? Playing the machine-computer game yields a trace containing the operations and machine states. The learner then has to fill in the column for the program states manually. Whenever two rows have different operations they cannot be assigned the same program state. The converse is not necessarily true, i.e. in Table 1 the fourth and fifth row have the same operation but different program states. The predicate sequences can also be filled in manually but this seems to be too complicated in practice. Alternatively, the machine-computer game can be slightly modified to yield these sequences as well. The predicates are hidden from the computer. In order to see one the computer has to ask the machine. The order in which the computer asks to see the predicates is the predicate sequence. After an operation is executed all predicates are hidden from the computer again. The key challenge for the learner is to understand under what circumstances two rows correspond to the same program state. After obtaining such a table the rest of this procedure can be automated.

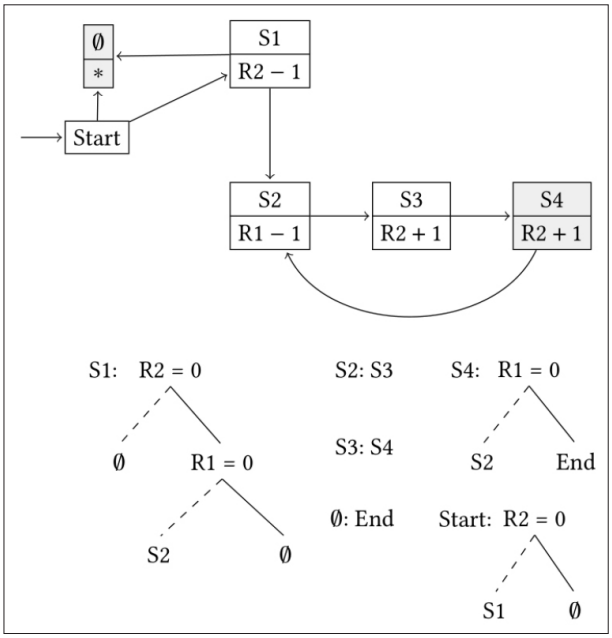

Fig. 3: Partial program constructed from Tab. 1

A less rigid way of applying this method is to leave out the predicate sequences. In this case the traces are only used to construct the CFG since the sequence of program states in a trace corresponds to a path through the CFG. Then the BDTs are developed separately knowing which states must occur as leaves from the CFG. Manually constructing BDTs makes it easier to exploit repeating subtrees in them. A hybrid approach could be to partially determine a BDT using a few predicate sequences and then complete it manually.

4 First Experiences

We carried out a trial run with two 10th graders (Gymnasialschüler, both age 16) over 5 days for 3 hours per day. Neither of them had previous experience with programming and both performed very well in math as attested by their math teacher. Moreover, the trial run was held during school vacation without any kind of reward except a certificate of participation implying both had a high intrinsic motivation.We prepared a collection of worksheets based on the tasks described in Section 2, which they had to solve by themselves.We only engaged if there was a conceptual misunderstanding, the description of a problem was unclear or to verify solutions, otherwise both of them worked independently on the worksheets.

The 1st worksheet presented the gummy bear factory problem and another similar one. To describe their solution for the tasks they were given trace tables with columns for the operation and machine state only. The first row in these tables was already given and they had to fill out the other rows to demonstrate how their algorithm would proceed on that input. The 2nd worksheet contained a description of counter machines and the tasks for them (except Div and Prime). Again, they were given trace tables to demonstrate their solutions. Additionally, they were also given the option to play the machine-compute game for the tasks. The 3rd worksheet contained a description of machine programs along with the program from Figure 2 and the trace from Table 1. They had to fill out two trace tables with predetermined inputs for this program to see whether they understood the semantics. The other

problem was to reconstruct a machine program from a given trace. The purpose here was to make them aware of the connection between traces and machine programs. The last problem was to implement the algorithms that they found for the previous worksheet as machine programs. The 4th and 5th worksheet described the stack machine and its tasks. The first problem was to find algorithms and demonstrate them using trace tables again. The second problem was to implement them. For the task Binary it contained an explanation of how a binary number can be converted by summing the appropriate powers of two.

At the end of the second day both had already found and implemented algorithms for all tasks up to Mult. Both were able to find algorithms for all of the tasks without help. The only exception was the task Binary for which one student didn't find an algorithm but was able to develop one with the help of an example trace that we gave them. Before the workshop ended one student wrote correct machine programs for all tasks except Concat and Binary and the other one for all tasks except Repeat. The programs that they had to write were quite large. For example, both their programs for Substr had 27 program states and over 60 nodes in the BDTs.

They were given laptops and an online interpreter to write machine programs (see [Ch19a]). At first, they tried to implement their algorithms ad hoc, i.e. directly typing something into the interpreter. While this worked for the small programs up to Add, they both struggled to implement their algorithms for Mult and the first tasks on the stack machine. Whenever they thought that they fixed a problem in their program by making some modification, they seemed to introduce another one. After realizing this, they switched to using our programming method and wrote their programs using pen and paper. When they copied their finished programs, the programs were either immediately correct or had a minor bug which they were able to quickly identify using the traces produced by the interpreter and resolved it quickly. We consider this to be a positive indicator for our programming method's utility.

The students did not use the column for the predicate sequences when they applied the mehtod. Instead, they only filled out the program state column and used this information to derive a partial CFG, which then was used as basis to construct the BDTs. It seems that a less rigid way of applying the method was more intuitive and useful to them. Additionally, the machine programs for the tasks on the stack machine contained many recurring subtrees in the BDTs; constructing those using predicate sequences would be needlessly repetitive. Since they were constructing the BDTs separately, they were able to recognize these common patterns and exploit them.

For the algorithmic part, the concept with the machine-computer game and writing traces to demonstrate their algorithms worked well in practice. Even before introducing machine programs both seemed to have grasped what constitutes an algorithm in the context of a model of computation. Alternatively, prompting them to play the machine-computer game for a particular task to check whether they have found a correct algorithm also worked well.

5 Outlook & Discussion

From our experiences with the trial run it seems that our approach and the difficulty of the tasks are suitable for a university-level programming course for non-CS students or maybe as an optional intro course for CS students who haven't started their first semester yet (Vorbereitungsveranstaltung). The learning unit outlined here will be implemented and evaluated in a CS course that is part of a preparatory college (Studienkolleg). Moreover, the worksheets and software are being refined to make them publicly available.

A more sophisticated variant of the programming method presented here that can be used to implement complex algorithms has been tried out in an advanced programming course for CS students and has been met with positive feedback from the students. Our goal is to extend the contents of the concept presented here such that it connects to this advanced method in which algorithms are expressed in a similar manner to machine programs.

Lastly, we want to briefly address the question of evaluating the effectiveness of a programming method. Programming in practice requires various skills such as a good understanding of the programming language in use and the ability to select adequate frameworks and libraries to accomplish a given task. One constant, inevitable aspect of programming, though, is the ability to formalize a given algorithm in some formal language. This ability can be tested in the context of models of computations (MoC) as follows.

The test subject gets a description of a MoC and an algorithm for it along with some example traces. The task is to implement this algorithm in a programming language of their choice such that the implementation is consistent with the example traces. It should be verified that the test subject has correctly understood the algorithm. For example, in an OOPL a MoC could be modeled as a class C whose methods represent the operations and predicates. Whenever an operation is called the operation name and the

machine state after applying that operation is printed. Then, implementing the given algorithm means to write a program that only uses a single variable whose type is C such that the traces it produces are identical to the given example traces.

This type of evaluation measures the ability to formalize a given algorithm in a precise sense in a way that prevents the test subject from implementing some 'similar' algorithm that might be easier to implement. This differs from the common testing method where an algorithmic problem is given and the task is to write a program which solves it. Often, efficient algorithms are more challenging to implement than their naive counterparts. Therefore the ability to formalize complex algorithms is desirable but difficult to assess with the common testing method.

Bibliography

[ADF19] Adam, Michel; Daoud, Moncef; Frison, Patrice: Direct Manipulation versus Text-based Programming: An experiment report. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, Aberdeen, Scotland, UK, July 15-17, 2019. pp. 353–359, 2019.

[BSS04] Brusilovsky, Peter; Shcherbinina, Olena; Sosnovsky, Sergey A.: Mini-languages for noncomputer science majors: what are the benefits? Interact. Technol. Smart Educ., 1(1):21–28, 2004.

[Ch19a] Chandoo, Maurice: , https://upsl.uber.space/aws19/info.txt, 2019.

[Ch19b] Chandoo, Maurice: A Systematic Approach to Programming. 2019.

[dB86] du Boulay, Benedict: Some Difficulties of Learning to Program. J. Educational Computing Research, 2(1), 1986.

[DvK10] Desel, Jörg; von Klenze, Leo: AMSEL - ein Lernsystem zum Algorithmenentwurf. In (Kerres, Michael; Ojstersek, Nadine; Schroeder, Ulrik; Hoppe, Ulrich, eds): DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V., 12.-15. September 2010, Universität Duisburg-Essen. volume P-169 of LNI. GI, pp. 33–44, 2010.

[FM10] Futschek, Gerald; Moschitz, Julia: Developing Algorithmic Thinking by Inventing and Playing Algorithms. In: Constructionism, Paris. 2010.

[Fu06] Futschek, Gerald: Algorithmic Thinking: The Key for Understanding Computer Science. In: Informatics Education - The Bridge between Using and Understanding Computers, International Conference in Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2006, Vilnius, Lithuania, November 7-11, 2006, Proceedings. pp. 159–168, 2006.

[HJ13] Hertz, Matthew; Jump, Maria: Trace-based teaching in early programming courses. In: The 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13, Denver, CO, USA, March 6-9, 2013. pp. 561–566, 2013.

[HLR19] Hilton, Andrew D.; Lipp, Genevieve M.; Rodger, Susan H.: Translation from Problem to Code in Seven Steps. In: Proceedings of the ACM Conference on Global Computing Education. CompEd '19, ACM, New York, NY, USA, pp. 78–84, 2019.

[La18] Lamport, Leslie: If You're Not Writing a Program, Don't Use a Programming Language. Bulletin of the EATCS, 125, 2018.

[RNH00] Reichert, Raimond; Nievergelt, Jürg; Hartmann, Werner: Ein spielerischer Einstieg in die Programmierung mit Java, Kara to Java - erste Schritte beim Programmieren. Inform. Spektrum, 23(5):309–315, 2000.

[Sc67] Scott, Dana S.: Some Definitional Suggestions for Automata Theory. J. Comput. Syst. Sci., 1(2):187–212, 1967.

[So13] Sorva, Juha: Notional machines and introductory programming education. TOCE, 13(2):8:1–8:31, 2013.

[Wi06] Wing, Jeannette: Computational Thinking. Communications of the ACM, 49:33–35, 03 2006.

Forschendes Lernen im Bachelor-Seminar „Software Engineering“

Leif Bonorden[1]

Abstract:
Forschendes Lernen ist eine Lehr-Lernform, in der Studierende einen eigenen Forschungsprozess vollständig durchlaufen. In Informatik-Studiengängen und insbesondere in Informatik-Bachelorstudiengängen ist die Forschungsorientierung allerdings nur gering ausgeprägt: Forschendes Lernen wird kaum eingesetzt, obwohl dies möglich und sinnvoll ist. Dieser Artikel stellt ein Konzept für ein Seminar *Software Engineering* im Bachelorstudium vor und beschreibt dessen Durchführung. Abschließend wird das Konzept diskutiert und sowohl aus Studierenden- als auch aus Lehrendensicht positiv evaluiert.

Keywords:
Forschendes Lernen; studentische Forschung; Seminarkonzept; Bachelorstudium; wissenschaftliches Arbeiten; wissenschaftliches Schreiben

1 Einführung

Neben der Vermittlung bekannter Kenntnisse und Methoden soll ein Studium auch zum wissenschaftlichen Arbeiten hinführen. Während Masterstudiengänge der Informatik häufig eine Forschungsorientierung aufweisen, nehmen forschende Tätigkeiten im Bachelorstudium meist nur eine untergeordnete Rolle ein.

Die Lehr-Lernform des forschenden Lernens ermöglicht es Studierenden, einen vollständigen Forschungsprozess zu durchlaufen, und ist insbesondere auch in Bachelorstudiengängen anwendbar. An der Universität Hamburg wurde ein Seminar *Software Engineering* des Bachelorstudiums in der Lehr-Lernform des Forschendes Lernens konzipiert und durchgeführt. Im Folgenden wird in das Forschende Lernen eingeführt und anschließend werden das Seminarkonzept sowie die Durchführung beschrieben und evaluiert.

2 Forschendes Lernen

2.1 Definition und Abgrenzung

Das *Forschende Lernen* ist eine Lehr-Lernform, die in ihrer heutigen Idee seit ca. 50 Jahren diskutiert wird [Bu70]. Eine moderne Definition des Begriffes stammt von Huber [Hu09]:

> Forschendes Lernen zeichnet sich vor anderen Lernformen dadurch aus, dass die Lernenden den Prozess eines Forschungsvorhabens, das auf die Gewinnung von auch für Dritte interessanten Erkenntnissen gerichtet ist, in seinen wesentlichen Phasen – von der Entwicklung der Fragen und Hypothesen über die Wahl und Ausführung der Methoden bis zur Prüfung und Darstellung der Ergebnisse in selbstständiger Arbeit oder in aktiver Mitarbeit in einem übergreifenden Projekt – (mit)gestalten, erfahren und reflektieren.

1 Universität Hamburg, MIN-Fakultät, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Deutschland, leif.bonorden@uni-hamburg.de

Die idealtypischen Phasen eines solchen (allgemeinen) Forschungsprozesses werden von Huber und Reinmann detaillierter beschrieben [HR19]:

1. Wahrnehmen eines Ausgangsproblems oder Rahmenthemas (Hinführung)
2. Finden einer Fragestellung, Definition des Problems
3. Erarbeiten von Informationen und theoretischen Zugängen (Forschungslage)
4. Auswahl von und Erwerb von Kenntnissen über Methoden
5. Entwickeln eines Forschungsplans bzw. Untersuchungsdesigns
6. Durchführung einer forschenden Tätigkeit
7. Erarbeitung und Präsentation der Ergebnisse
8. Reflexion des gesamten Prozesses

Abzugrenzen ist das *Forschende Lernen* gegenüber anderen Lernformen [HR19; Hu09]: Im *Forschungsnahen Lernen* werden Studierende an einen Forschungsprozess herangeführt, wobei aber Vorstellung und Diskussion von Forschung anderer im Vordergrund stehen und die Studierenden nur wenige Teile des Prozesses selbst ausführen oder simulieren. Auch *Problemorientiertes Lernen* weist einige Gemeinsamkeiten auf, jedoch ist hier das Problem bzw. die Fragestellung nicht selbstgewählt und es werden im Allgemeinen keine auch für Dritte interessante Erkenntnisse angestrebt. Ein *Projektstudium* im engeren Sinne fokussiert ebenfalls ein anderes Ziel: die Entwicklung eines Produktes oder ein anderes konkretes Ergebnis – im Software Engineering häufig ein Software-System.

2.2 Gründe und Ziele

Forschendes Lernen folgt den Grundsätzen *Bildung durch Wissenschaft* sowie *Einheit von Lehre und Forschung*, vermittelt allgemeine (und insbesondere auch berufsbezogene) Kompetenzen und ist didaktisch fundiert [Eu05; HR19; Hu09]. Huber postuliert, dass Forschendes Lernen zu einem wissenschaftlichen Studium gehört: Forschendes Lernen sei mehr als ein „didaktischer Trick", mache Wissenschaft als sozialen Prozess erfahrbar, und eigne sich insbesondere auch für Bachelorstudiengänge [Hu04].

Lübcke und Heudorfer beschreiben Ziele Forschenden Lernens auf mehreren Abstraktionsebenen [LH19]. Als Detailziele werden insbesondere Berücksichtigung studentischer Interessen, Wecken von Neugier, Anknüpfung an bestehende Forschung, Erwerb von Methoden- und Schreibkenntnissen, sowie Erkennen von Zusammenhängen zwischen Studieninhalten identifiziert.

2.3 Forschendes Lernen in der Informatik

In der Informatik und nahen Disziplinen scheint Forschendes Lernen weniger verbreitet zu sein – insbesondere im Bachelorstudium. Eine Ursache liegt sicherlich in der ursprünglichen Konzeption von Bachelor- und Masterstudiengängen für die Informatik im Zuge der Bologna-Reform, bei der Informatik-Bachelorstudiengänge „grundlagen- und methodenorientiert" seien und „Grundlagen des Faches in der Breite" legen sollten. Erst die Informatik-Masterstudiengänge sollten „forschungsorientiert" gestaltet werden [Fa04]. Auch wenn schon vier Jahre später die Empfehlung folgte, „bereits die Bachelor-Studiengänge explizit als ‚stärker forschungsorientiert' zu konzipieren" [Fa08], steht dies bis heute selten im Vordergrund und die Bachelorarbeit ist meist die einzige Studienleistung, bei der ein Forschungszyklus von Fragestellung bis Ergebnispräsentation durchlaufen wird.

Als stark strukturierte Disziplin sieht die Informatik zudem einen im Vergleich zu anderen Disziplinen großen Fachkanon von Grundlagenkenntnissen und -methodik für den Beginn des Studiums vor. Zusätzlich ist auch die Employability, also der Erwerb von berufsrelevanten Kenntnissen und Fähigkeiten, stark fokussiert – insbesondere für Tätigkeiten in der Softwareentwicklung.

Im Folgenden werden einige Veröffentlichungen zum Forschenden Lernen in der Informatik kurz vorgestellt – zunächst in der allgemeineren Disziplin, schließlich konkret im Software Engineering:

Das Projekt FOLASMART hat im Lehramtsstudium der Fächer Mathematik und Informatik ein Projekt für die Entwicklung von Mathematik-Lernapps konzipiert. Für die App-Gestaltung haben die Studierenden dabei zu didaktischen Fragestellungen geforscht. [RE12]

An der Universität Bremen bietet der Masterstudiengang *Systems Engineering* eine Studienrichtung *Forschungsvertiefung* an, in der Studierende besonders für wissenschaftliche Tätigkeiten ausgebildet werden. Forschendes Lernen wird dabei als Ansatz für die Studiengangsgestaltung verwendet. [Ba20]

Ein interdisziplinäres Seminar der Informatik und Soziologie wurde von Knoth, Lucke und Zifonun konzipiert und durchgeführt. Studierende haben sich in Tandems zunächst mit den Methoden der beiden Disziplinen auseinandergesetzt und anschließend interdisziplinäre Forschungsanträge erarbeitet. [KLZ15]

Mottok et al. haben interne (Forschungs-)Konferenzen für ein Modul *Research Methods and Strategies* im Master-Schwerpunkt *Software Engineering* konzipiert. Dabei werden Einreichungen reviewt und entweder als Vortrag zugelassen oder andernfalls als Poster präsentiert. [Mo12]

Als Ergänzung zu einer Vorlesung zum Thema *Software-Architektur* nutzt Gast Forschendes Lernen im Rahmen kleinerer Projekte. Der Fokus liegt dabei auf praxisrelevanter Forschung und Handlungskompetenz. [Ga07]

An der Universität Paderborn wurde Forschendes Lernen in ein Praxisprojekt über zwei Semester integriert. Studierende haben hier den im Projekt selbst eingesetzten Softwareentwicklungsprozess beforscht. [SOF18]

An der Hochschule Heilbronn haben Studierende im ersten Semester des Masterstudiengangs Software *Engineering and Management* aus gegebenen Forschungsthemen eines ausgewählt, bearbeitet und in Form eines möglichst publizierbaren Artikels präsentiert. [HH13]

3 Kontext

Das Modul *Seminar* liegt im Pflichtbereich mehrerer Informatik-Studiengänge und ist für das 5. oder 6. Fachsemester vorgesehen. Die Studierenden können in jedem Semester zwischen 5 bis 10 verschiedenen Lehrveranstaltungen wählen, um diese Pflicht zu erfüllen. Laut Modulbeschreibung umfassen die angestrebten Lernergebnisse „die Fähigkeit zur wissenschaftlichen Recherche und zur Präsentation wissenschaftlicher Erkenntnisse", insbesondere sollen „die Studierenden bereits im Bachelor-Studiengang in Kontakt mit Forschungsfragen und Forschungsmethodik der Informatik" kommen. Nach dem Wissen des Autors wurde bisher keine dieser Lehrveranstaltungen in Form des Forschenden Lernens angeboten.

Das Modul umfasst 3 LP nach ECTS, wobei für die Präsenz- bzw. synchronen Anteile 14 Wochen mit je 2 Stunden Arbeitsaufwand berechnet werden, und Selbststudium und Prüfungsvorbereitung 62 Stunden umfassen. Die Prüfungsleistung besteht aus Referat und schriftlicher Ausarbeitung. Zusätzliche Studienleistung ist die „aktive Teilnahme".

Formale Teilnahmevoraussetzung ist der Abschluss eines Proseminars – vorgesehen für das 2. oder 3. Fachsemester –, in dem Studierende „Schlüsselqualifikationen im Bereich des selbstständigen Recherchierens, Strukturierens, Präsentierens und Moderierens" erlangen. Zudem sollten Studierende bereits ungefähr ein Drittel ihres Studiums (ohne Bachelorarbeit) oder mehr absolviert haben.

4 Gestaltung der Lehrveranstaltung

Angelehnt an [So17] besteht die Lehrveranstaltung aus drei grundsätzlichen Phasen: Einstieg, Forschungsphase, sowie Abschluss und Nachbereitung. Der Einstieg nimmt dabei etwa die Hälfte der 14-wöchigen Vorlesungszeit ein; die Forschungsphase beginnt bereits parallel zum Einstieg.

4.1 Einstieg

Die wöchentlichen 90-minütigen Termine in der Einstiegsphase sind als jeweils thematisch abgeschlossene Workshops konzipiert: Nach einem kurzen lehrendengeleiteten Auftakt erfolgt die Bearbeitung und Diskussion des jeweiligen Themas durch die Studie-

renden. Die angeleiteten studentischen Aktivitäten beschränken sich – mit einer Ausnahme – auf die Präsenz- bzw. synchronen Termine. Für ein vertiefendes Selbststudium werden Material und Anregungen zur Verfügung gestellt.

Einführung & Organisation
Im ersten Workshop werden verschiedene Sichtweisen auf Softwareentwicklung diskutiert: Softwareentwicklung als Handwerk, als Strukturwissenschaft, als Ingenieursdisziplin, als Kunst. Zudem werden mögliche Themenbereiche für die Forschungsvorhaben vorgestellt und Rahmenbedingungen besprochen.

Schreibprozess
Der Forschungs- bzw. Schreibprozess als Ganzes steht im Vordergrund: von Themeneingrenzung über Strukturierung und Arbeit an konkretem Material zu Textentwürfen und -überarbeitung. Dieser Workshop findet in Kooperation mit dem Schreibzentrum der Universität statt.

Forschung
Typische Forschungsfragen des Software Engineering [Sh03], der Methodenreichtum der Disziplin [Ra21; SF18; Wi14] sowie Veröffentlichungsarten für Ergebnisse (insbesondere Konferenz, Journal, Workshop) werden diskutiert. Die Studierenden identifizieren und vergleichen diese Elemente an Beispielen [FMP19; LR19; Mu18; MW15; Po16; Sa19; Ts18].

Literatur & Recherche
Die Studierenden erproben eine Lesemethode an einem der Beispielartikel. Anschließend werden Bibliotheksangebote, Suchmaschinen und Datenbanken sowie Vorgehensweisen (z.B. Snowballing, Bewertung von Quellen) für die Literaturrecherche vorgestellt. Die Studierenden führen eine kurze Recherche zu ihrem Forschungsthema durch.

Schriftliche Arbeiten
Dieser Workshop ist zweigeteilt: Im ersten Teil geht es um wissenschaftlichen Schreibstil, Methoden zur Textüberarbeitung und Schreibtypen. Im zweiten Teil erarbeiten die Studierenden anhand der Beispielartikel eine typische Artikelstruktur für das Software Engineering. Zudem werden Zitieren und Plagiate thematisiert. Der erste Teil findet in Kooperation mit dem Schreibzentrum der Universität statt.

Vorträge & Präsentationen
Die Studierenden sehen sich in Vorbereitung auf diesen Workshop Videos von verschiedenen Vorträgen an: mehrere wissenschaftliche Vorträge, aber auch einen TED-Talk, einen Lightning Talk auf einer Entwicklerkonferenz sowie eine politische Rede. Die Studierenden diskutieren die grundsätzlichen Elemente eines guten Vortrags und grenzen wissenschaftliche Vorträge gegenüber anderen Formen ab.

Teaser & Abstracts
Mit der „five-element organization" [Zo15] verfassen die Studierenden erste Entwürfe für einen Abstract ihrer schriftlichen Ausarbeitung. Zudem stellen Sie den aktuellen Stand ihres Forschungsvorhabens in einem 1–2-minütigen Teaser-Vortrag vor.

4.2 Forschungsphase

In den ersten Terminen der Lehrveranstaltung werden mögliche Themenbereiche für Forschungsvorhaben vorgestellt und von den Studierenden ausgewählt. Die Studierenden können weitere Themenbereiche vorschlagen.

Die Studierenden schränken ihren Themenbereich weiter ein, recherchieren geeignete Literatur und formulieren eine Fragestellung für ihr Forschungsvorhaben. Sie wählen eine geeignete Forschungsmethode und führen das Vorhaben allein oder zu zweit durch. Die Ergebnisse bereiten sie in Form eines Vortrags sowie als wissenschaftlichen Artikel im Format der *Lecture Notes in Informatics* der Gesellschaft für Informatik auf.

Spätestens nach dem vierten Seminartermin (Thema *Literatur & Recherche)* sind die Studierenden in der Lage, zu ihrem Thema zu recherchieren, erste Kandidaten für Forschungsfragen zu formulieren und geeignete Forschungsmethoden zu diskutieren. Spätestens nach dem Thema *Schriftliche Arbeiten* können die Studierenden die schriftliche Ausarbeitung zu ihrem Thema verfassen.

Lehrendenfeedback und weitere Absprachen erfolgen auf Studierendeninitiative. Feste Zwischenpräsentationen oder andere Rückkopplungen sind nicht vorgesehen.

4.3 Abschluss und Nachbereitung

Zum Abschluss findet eine Abschlussveranstaltung mit Vorträgen und Diskussionen statt – angelehnt an die Form einer wissenschaftlichen Tagung. Diese wird als Blockveranstaltung in der vorlesungsfreien Zeit veranstaltet. Die schriftlichen Forschungsergebnisse werden in Form von Tagungsproceedings zusammengefasst.

Im Rahmen der Abschlussveranstaltung wird auch die Lehrveranstaltung mithilfe eines Evaluationsfragebogens und im offenen Gespräch ausgewertet. Die Studierenden werden zudem aufgefordert, ihren individuellen Forschungs- und Lernprozess schriftlich zu reflektieren, insbesondere erfolgreiche und kritische Aspekte ihrer eigenen Forschung zu benennen sowie offene Fragen zu identifizieren. Auf Wunsch kann die schriftliche Reflexion eingereicht werden und erhält zusätzliches Lehrendenfeedback.

5 Durchführung der Lehrveranstaltung

Die Lehrveranstaltung wurde im Sommersemester 2020 mit 9 Studierenden sowie im Wintersemester 2020/2021 mit 20 Studierenden durchgeführt. Die Studierenden befanden sich mindestens im 4. Fachsemester, viele kurz vor Abschluss ihres Bachelorstudiums. Im Kontext der Corona-Pandemie fanden alle synchronen Termine via Zoom statt, weitere Elemente für die gemeinsame Arbeit waren in der Lernplattform Moodle eingebunden. Die Abschlussveranstaltung fand nicht als Blockveranstaltung, sondern über mehrere Wochen am Ende der Vorlesungszeit statt.

Im Sommersemester 2020 wurden 2 Themen zu zweit bearbeitet, 5 Themen allein. Die Ausarbeitung haben 7 Studierende in deutscher Sprache verfasst, 2 in englischer. Die Leistungen waren – bis auf eine Ausnahme – im guten bis sehr guten Bereich.

Im Wintersemester 2020/2021 wurden 7 Themen zu zweit bearbeitet, 6 Themen allein. Die Ausarbeitung haben 14 Studierende in deutscher Sprache verfasst, 5 in englischer; zu einem Thema erfolgte keine Abgabe. Die Leistungen waren im guten bis sehr guten Bereich.

5.1 Themenbereiche

Die vorgeschlagenen Themenbereiche kamen in beiden Semestern aus drei Bereichen:

Klassische Untersuchungen der Softwaretechnik
Themen, die seit mindestens 20 Jahren erforscht und diskutiert werden, z. B. Modularisierung von Softwaresystemen. Als Einstieg war jeweils eine klassische Veröffentlichung angegeben.

Aktuelle Forschung der Softwaretechnik
Themen, zu denen in den vergangenen zwei Jahren auf großen Konferenzen Beiträge erschienen sind, z. B. Commits in Versionskontrollsystemen. Als Einstieg war jeweils die möglichst aktuelle Veröffentlichung angegeben.

Aktuelle Themen aus der Praxis
Themen, die im *Technology Radar* des Unternehmens ThoughtWorks[2] aufgeführt sind, z. B. Polyglot Programming. Als Einstieg war jeweils der Eintrag im Technology Radar angegeben.

5.2 Beispiele für studentische Arbeiten

Eine Studentin hat die Verwendung von Product Backlogs in Open-Source-Software untersucht. Zunächst wurden verwandte Arbeiten präsentiert und wichtige Ergebnisse zusammengefasst. Anschließend wurde in einer Studie von vier Software-Projekten untersucht, ob Eigenschaften, die Product Backlogs in der Literatur zugeschrieben werden, auch in Open-Source-Projekten zutreffen. Während die meisten Eigenschaften – u. a. keine Design-Dokumente oder Anforderungen; informelles Modell für Kommunikation und Koordination – bestätigt werden konnten, trafen andere – insbesondere bezüglich der Priorisierung von Einträgen – nicht auf alle Projekte zu.

2 www.thoughtworks.com/radar

Zwei Studierende haben Performance-Unterschiede zwischen GraphQL- und REST-Schnittstellen in einem konkreten Setting des Frameworks *Spring Boot* untersucht. Auf Grundlage von Literatur zum Thema wurden geeignete Metriken ausgewählt und Hypothesen für die konkrete Situation aufgestellt, u. a. wurde eine geringere CPU-Zeit bei der Nutzung von GraphQL sowohl für lesende als auch für schreibende Zugriffe erwartet. In einem Experiment konnte dies für lesenden Zugriff bestätigt werden, musste aber für schreibenden Zugriff verworfen werden.

6 Evaluation

Im Rahmen des Seminars durchlaufen die Studierenden alle in 2.1 genannten Phasen des Forschungsprozesses. Für die erste sowie die achte Phase sind Vorgaben und Steuerung durch Lehrende stärker als für die übrigen Phasen. Die in 2.2 genannten (Detail-) Ziele des Forschenden Lernens werden erreicht.

Das Seminarkonzept lässt sich insbesondere entlang von drei Gestaltungsfeldern analysieren [HR19; LRH19]. **Forschungscharakter**: Die studentische Forschung ist disziplinär verankert und erfolgt unabhängig von Forschungsaktivitäten der Lehrenden und der Institution. Die Ergebnisse werden primär intern präsentiert und diskutiert. **Autonomie**: Die Studierenden wählen Forschungsthema und Forschungsmethode innerhalb eines Rahmens selbst und werden bei Planung und Durchführung nur bedarfsorientiert begleitet. **Soziale Eingebundenheit**: Die Studierenden führen ihre Forschungsprojekte allein oder zu zweit durch. Feedback der Lehrenden kann selbstständig bei Bedarf eingeholt werden, eine abschließende Reflexion ist angeleitet.

Alle Studierenden bewerten das Seminar sowohl im offenen Feedbackgespräch als auch im anonymen Evaluationsfragebogen insgesamt positiv. Hervorgehoben werden die vielfältigen Themenvorschläge, die gemeinsam bearbeiteten Themen im Einstieg, die Mitwirkung des Schreibzentrums sowie die Vorbereitung auf eine Bachelorarbeit. Verbesserungspotential sehen die Studierenden vor allem im Lehrendenfeedback: Zu aufwändige Forschungsvorhaben sollten frühzeitig identifiziert und angepasst werden. Zudem wird ein hoher Arbeitsumfang zwar angemerkt, jedoch kaum negativ gesehen. Die in der Evaluation angegebenen Arbeitsaufwände liegen im Rahmen des Umfangs von 3 LP nach ECTS.

Aus Lehrendenperspektive wird das Seminar ebenfalls überaus positiv bewertet. Ein initial hoher Aufwand für die Konzeption der Lehrveranstaltung ermöglicht erfreuliche, gewinnbringende Seminararbeit und gute Arbeitsergebnisse. Der individuelle Betreuungsaufwand außerhalb der synchronen Termine blieb gering. Ein direkter Vergleich zu anderen Seminaren erscheint nicht sinnvoll, aber im subjektiven Eindruck verstehen die Studierenden die Disziplin *Software Engineering* besser als in der Vorgängerveranstaltung in anderer Lehr-Lernform. Insbesondere bei anschließenden Gesprächen über mögliche Bachelorarbeiten zeigen Teilnehmende des Seminars ein besseres Verständnis für Forschung, Fragestellungen und Themen des Software Engineering als Studierende, die nicht an diesem Seminar teilgenommen haben. Ergebnisse solcher Bachelorarbeiten liegen noch nicht vor.

Die Gestaltung hinsichtlich des Forschungscharakters – Verzicht auf Interdisziplinarität, Unabhängigkeit von Forschung anderer, Fokus auf interne Ergebnisvorstellung – scheint für die Situation der Studierenden im Bachelorstudium hilfreich zu sein. Die umfangreiche Einstiegsphase hat den Studierenden einen guten Umgang mit der weitgehenden Autonomie ermöglicht. In Hinsicht auf die soziale Eingebundenheit konnten die Studierenden nicht immer mit den Selbstorganisationserfordernissen umgehen; angeleitete Feedback- und Austausch-Prozesse könnten für diese Lehrveranstaltung besser geeignet sein.

Die digitale Durchführung mit Zoom und Moodle ist insgesamt gut verlaufen. Ohne Präsenz-Möglichkeit musste die Abschlussveranstaltung zur Vermeidung sehr langer Videokonferenzen über mehrere Wochen verteilt stattfinden, was sich als akzeptable Lösung herausgestellt hat, auch wenn die Diskussionsbeteiligung gering war.

7 Zusammenfassung und Ausblick

Ein Seminar *Software Engineering* im Bachelorstudium wurde in der Lehr-Lernform des Forschenden Lernens konzipiert und zweimal durchgeführt. Sowohl aus Sicht der Studierenden als auch aus Lehrendenperspektive war das Seminar erfolgreich und wurde positiv evaluiert. Durch eine umfangreiche Einstiegsphase konnen auch bei unterschiedlichen Vorkenntnissen geeignete Grundlagen für eigene Forschungstätigkeit geschaffen werden. Die Studierenden konnten auch im Bachelorstudium gut mit der Autonomie umgehen.

Aus der Analyse entlang der genannten Gestaltungsfelder und der weiteren Evaluation ergeben sich direkte Anpassungsoptionen. Insbesondere sollten Studierende bei der Abschätzung des Aufwandes für ihr Forschungsvorhaben stärker unterstützt werden. Weitere Feedback- und Austausch-Prozesse zwischen Peers oder mit Lehrenden sollten angeleitet werden.

Literatur

[Ba20] Bačić, I.; Rodenhauser, A.; Kuhfuss, B.; Colombi Ciacchi, L.: Forschendes Lernen im Masterstudiengang Systems Engineering – Bausteine erhalten, Bausteine zusammensetzen, Ergebnisse reflektieren. In (Hoffmeister, T.; Koch, H.; Tremp, P., Hrsg.): Forschendes Lernen als Studiengangsprofil. Springer, Wiesbaden, S. 285–300, 2020, isbn: 978-3-658-28824-2.

[Bu70] Bundesassistentenkonferenz, Hrsg.: Forschendes Lernen – Wissenschaftliches Prüfen: Ergebnisse des Ausschusses für Hochschuldidaktik. Bonn, 1970.

[Eu05] Euler, D.: Forschendes Lernen. In (Spoun, S.; Wunderlich, W., Hrsg.): Studienziel Persönlichkeit: Beiträge zum Bildungsauftrag der Universität heute. Campus, Frankfurt a. M., S. 253–271, 2005.

[Fa04] Fakultätentag Informatik, Hrsg.: Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten, 2004.

[Fa08] Fakultätentage der Ingenieurwissenschaften und der Informatik an Universitäten, Hrsg.: Qualifikationsrahmen für Absolventen „stärker forschungsorientierter" Studiengänge und Promovierte in den Ingenieurwissenschaften und der Informatik, 2008.

[FMP19] Fleck, G.; Moser, M.; Pichler, J.: Improving Quality of Data Exchange Files: An Industrial Case Study. In (Franch, X.; Männistö, T.; Martínez-Fernández, S., Hrsg.): Product-Focused Software Process Improvement. Bd. 11915. Lecture Notes in Computer Science, Springer International Publishing, Cham, S. 161–175, 2019.

[Ga07] Gast, H.: Forschendes Lernen am Beispiel derVorlesung »Software-Architektur«. Forschendes Lernen als hochschuldidaktisches Prinzip – Grundlegung und Beispiele, Tübinger Beiträge zur Hochschuldidaktik 3/1, hrsg. von Reiber, K., 2007, issn: 1861-213X.

[HH13] Hesenius, M.; Herzberg, D.: Forschung mit Master-Studierenden im Software Engineering. In (Spillner, A.; Lichter, H., Hrsg.): Tagungsband des 13. Workshops "Software Engineering im Unterricht der Hochschulen". Bd. 956. CEUR Workshop Proceedings, Aachen, S. 115–116, 2013.

[HR19] Huber, L.; Reinmann, G.: Vom forschungsnahen zum forschenden Lernen an Hochschulen: Wege der Bildung durch Wissenschaft. Springer VS, Wiesbaden, 2019, isbn: 978-3-658-24948-9.

[Hu04] Huber, L.: Forschendes Lernen: 10 Thesen zum Verhältnis von Forschung und Lehre aus der Perspektive des Studiums. Die Hochschule : Journal für Wissenschaft und Bildung 13/2, S. 29–49, 2004, issn: 1618-9671.

[Hu09] Huber, L.: Warum Forschendes Lernen nötig und möglich ist. In (Huber, L.; Hellmer, J.; Schneider Friederike, A., Hrsg.): Forschendes Lernen im Studium. Universitätsverlag Webler, Bielefeld, S. 9–35, 2009, isbn: 978-3-937026-66-4.

[KLZ15] Knoth, A.; Lucke, U.; Zifonun, D.: Lehre im Format der Forschung: ein interdisziplinäres Seminarkonzept. In (Nistor, N.; Schirlitz, S., Hrsg.): Digitale Medien und Interdisziplinarität.GMW2015. Bd. 68. Medien in derWissenschaft, Waxmann, Münster, S. 217–227, 2015, isbn: 978-3-8309-3338-0.

[LH19] Lübcke, E.; Heudorfer, A.: Die Ziele forschenden Lernens: Eine empirische Analyse im Rahmen der QPL-Begleitforschung. In (Reinmann, G.; Lübcke, E.; Heudorfer, A., Hrsg.): Forschendes Lernen in der Studieneingangsphase. Springer FachmedienWiesbaden,Wiesbaden, S. 17–58, 2019, isbn: 978-3-658-25311-0.

[LR19] Licker, N.; Rice, A.: Detecting Incorrect Build Rules. In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, Montreal, QC, Canada, S. 1234–1244, 2019.

[LRH19] Lübcke, E.; Reinmann, G.; Heudorfer, A.: Entwicklung eines Instruments zur Analyse forschenden Lernens. In (Reinmann, G.; Lübcke, E.; Heudorfer, A., Hrsg.): Forschendes Lernen in der Studieneingangsphase. Springer, Wiesbaden, S. 127–147, 2019, isbn: 978-3-658-25311-0.

[Mo12] Mottok, J.; Schroll-Decker, I.; Niemetz, M.; Scharfenberg, G.; Hagel, G.: Internal Conferences as a Constructivism Based Learning Arrangement for Research Master Students in Software Engineering. In: Proceedings of the 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering FECS 2012. CSREA, Las Vegas, S. 292–299, 2012, isbn: 978-1-60132-214-2.

[Mu18] Murphy, L.; Kery, M. B.; Alliyu, O.; Macvean, A.; Myers, B. A.: API Designers in the Field: Design Practices and Challenges for Creating Usable APIs. In: 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Lisbon, S. 249–258, 2018.

[MW15] Méndez Fernández, D.; Wagner, S.: Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany. Information and Software Technology 57/, S. 616–643, 2015.

[Po16] Politowski, C.; Fontoura, L.; Petrillo, F.; Guéhéneuc, Y.-G.: Are the old days gone? A Survey on Actual Software Engineering Processes in Video Game Industry. In: Proceedings of the 5th International Workshop on Games and Software Engineering - GAS '16. the 5th International Workshop. ACM Press, Austin, Texas, S. 22–28, 2016.

[Ra21] Ralph, P.; bin Ali, N.; Baltes, S.; Bianculli, D.; Diaz, J.; Dittrich, Y.; Ernst, N.; Felderer, M.; Feldt, R.; Filieri, A.; de França, B. B.N.; Furia, C. A.; Gay, G.; Gold, N.; Graziotin, D.; He, P.; Hoda, R.; Juristo, N.; Kitchenham, B.; Lenarduzzi, V.; Martínez, J.; Melegati, J.; Mendez, D.; Menzies, T.; Molleri, J.; Pfahl, D.; Robbes, R.; Russo, D.; Saarimäki, N.; Sarro, F.; Taibi, D.; Siegmund, J.; Spinellis, D.; Staron, M.; Stol, K.; Storey, M.-A.; Taibi, D.; Tamburri, D.; Torchiano, M.; Treude, C.; Turhan, B.; Wang, X.; Vegas, S.: Empirical Standards for Software Engineering Research, 2021, arXiv: 2010.03525 [cs].

[RE12] Romeike, R.; Eichler, K.-P.: Forschendes Lernen mit Apps für Smartphones und Tablets – Studentische Forschungspartnerschaften im Lehramtsstudium Informatik/Mathematik. In (Desel, J.; Haake, J. M.; Spannagel, C., Hrsg.): DeLFI 2012: Die 10. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. Hagen, S. 279–290, 2012.

[Sa19] dos Santos, H. M.; Durelli, V. H. S.; Souza, M.; Figueiredo, E.; da Silva, L. T.; Durelli, R. S.: CleanGame: Gamifying the Identification of Code Smells. In: Proceedings of the XXXIII Brazilian Symposium on Software Engineering. SBES 2019: XXXIII Brazilian Symposium on Software Engineering. ACM, Salvador Brazil, S. 437–446, 2019.

[SF18] Stol, K.-J.; Fitzgerald, B.: The ABC of Software Engineering Research. ACM Transactions on Software Engineering and Methodology 27/3, S. 1–51, 2018.

[Sh03] Shaw, M.: Writing Good Software Engineering Research Papers. In: ICSE '03: Proceedings of the 25th International Conference on Software Engineering. IEEE, Portland, S. 726–736, 2003.

[So17] Sonntag, M.; Rueß, J.; Ebert, C.; Friederici, K.; Schilow, L.; Deicke, W.: Forschendes Lernen im Seminar: Ein Leitfaden für Lehrende. Humboldt-Universität zu Berlin, Berlin, 2017, isbn: 978-3-86004-328-8.

[SOF18] Senft, B.; Oberthür, S.; Fischer, H.: Forschendes Lernen in der Informatik – In praxisnaher Projektgruppe einen Softwareentwicklungsprozess erforschen. In (Neuber, N.; Paravicini, W.; Stein, M., Hrsg.): Forschendes Lernen – The wider view. Bd. 3. Schriften zur Allgemeinen Hochschuldidaktik, WTM, Münster, 2018.

[Ts18] Tsantalis, N.; Mansouri, M.; Eshkevari, L. M.; Mazinanian, D.; Dig, D.: Accurate and Efficient Refactoring Detection in Commit History. In: Proceedings of the 40th International Conference on Software Engineering. ICSE '18: 40th International Conference on Software Engineering. ACM, Gothenburg Sweden, S. 483–494, 2018.

[Wi14] Wieringa, R. J.: Design Science Methodology for Information Systems and Software Engineering. Springer, Berlin, 2014, isbn: 978-3-662-43838-1. [Zo15] Zobel, J.: Writing for Computer Science. Springer, London, 2015, isbn: 978-1-4471-6638-2.

[Zo15] Zobel, J.: Writing for Computer Science. Springer, London, 2015, isbn: 978-1-4471-6638-2.

Entwicklung eines digitalen Lehrformats unter Berücksichtigung des student engagement

Jonas Kötter[1], Uwe Hoppe[2]

Abstract:
Die Corona-Pandemie hat zu einer umfassenden Veränderung der Lehre an deutschen Universitäten beigetragen. Innerhalb kürzester Zeit mussten analoge Lehrformate in digitale Lehrformate umgewandelt werden. Mithilfe eines Fallbeispiels einer deutschen Universität soll der Fragestellung nachgegangen werden, wie ein zuvor klassisches Lehrformat (Präsenzvorlesung) in ein digitales Lehrformat umgewandelt werden kann und welche Auswirkungen die Veränderungen auf die Zufriedenheit und die durchschnittliche Note der Studierenden haben. Dazu werden anhand des student engagement unterschiedliche Dimensionen des Verhaltens genauer analysiert. Aus den Ergebnissen der Forschung werden Handlungsempfehlungen für derartige digitale Lehrformate im Allgemeinen abgeleitet.

Keywords:
Digitale Lehr- und Lernformen; student engagement; universitärer Kontext

1 Einleitung

Die im Frühjahr 2020 in Europa auftretende Corona-Pandemie hat zu einem Wandel der Lehre im universitären Kontext geführt [Ch20]. Innerhalb kürzester Zeit wurden an vielen deutschen Hochschulen digitale Lehrformate entwickelt und eingesetzt, um ein Distanzlernen der Studierenden zu ermöglichen [BVH20]. Dies führte sowohl bei den Studierenden und Lehrenden als auch bei den Institutionen selbst zu zahlreichen neuen Herausforderungen und Problemen [Ch20], wie bspw. unzureichende Hilfsangebote seitens der Universitäten im Bereich digitale Lehre [HB20]. Etwa ein Jahr später konnten bereits zahlreiche Erfahrungen in der Entwicklung und Umsetzung digitaler Lehrformate gesammelt werden [BVH20].

Besonders positive Erfahrungen, die sich aus der digitalen Lehre ergeben haben, sollen dauerhaft sowohl in der Strategie der Hochschulen als auch in ihrer Struktur und Kultur verankert werden, damit die Hochschulen krisensicherer und zukunftsfähiger aufgestellt sind. Entscheidend für eine nachhaltige Entwicklung an Hochschulen ist die frühzeitige Sammlung von richtungsweisenden Informationen, die nutzenbringend in den strategischen Entwicklungsprozess miteingebracht werden [Bi20]. Deswegen ist es notwendig, dass nicht nur Top-Down, sondern auch Botton-Up Ideen für die Weiterentwicklung der digitalen Lehre identifiziert werden [SB18]. Neben unterschiedlichen Perspektiven, die in den Strategieentwicklungsprozess einbezogen werden sollen, sollte laut [Ch20] besonders der zeitliche Rahmen betrachtet werden, unter dem die Entscheidungen getroffen wurden. Beispielsweise können die getroffenen Annahmen und Abwägungen bei ad hoc Entscheidungen anders aussehen, als bei Entscheidungen, die einen größeren zeitlichen Rahmen haben [Bi20]. Daher ist es umso wichtiger, Vorbilder gelungener digitaler Lehre frühzeitig zu identifizieren und anhand von diesen Erfahrungen zu lernen [SB18]. Diese Arbeit soll einen Beitrag dazu leisten, relevante Informationen über die Gestaltung eines digitalen Lehrformates im universitären Kontext als Best-Practice Beispiel aufzuzeigen. Grundlage dafür ist die Veranstaltung „Projektmanagement", welche zuvor als klassische Vorlesung in Präsenz durchgeführt wurde und aufgrund der Corona-Pandemie in eine E-Learning Veranstaltung, im Speziellen als Inverted Classroom, transformiert wurde. Die Zielsetzungen dieser Arbeit bestehen zum einen darin aufzuzeigen, wie ein digitales Lehr- und Lernformat gestaltet sowie evaluiert werden kann. Zum anderen sollen die gewonnen Erkenntnisse aus der Evaluation dazu genutzt werden, Handlungsempfehlungen für zukünftige digitale Lehr- und Lernformate Bottom-Up zu erzeugen. Aus diesen Überlegungen ergeben sich die folgenden Forschungsfragen:

1 Universität Osnabrück, BWL/Organisation und Wirtschaftsinformatik, Katharinenstr. 1, 49069 Osnabrueck, Deutschland, jonas.koetter@uni-osnabrueck.de

2 Universität Osnabrück, BWL/Organisation und Wirtschaftsinformatik, Katharinenstr. 1, 49069 Osnabrueck, Deutschland, uwe.hoppe@uni-osnabrueck.de

FF1: Wie kann ein digitales Lehrformat zeitlich und örtlich gestaltet werden, welches zuvor in Präsenz durchgeführt wurde?

FF2: Führt eine Veränderung der verhaltensbeeinflussenden Faktoren bei Studierenden zu einer Veränderung der Zufriedenheit und der durchschnittlichen Note?

FF3: Welche Handlungsempfehlungen lassen sich zur Ent- und Weiterentwicklung des digitalen Lehrformates ableiten?

Im Kapitel zwei wird zunächst die derzeitige didaktische Denkweise, wie erfolgreiches Lernen ermöglicht wird, dargestellt. Darauf aufbauend wird das Modell des *student engagement* i. V. m. den Outcomes vorgestellt, um eine Möglichkeit der Evaluation der Lehrveranstaltung aufzuzeigen. Im dritten Kapitel werden sowohl das ehemalige Veranstaltungskonzept als auch das neu entwickelte Veranstaltungsformat erläutert **(FF1)**. Die Evaluation der Ergebnisse bildet das vierte Kapitel, bei dem die Lehrveranstaltung aus Sicht der Studierenden sowohl qualitativ als auch quantitativ evaluiert wird **(FF2)**. Ergänzend zu den gewonnenen Erkenntnissen werden Handlungsmaßnahmen für das Fallbeispiel bzw. für zukünftige digitale Lehrveranstaltungen abgeleitet **(FF3)**. Zum Schluss der Arbeit finden sich im Fazit ein Resümee der relevanten Erkenntnisse, die Grenzen der Arbeit und weitere mögliche Forschungslücken.

2 Student engagement

Im Rahmen der Transformation der Lehrveranstaltung wurde ein Paradigmenwechsel unter dem Motto *Shift from Teaching to Learning* gestartet. Statt der klassischen Fokussierung auf die Inhalte wurde der Fokus des Lehrformats auf die zu erwerbenden Kompetenzen gelegt. Bei dem neuen Ansatz sollten nach [Ba14] folgende drei Bereiche beachtet werden: Zum ersten stehen die Studierenden im Mittelpunkt der Lehre. Zum zweiten sollen die Lehrenden als Moderatoren oder Coaches fungieren. Zusätzlich sollen die Eigenständigkeit und die Heterogenität der Studierenden berücksichtigt werden [Ba14]. Nachdem die didaktischen Grundüberlegungen zur Entwicklung der Lehrveranstaltung erläutert wurden, soll nun ein Indikator betrachtet werden, der zur Beurteilung der digitalen Lehre herangezogen werden kann. Laut einer systematischen Literaturrecherche von [Tr10] zählt das *student engagement* zu den relevantesten Determinanten, die das digitale Lernen beeinflussen. Nur wenn Studierende Zeit und Energie aufwenden, kann die persönliche Entwicklung vorangetrieben werden [Ba14]. Das *student engagement* wird durch unterschiedliche Facetten wie bspw. Interaktion, Aufwand und Zeit, die ein*e Studierende*r aufwendet, beeinflusst. Das Ziel der Veränderung des *student engagement* ist die Entwicklung der Studierenden positiv zu beeinflussen und die Lernerfahrung zu verbessern [Tr10]. Somit beschreibt der Begriff des *student engagement* nicht die Engagiertheit, sondern die Einflussfaktoren, die bewirken, dass ein*e Studierende*r sich engagiert zeigt. Da sich das student engagement nicht nur mit der Beteiligung oder Teilnahme von Studierenden beschäftigt [Tr10], soll an dieser Stelle das Dreikomponentenmodell (affektiv, kognitiv und Konation) von Rosenberg und Hoveland aus dem Jahr 1960 zur weiteren Einteilung des *student engagement* verwendet werden (vgl. Abb. 1) [RH60].

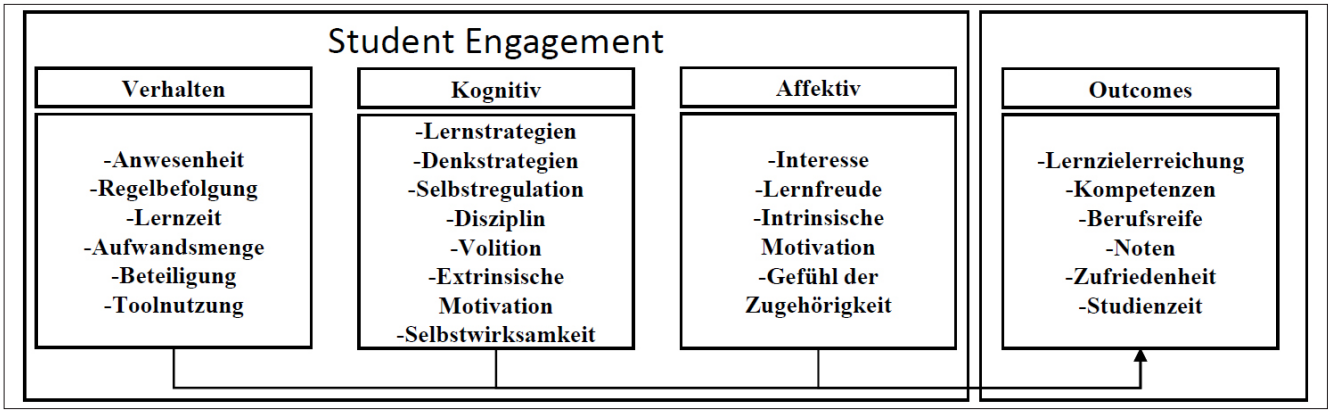

Abb. 1: Darstellung des Modells: Student engagement (In Anlehnung an Persike 2019)

Eine günstige Konstellationen der Dimensionen in den Konstrukten erster Ordnung (affektiv, kognitiv und Konation) deutet auf eine hohe Bereitschaft der Studierenden hin, den Kompetenzerwerb eigenverantwortlich so zu konzipieren, sodass bspw. die persönlichen oder akademischen Ziele mit Erfolg erreicht werden [Pe19]. Das affektive Konstrukt beinhaltet Gefühle und Emotionen, die sich auf den Einstellungsgegenstand beziehen. Zu einem Einstellungsgegenstand kann all das gezählt werden, was bspw. Studierende wahrnehmen. Einstellungsobjekte können dagegen konkret oder abstrakt sein [Bo03]. Zu dem affektiven Konstrukt zählen Dimensionen wie Interesse, Lernfreude oder auch das Zugehörigkeitsgefühl [Pe19]. Das kognitive Konstrukt wird geprägt

durch Meinungen der Studierenden bezüglich des Einstellungsobjektes [Bo03]. Darunter fallen unter anderem Dimensionen wie Lern- und Denkstrategien, Disziplin, Selbstwirksamkeit oder Volition [Pe19]. Das Konstrukt des Verhaltens (=Konation) schließt zum einen Verhaltensabsichten der Studierenden ein. Zum anderen werden damit Handlungen gemeint, welche den Einstellungsgegenstand betreffen [Bo03]. Zu den Dimensionen gehören in diesem Konstrukt unter anderem die *Anwesenheit* der Studierenden, die *Regelbefolgung*, *Lernzeiten*, die *Aufwandsmenge* und die *Beteiligung* sowie die *Toolnutzung* [Pe19]. Die Anwendbarkeit des oben dargelegten Modells wurde im Jahr 1984 von Breckler belegt [Bo03]. Mithilfe von Outcomes wie bspw. Noten, Zufriedenheit, Studienzeit, Berufsreife, Kompetenzen oder Lernzielerreichung lässt sich feststellen, ob sich bspw. durch eine Veränderung der Konstrukte erster Ordnung das *student engagement* verändert. Die drei Konstrukte erster Ordnung (Verhalten, kognitiv, affektiv) beeinflussen die Outcomes [Pe19].

In dem Fallbeispiel wird diesesWissen genutzt, um die Outcomes zu maximieren. Allgemein, so beschreiben es die Autoren [HO10], ist in der Lehr- und Lernforschung bekannt, dass Blended Learning zu mehr *student engagement* führen kann. Die in Kapitel vier evaluierte Fallstudie beschäftigt sich mit dem Konstrukt des Verhaltens und den damit veränderten Outcomes der durchschnittlichen Noten und mit der Zufriedenheit der Studierenden. Die Fokussierung begründet sich darin, dass sich zum einen die Mehrzahl der veränderten Dimensionen des neu konzipierten Inverted Classroom Formats in dem Konstrukt des Verhaltens befindet (vgl. Kapitel 3.1). Zum anderen schreiben zahlreiche Autoren wie [SM11] und [Tr10], dass die *Beteiligung* zu den wichtigsten Dimensionen zählt, um die Outcomes zu maximieren. 1957 stellte Festinger die Theorie kognitiver Dissonanz auf, die davon ausgeht, dass das kognitive Konstrukt durch das Konstrukt des Verhaltens beeinflusst wird [Fe57]. Daher soll an dieser Stelle zunächst nur das Verhalten betrachtet werden.

3 Fallbeispiel - Ausgangslage

Ausgangspunkt der zu transformierenden Lehrveranstaltung bildet die in Präsenz durchgeführteVorlesung Projektmanagement,welche von etwa 60 Studierenden in demWintersemester 2019/2020 besucht wurde. Bei der Veranstaltung handelt es sich um ein-Wahlpflichtmodul in den Masterstudiengängen Betriebswirtschaftslehre und Wirtschaftsinformatik. In der Präsenzphase wurden etwa vier bis acht freiwillige Präsentationen von Gruppen von bis zu vier Studierenden im Semester gehalten und Literaturdiskussionen geführt. Nach der Präsenzphase haben die Studierenden die Zeit bis zur nächsten Veranstaltung für die Vorund Nachbereitung genutzt, um bspw. Präsentationen zu konzipieren oder auch die vom Lehrenden vorbereiteten Foliensätze durchzuarbeiten und die Inhalte zu verinnerlichen. Die Note am Ende des Semesters setzt sich aus einer Klausur im Umfang von 90 Minuten und der freiwilligen Präsentation im Semester zusammen.

3.1 Inverted Classroom Konzept

Auf Grundlage des Drittmittelantrages i. V. m. der fortschreitenden Corona-Pandemie musste innerhalb von neun Monaten, bis zum Semesterstart (Wintersemester 2020/2021), ein Blended Learning Format entwickelt werden, welches im digitalen Semester eingesetzt werden konnte und nach der Pandemie wieder in Präsenz durchführbar ist. Das Ergebnis ist ein Blended Learning Format in Form eines Inverted Classrooms. Bei einem Inverted Classroom, auch Flipped Classroom genannt, werden die Inhalte zu einer Vorlesung vorab bereitgestellt, sodass in der *In-Class* Phase genügend Zeit zur Vertiefung der Inhalte zur Verfügung steht [Lo13]. Das Training bzw. die Vertiefung der Inhalte finden somit in der *In-Class* Phase mit anderen Studierenden statt und in der *Out-of-Class* Phase findet der Wissenstransfer statt [Ha18]. Sowohl die Phase *Before Class* als auch die Phase *During Class* können im Inverted Classroom der Phase *In-Class* zugeordnet werden [Lo13]. Als Erweiterung zu den bestehenden zwei Phasen wurde

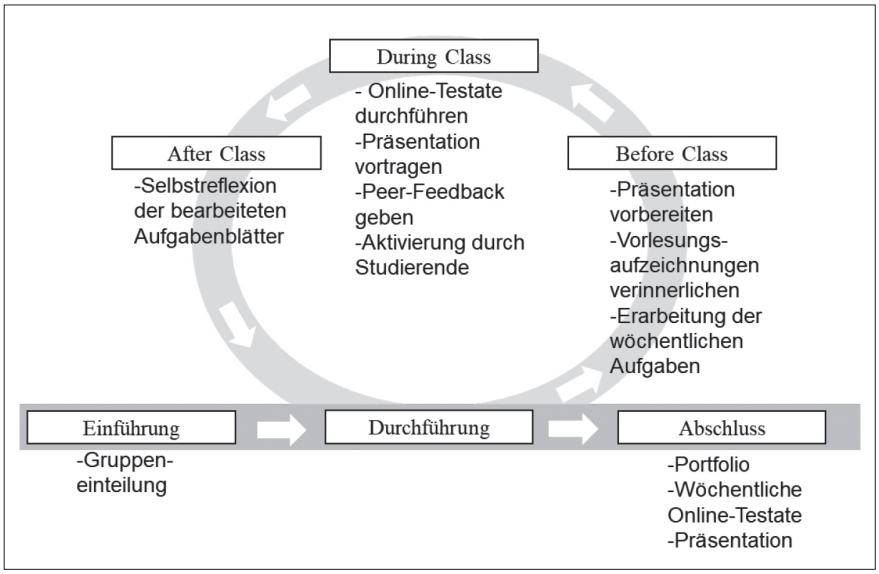

Abb. 2: Darstellung der neuen Struktur der Lehrveranstaltung

eine neue Einteilung in drei Phasen *(Before Class, During Class und After Class)* vorgenommen [Mo14], damit der Aufbau und Ablauf der Lehrveranstaltung für die Studierenden transparenter dargestellt werden kann (vgl. Abb. 2).

Der Prozess soll nicht als Zeitstrahl verstanden werden, der von Phase zu Phase (Einführung, Durchführung und Abschluss) verläuft, sondern als sich immer wiederholender Prozess zwischen den Phasen *Before Class*, *During Class* und *After Class*, welche nach 13 Veranstaltungen endet (Abschluss). Vorgeschaltet vor dem Phasenprozess wird eine Einleitungsveranstaltung (Einführung) gehalten, bei der die Studierenden die Struktur und den Ablauf der Veranstaltung kennenlernen und sich mithilfe eines Persönlichkeitsselbsttests in heterogene Gruppen einteilen. Beim zweiten Termin von den insgesamt 14 beginnt der oben skizzierte Prozess mit der Phase *Before Class*.

Im Nachfolgenden wird zum einen die oben dargestellte Abbildung näher erläutert und zum anderen werden verhaltensbezogene Maßnahmen auf Basis der sechs identifizierten verhaltensbezogenen Dimensionen vorgestellt. Innerhalb von Kleingruppen bis zu vier Studierenden werden die Präsentationen für die *During Class* Phase erarbeitet. Die Inhalte ergeben sich zum einen aus den Videos, welche zuvor in Courseware bereitgestellt wurden. Die Coureware ist ein Lernmanagementsystem, welches in der Veranstaltung Projektmanagement dazu genutzt wird, Inhalte wie Videos und Aufgabenblätter in zeitlichen Abständen sichtbar zu schalten *(Toolnutzung)*. Am Ende der Videos werden den Studierenden weitere Bücher empfohlen, um die Inhalte aus der Vorlesung weiter vertiefen zu können *(Aufwandsmenge)*. In der Phase *During Class* finden zu Beginn die wöchentlichen Online-Testate statt *(Anwesenheit/ Regelbefolgung)*. Die Online-Testate umfassen zehn Single- oder Multiple-Choise-Fragestellungen und haben einen zeitlichen Rahmen von 20 Minuten. Die Fragestellungen innerhalb der Testate dienen zur Überprüfung der kontinuierlichen Mitarbeit *(Lernzeit)* außerhalb der *During Class* Phase. Die Online-Testate werden Jahr für Jahr erweitert, sodass über Jahre ein umfassender Fragenpool etabliert werden kann. Im Anschluss an die Online-Testate erfolgt eine 50-minütige Präsentation inkl. einer 15-minütigen Aktivierung sämtlicher Studierender mithilfe von bspw. Quizzen oder Rollenspielen, die von der vortragenden Gruppe vorbereitet werden *(Beteiligung)*. Im Anschluss wird eine Gruppe zufällig ausgewählt, die geordnetes Feedback zu der Präsentation und den Inhalten inkl. Aktivierung gibt. Zusätzlich dazu geben die Lehrenden Feedback zum Vortag und Inhalt *(Beteiligung)*. Die dritte Phase bildet die *After Class* Phase. Innerhalb dieser Phase bearbeiten die Studierenden die Aufgabenblätter für die nächste Woche *(Aufwandsmenge)*. Wie auch die Videos werden die Aufgabenblätter automatisiert in Courseware zeitlich versetzt sichtbar geschaltet, sodass die Studierenden einen zeitlich angepassten und definierten Workload haben *(Lernzeit)*. Zusätzlich zu dem wöchentlichen Aufgabenblatt führen die Studierenden eine Selbstreflexion durch. Dazu vergleichen sie die Inhalte der Präsentation mit den eigens ausgearbeiteten Inhalten der Aufgabenblätter. Falls negative Differenzen festgestellt werden, können diese korrigiert werden, sodass ein Lerneffekt eintritt. Aber auch positive Abweichungen sollen festgehalten werden. Neben den Phasen hat sich die Prüfungsleistung verändert. Statt einer Klausur zum Ende des Semesters muss ein Portfolio pro Gruppe abgegeben werden, welches sowohl die Lösungen zu sämtlichen Aufgabenblättern als auch die Selbstreflexionen beinhalten. Da das Portfolio eine Gruppenarbeit ist, wird dieses auch mit einer Gruppennote bewertet, die zu 70 Prozent in die Note eingeht. Zusätzlich wird die Präsentation im Semester individuell bewertet. Diese geht zu 30 Prozent in die Note ein. Als Grundvoraussetzung dienen die wöchentlichen Online-Testate, die zu 70 Prozent bestanden sein müssen, um das Modul bestehen zu können. Übergeordnet über die Phasen hinweg wurden im Vorfeld Lernziele in Anlehnung an [MK06] und [Kr02] formuliert. Bei der Konzipierung der Lernmaterialien wurden unterschiedliche Taxonomiestufen bei der Formulierung der Lernziele für die Videos und Aufgabenblätter berücksichtigt. In der *Before Class* Phase wurden kognitive Lernziele für die Stufen Remember und Understand formuliert [Kr02]. Die Lernziele wurden in jeder Vorlesungsaufzeichnung erläutert. In der Phase *During Class* erfolgte die Überprüfung der zuvor gesetzten Lernziele mithilfe der Online-Testate. Im Anschluss an die Lehrveranstaltung in der Phase *After Class* konnten weitere kognitive Lernziele anhand der Taxonomiestufen wie Apply, Analyze und Evaluate formuliert werden [Kr02]. Zum besseren Verständnis wurden die Lernziele zu Beginn der Aufgabenblätter dargestellt. Mithilfe des Portfolios konnte im Nachgang eine Überprüfung der Lernziele der Aufgabenblätter vorgenommen werden. Die Lernziele aus den Vorlesungsaufzeichnungen wurden durch die wöchentlichen Online-Testate überprüft.

3.2 Methodik

Die Befragung der Studierenden erfolgte innerhalb der letzten Lehrveranstaltung Projektmanagement mithilfe der online Software Limesurvey. Insgesamt haben an der standardisierten und anonymen Befragung 22 der 29 Studierenden teilgenommen. Die Zielpopulation besteht aus 15 männlichen und sechs weiblichen Studierenden sowie einer Person ohne Angabe des Geschlechts. Das Alter der Studierenden beträgt im Durchschnitt 25 Jahre. Insgesamt haben 11 Studierende aus der Wirtschaftsinformatik und 11 Studierende aus der Betriebswirtschaftslehre an der Erhebung teilgenommen. Die Auswertung der Outcomes erfolgte anhand von zwei offenen Fragestellungen, die nach [Ma20] in Form einer Inhaltsanalyse ausgewertet wurden (vgl. Tab. 1). Die Studierenden konnte angeben was ihnen in der Veranstaltung besonders gut gefallen hat und was ihnen nicht gut gefallen hat. Für die einzelnen Dimensionen der drei Konstrukte wurden verschiedenen Dimensionen, bspw. Anwesenheit oder Lernzeit, im Vorfeld definiert, um im nachfolgenden Kreuzcodierungsverfahren alle Aussagen zielgerecht zuordnen zu können. Bei der Formulierung der Fragestellungen

wurde berücksichtigt, dass die Dimensionen sowohl positiv als auch negativ ausgeprägt sein können. Zudem weisen negative Aussagen nicht auf negative Outcomes hin [Tr10]. Mithilfe der zuvor selektierten sechs verhaltensbezogenen Dimensionen (vgl. Kapitel 2) wurde ein deduktives Vorgehen angewandt. Zur Messung des Outcomes der Zufriedenheit wurden zwei Items abgefragt (vgl. Tab. 1). Bei den Fragen handelt es sich um geschlossene Fragen, welche mithilfe eine vorgegeben Likert-Skala (1 = Stimme voll zu bis 5 = Stimme überhaupt nicht zu) beantwortet worden konnten. Zusätzlich wurden die durchschnittlichen Noten aus dem Wintersemester 2019/2020 (ehemaliges Lehrkonzept) und aus dem Wintersemester 2020/2021 (neues Lehrkonzept) miteinander verglichen.

4 Evaluation

Nachfolgend werden die Ergebnisse der verhaltensbezogenen Dimensionen nach Mayring erläutert und in einer Tabelle zusammengefasst (vgl. Tab. 1). Insgesamt konnten in dem aufgezeigten Modell aus Kap. 2 alle sechs Dimensionen bestätigt werden. Zusätzlich wurde aus dem kognitiven Konstrukt die Dimension *Lernstrategien* und aus dem affektiven Konstrukt die Dimension *Interesse* identifiziert. Bei den aufgestellten Regeln *(Regelbefolgung)* konnten insgesamt 26 Aussagen festgehalten werden. Positiv haben die Studierenden die Regeln bezüglich der heterogenen Gruppeneinteilung, der freien Ausgestaltung des Portfolios durch offen formulierte Aufgabenblätter und die regelmäßig stattfindenden Veranstaltungen beurteilt. Die Studierenden wünschen sich weiterführende Hinweise zur Erstellung des Portfolios nicht nur in mündlicher Form, sondern auch in schriftlicher Form. Zusätzlich wurde von zwei Studierenden kritisiert, dass sie nicht wussten, ob es sich um eine Vorlesung oder ein Seminar handelt. Von den 22 Studierenden haben 10 Studierende angegeben, dass die *Aufwandsmenge* im Vergleich zu anderen Veranstaltungen deutlich höher ist. Zwei Studierende empfanden es als positiv, dass am Ende des Semesters die Klausur entfällt und dass es dadurch zu einer Entlastung in der Prüfungsphase kommt. Sechs Studierende haben die positive *Beteiligung* in der Gruppenarbeit hervorgehoben. Dabei wurde die *Beteiligung* nicht nur auf das Feedback und die Aktivierung sämtlicher Studierender nach den Präsentationen bezogen, sondern auch auf den Austausch innerhalb der Gruppen

Tab. 1: Darstellung der identifizierten Konstrukte und Dimensionen

Dimensionen	Was hat Ihnen besonders gut gefallen	Was hat Ihnen nicht gut gefallen
Regelbefolgung (v)	10	16
Aufwandsmenge (v)	2	10
Beteiligung (v)	6	0
Lernzeit (v)	6	1
Toolnutzung (v)	4	1
Anwesenheit (v)	0	2
Lernstrategien (k)	10	0
Interesse (a)	3	0

v = Verhalten ; k = kognitiv ; a = affektiv

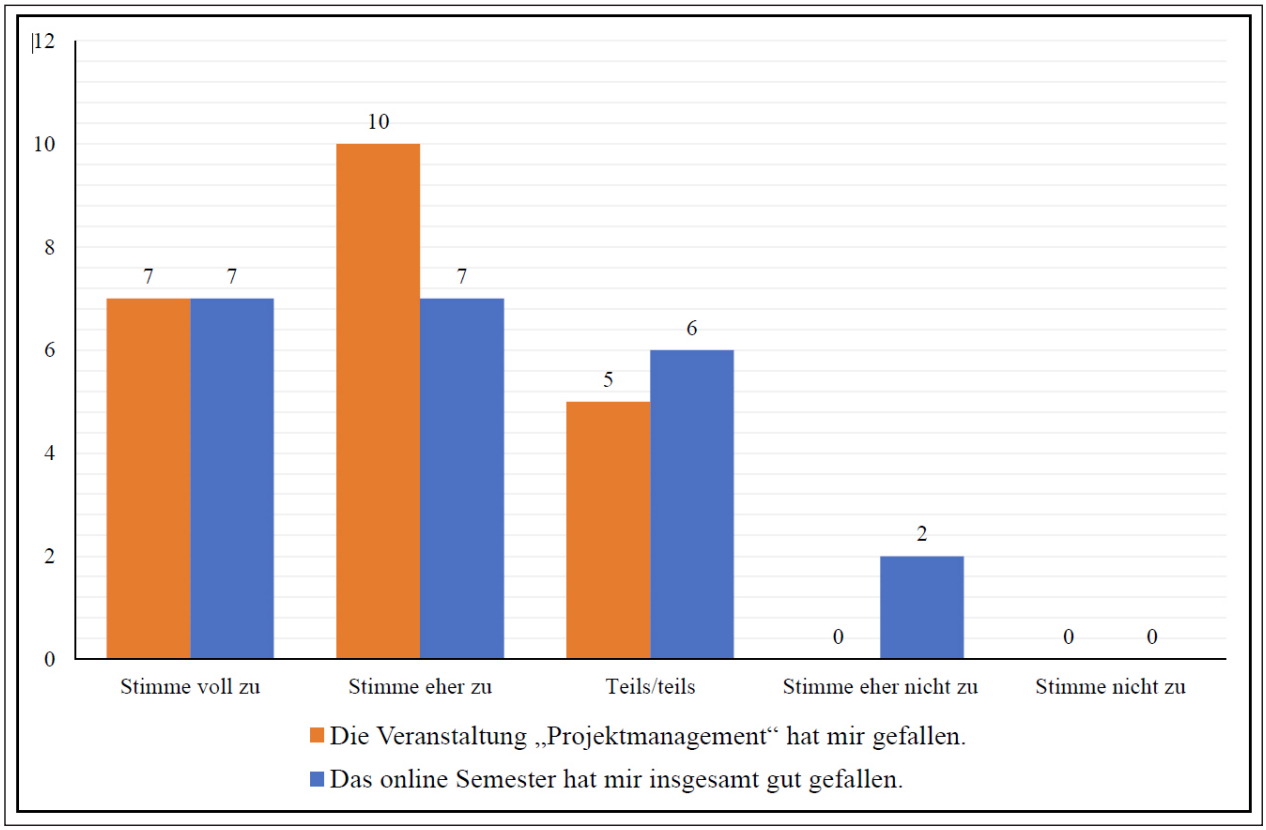

Abb. 3: Darstellung des Outcomes - Zufriedenheit der Studierenden

in der *After Class* Phase. Bezüglich der *Lernzeit* der Studierenden lässt sich festhalten, dass diese sechs Studierenden den wöchentlichen Upload der Unterlagen und die damit verbundene fest definierte *Lernzeit* als positiv beschrieben haben. Eine negative Aussage konnte bezüglich der *Lernzeit* identifiziert werden. Ein Studierender berichtete über eine ungleich verteile *Lernzeit* für Studierende innerhalb der Gruppe. Positiv wurde von vier Studierenden bei der *Toolnutzung* festgestellt, dass die Videos in einem hochwertigen Format waren und einen passenden zeitlichen Umfang hatten. Ein Studierender hätte sich gewünscht, dass die Videos Sprungmarken aufweisen sollten. Die *Anwesenheit* wurde von zwei Studierenden als negativ bewertet, da zum einen die Flexibilität durch verbindliche wöchentliche Online-Testate verloren geht und zum anderen wurde bemängelt, dass die Wissensvertiefung in der *During Class* Phase nicht ausreichend umgesetzt wurde. Insgesamt haben 10 Studierende sich positiv zu der Art und Weise der Lehre *Lernstrategien* geäußert. Als positiv konnte die schrittweise Erarbeitung der Inhalte sowie der praktische Bezug festgestellt werden. Drei Studierende haben besonderes *Interesse*, auch im Nachgang weiter im Projektmanagement aktiv zu werden.

Im Folgenden wird das Outcome der Zufriedenheit der Studierenden anhand der zwei formulierten Items mit der Skala (1 = Stimme voll zu bis 5 = Stimme überhaupt nicht zu) dargestellt. Die Fragestellungen finden sich in der Legende der Abbildung wieder. Die Zufriedenheit der Studierenden mit der Lehrveranstaltung Projektmanagement wurde mit der Zufriedenheit der Studierenden mit dem gesamten digitalen Semester verglichen.

Etwa 77 Prozent (n=17) der Befragten stimmten der Aussage, dass ihnen die Veranstaltung gut gefallen hat, voll bzw. eher zu. 23 Prozent (n=5) der Studierenden hatten keine klare Meinung (teils/teils) dazu. Der Durchschnitt der gemessenen Zufriedenheit der Veranstaltung Projektmanagement befindet sich bei 1,91. Zusätzlich wurde die Zufriedenheit der Studierenden bezüglich des gesamten digitalen Semesters abgefragt. 64 Prozent (n=14) der Studierenden stimmten der Aussage voll bzw. eher zu, dass ihnen das digitale Semester gefallen hat. 27 Prozent (n=6) der Studierenden hatten keine klare Meinung (teils/teils) und neun Prozent (n=2) der Studierenden stimmten dieser Aussage eher nicht zu. Im Durchschnitt wurde das digitale Semester mit 2,14 bewertet. Zusammenfassend lässt sich festhalten, dass die Veranstaltung Projektmanagement positiver bewertet wurde als das digitale Semester. Neben der Zufriedenheit soll im Anschluss das Outcome der durchschnittlichen Note der Studierenden betrachtet werden. Im Wintersemester 19/20 lag der Notenschnitt der Veranstaltung Projektmanagement bei 3,5. Der Notenschnitt im Wintersemester 20/21 lag im Schnitt bei 1,66.

4.1 Handlungsempfehlungen

Insgesamt konnten 41 positive und 30 negative Aussagen der Studierenden identfiziert werden. Für zukünftige digital geprägte Veranstaltungen lassen sich die folgenden Maßnahmen ableiten, um die Outcomes des *student engagement* nachhaltig zu maximieren: Der aufgestellte Ablauf (vgl. Abb. 2) hat sich für die Studierenden als sehr verständlich herausgestellt. Innerhalb der Einführungsveranstaltung wird weiterhin die Gruppeneinteilung mithilfe von Persönlichkeitstests stattfinden, um die Heterogenität anhand der Persönlichkeitsmerkmale der Studierenden innerhalb der Gruppen zu gewährleisten und zu fördern. Zusätzlich sollen in der ersten Veranstaltung die Erwartungen sowohl schriftlich als auch mündlich kommuniziert werden (Anforderungskatalog), damit die Studierenden im Nachgang mehr über das neue Lehrkonzept, die damit verbundenen Anforderungen und die neue Prüfungsform in Erfahrung bringen können *(Regelbefolgung)*. Zudem soll ein Beispiel für ein Portfolio vorgestellt werden, damit die Studierenden verstehen, was ein Portfolio ist und welche Kriterien bei der Konzeption zu beachten sind. In der *Before Class* Phase, werden weiterhin kurze Videos mit definierten Lernzielen zur Verfügung gestellt, damit die Studierenden die Inhalte verinnerlichen. Ergänzend dazu könnten in den Videos Fragen zu den Inhalten für die Studierenden eingepflegt werden, sodass diese sich noch aktiver mit den Inhalten beschäftigen. Zusätzlich könnten Sprungmarken in den Videos eingesetzt werden, um den Lernprozess der Studierenden zielführender zu gestalten. Durch die Einbindung von Praxispartnern soll in Zukunft die Attraktivität der Lehrveranstaltung in der *During Class* Phase weiter gesteigert werden *(Anwesenheit)*. Bezüglich des Peer-Feedbacks nach Präsentationen durch eine zufällig ausgeloste Gruppe lässt sich festhalten, dass dadurch sowohl eine hohe *Anwesenheit* der Studierenden innerhalb der Vorlesung erreicht wurde als auch ein extrinsischer Druck, der Gruppe zuzuhören und Nachfragen *(Beteiligung)* bei Unklarheiten zu stellen. Eine individuelle Zeitaufschreibung der Studierenden, die im Portfolio angefertigt werden musste, hilft dabei Aufgabenblätter zu identifizieren, bei denen der zeitliche Umfang überschritten wurde *(Aufwandsmenge)* oder die *Aufwandsmenge* ungleich zwischen den Aufgaben innerhalb der Aufgabenblättern verteilt waren *(Toolnutzung)*. Zudem werden die nicht benoteten wöchentlichen Online-Testate beibehalten. Dadurch wird zum einen die *Anwesenheit* in der *During Class* weiter gestärkt und zum anderen wird dadurch eine dauerhafte und aktive Arbeit der Studierenden im Semester gewährleistet *(Lernzeit)*. Die in der *After Class* Phase durchzuführende Selbstreflexion soll auch weiterhin beibehalten werden, damit die Studierenden eigene Stärken und Schwächen ihrer Ergebnisse analysieren und anpassen können. Übergreifend über alle Phasen hinweg soll das LMS weiterhin genutzt werden, um den Studierenden eine vertraute und stetig optimierte Lernumgebung zu Verfügung zu stellen *(Toolnutzung)*. Durch die Nutzung des LMS ergeben sich auch Vorteile für die Lehrenden, indem die Online-Testate Jahr für Jahr durch neue Fragen erweitert werden *(Toolnutzung)*.Ergänzend dazu sollte ein Forum für die Studierenden eingerichtet werden, um die asynchrone *Beteiligung* zu steigern.

5 Fazit

In dieser Arbeit wurde aufgezeigt, wie ein zuvor synchrones und in Präsenz durchgeführtes Lehrformat in ein digitales Lehrformat transformiert wurde. Innerhalb des neu entwickelten Lehrformates wurden drei unterschiedliche Phasen (*Before Class*, *During Class* und *After Class*) unterschieden, es wurde eine neue Prüfungsform für die Studierenden eingeführt und phasenübergreifende Lernziele wurden formuliert (**FF1**). Zusätzlich wurden verhaltensverändernde Maßnahmen im Vorfeld der Veranstaltung formuliert, um das *student engagement* und die damit verbundenen Einflussfaktoren zu verändern. In zwei offenen Fragen konnten insgesamt 58 Aussagen zu den verhaltensbeeinflussenden, 10 Aussagen zu den kognitiven und drei Aussagen zu den affektiven Einflussfaktoren identifiziert werden. Dabei wurde festgestellt, dass bspw. ein erhöhter Arbeitsaufwand für die Studierenden und neue Regeln innerhalb der Lehrveranstaltung zu einer Steigerung des *student engagement* führen. Das Ergebnis (Outcome) der Veränderung des *student engagement* hat sich zum einen in der verbesserten durchschnittlichen Note der Studierenden ausgedrückt und zum anderen ist die Zufriedenheit der Studierenden in der Lehrveranstaltung höher als in anderen digitalen Veranstaltungen im Wintersemester 2020/2021 (**FF2**). Veränderungen innerhalb der verhaltensbezogenen Dimensionen sowohl positiver als auch negativer Art führen somit zu einer positiven Veränderung der Outcomes. Aus den gesammelten Erkenntnissen wurden Handlungsempfehlungen (**FF3**) für die acht identifizierten Dimensionen erstellt, die Lehrende, die vor ähnlichen Herausforderungen stehen, berücksichtigten können, um die digitale Lehre erfolgreich zu gestalten. Die Limitationen dieser Arbeit resultieren zum einen daraus, dass eine neue Prüfungsform eingeführt wurde, die ggf. zu anderen durchschnittlichen Noten führt. Zum anderen haben im Wintersemester 20/21 weniger Studierende als üblich an der Veranstaltung und somit auch an der Erhebung teilgenommen. Für die Zukunft sind erneute Befragungen zum *student engagement* im Rahmen der Lehrveranstaltung Projektmanagement geplant. Neben der Messung von weiteren Outcomes wie dem Kompetenzniveau sollen die kognitiven und affektiven Einflussfaktoren auf das *student engagement* genauer analysiert werden. Zudem soll die Veränderung der Zufriedenheit zwischen dem jetztigen und zukünftigen Semestern analysiert werden.

Literatur

[Ba14] Bachmann, H.: Hochschullehre neu definiert–shift from teaching to learning. In (Bachmann, H., Hrsg.): Forum Hochschuldidaktik und Erwachsenenbildung. Bd. 1, Hep Verlag, Bern, S. 14–31, 2014.

[Bi20] Bils, A.; Braun, B.; Bünemann, T.; Scheuring, T.; Sutter, C.; Meyer, V.; Neuner, S.; Wagner, B.; Wistuba, Y.: Corona-Semester 2020–Ad-hoc-Maßnahmen evaluieren und nachhaltig verankern. Diskussionspapier Nr. 11. Hochschulforum Digitalisierung, Berlin, 2020.

[Bo03] Bohner, G.: Einfuehrung. In (Stroebe, W.; Jonas, K.; Hewstone, M., Hrsg.): Sozialpsychologie: Eine Einführung (4. Aufl.) Springer-Verlag, Berlin/Heidelberg, S. 265–313, 2003.

[BVH20] Blömer, L.; Voigt, C.; Hoppe, U.: Corona-Pandemie als Treiber digitaler Hochschullehre. In (Zender, R.; Ifenthaler, D.; Leonhardt, T.; Schumacher, C., Hrsg.): DELFI 2020-Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V. Bonn, S. 343–348, 2020.

[Ch20] Cheema, M.: Covid-19 revolutionising higher education: An educator's viewpoint of the challenges, benefits and the way forward. Life Sci. Med. Biomed 4/9, S. 1–6, 2020.

[Fe57] Festinger, L.: A Theory of Cognitive Dissonance. Stanford University Press, California, 1957.

[Ha18] Hahnzog, S.: Coaching macht E-Learning lebendig:„Inverted-Classroom-Plus". In (Heller, J.; Triebel, C.; Hauser, B.; Koch, A., Hrsg.): Digitale Medien im Coaching. Grundlagen und Praxiswissen zu Coaching-Plattformen und digitalen Coaching-Formaten. Springer Verlag, Berlin, S. 165–172, 2018.

[HB20] Hubble, S.; Bolton, P.: Coronavirus: Update implications for the further and higher education sectors. 1/8908, 2020.

[HO10] Holley, D.; Oliver, M.: Student engagement and blended learning: Portraits of risk. Computers & Education 54/3, S. 693–700, 2010.

[Kr02] Krathwohl, D. R.: A revision of Bloom's taxonomy: An overview. Theory into practice 41/4, S. 212–218, 2002.

[Lo13] Loviscach, J.: The Inverted Classroom: Where to Go from Here. In (Handke, J.; Kiesler, N.; Wiemeyer, L., Hrsg.): The Inverted Classroom Model. The 2nd German ICM-Conference – Proceedings. Oldenbourg Wissenschaftsverlag, München, S. 3–14, 2013.

[Ma20] Mayring, P.: Qualitative Forschungsdesigns. In (Mey, G.; Mruck, K., Hrsg.): Handbuch Qualitative Forschung in der Psychologie. Band 2: Designs und Verfahren (2 Aufl.). Springer Fachmedien, Wiesbaden, S. 3–18, 2020.

[MK06] Marzano, R. J.; Kendall, J. S.: The New Taxonomy of Educational Objectives (2. Aufl.). Corwin Press, California, 2006.

[Mo14] Mok, H.N.: Teaching tip: The flipped classroom. Journal of information systems education 25/1, S. 7–11, 2014.

[Pe19] Persike, M.: Denn sie wissen, was sie tun: Blended Learning in Großveranstaltungen. In (Kauffeld, S.; Othmer, J., Hrsg.): Handbuch Innovative Lehre. Springer, Wiesbaden, S. 65–86, 2019.

[RH60] Rosenberg, M.; Hovland, C.: Cognitive, Affective and Behavioral Components of Attitudes. In (Rosenberg, M. J.; Hovland, C. I.; McGuire,W. J.; Abelson, R. P.; Brehm, J.W., Hrsg.): Attitude organization and change: An analysis of consistency among attitude components, Vol. III. Yale studies in attitude und communication, New Haven, 1960.

[SB18] Schünemann, I.; Budde, J.: Hochschulstrategien für die Lehre im digitalen Zeitalter.Keine Strategie wie jede andere! Arbeitspapier Nr. 38. Hochschulforum Digitalisierung, Berlin, 2018.

[SM11] Sarcletti, A.; Müller, S.: Zum Stand der Studienabbruchforschung. Theoretische Perspektiven, zentrale Ergebnisse und methodische Anforderungen an künftige Studien. Zeitschrift für Bildungsforschung 1/3, S. 235–248, 2011.

[Tr10] Trowler,V.: Student engagement literature review. The higher education academy 11/1, S. 1–15, 2010.

Inverted Classroom kombiniert mit Scrum für die Informatik-Lehre

Karsten Morisse[1], Christian Heidemann[2]

Abstract:
Dieser Werkstattbericht stellt die Kombination des Inverted Classroom Modells mit der agilen Entwicklungsmethodik Scrum in einer Grundlagenveranstaltung der Informatik vor. Neben der fachspezifischen Lehre wird damit auch der überfachliche Kompentenzerwerb adressiert. Der Beitrag stellt die Umsetzung der agilen Lehrmethodik vor und gibt erste Rückmeldungen aus Sicht von Studierenden und Lehrenden.

Keywords:
Inverted Classroom Model; Scrum; Agile Lehre; Future Skills

1 Einleitung

Das Inverted Classroom Modell (ICM) erfreut sich seit einigen Jahren großer Beliebtheit in der Hochschullehre. Beim ICM wird die Phase der Wissensvermittlung aus der traditionellen Lehrveranstaltung umgedreht: Was bisher während der gemeinsamen Veranstaltungszeit präsentiert wurde, wird nun über Texte, Videos u.a. in eine vorgelagerte Selbstlernphase ausgelagert. Die Präsenzzeit wird für aktives Lernen, Vertiefung, Diskussion oder andere aktive Formate genutzt. Das ICM-Konzept wird disziplin- und veranstaltungsübergreifend in der Hochschullehre genutzt. [Mo19], [BW18], [BS18], [HSB19], [Bu18], [Br20].

Die von Sutherland und Schwaber auf [TN86] basierende Scrum-Methodik [SS21] ist ein etabliertes Vorgehensmodell in der Software-Entwicklung. Scrum bietet durch definierte Rollen, Artefakte und Ereignisse einen Rahmen in dem inkrementell an der Entwicklung eines Produktes gearbeitet werden kann. Diese Inkremente werden in Arbeitszyklen (Sprints) erarbeitet, bei denen die stetige Verbesserung des Produktes und der Arbeitsweise im Fokus stehen. Grundgedanke hinter diesem Vorgehen ist die auf Empirie basierende Prozesskontrolle, bei der die im vorangegangenen Zyklus gesammelten Erfahrungen in die Planung und Durchführung des nächsten Zyklus einfließen. Dies geschieht auf Basis der Grundpfeiler Transparenz, Inspektion und Adaption. Mit eduScrum werden die Ideen von Scrum in die Lehre übertragen. Während die Methodik beispielsweise im niederländischen Raum insbesondere in Schulen sehr vielversprechend eingesetzt wird [WS19], ist sie im deutschsprachigen Raum bislang kaum verbreitet. Die große Ähnlichkeit zur Scrum-Entwicklungsmethodik machen dieses Vorgehen für Informatik-Studierende zu einem sehr vielversprechenden Ansatz, da hier neben dem fachlichen Kompetenzerwerb die für die Informatik wichtigen überfachlichen Kompetenzen, beispielsweise Kollaborations- und Kommunikationskompetenz als Future Skills [Hi19], und berufsrelevante Vorgehensmodelle vermittelt werden. Das macht diesen didaktischen Ansatz reizvoll.

Der vorliegende Beitrag beschreibt als Werkstattbericht die Durchführung von ICM in Kombination mit Scrum in einem Grundlagenfach der Informatik in der Hochschullehre. Damit wird das in der späteren Berufspraxis häufig anzutreffenden agile Vorgehen bereits frühzeitg im Rahmen des Studiums erfahren und geübt und dient so dem überfachlichen Kompentenzerwerb. Im Rahmen des Kurses Algorithmen und Datenstrukturen (A+D) an der Hochschule Osnabrück wurden diese beiden Konzepte unter Nutzung verschiedener elektronischer Tools (edX, MS Teams, GitLab) im zweiten Fachsemester zu einem Veranstaltungskonzept integriert. Nach einer Beschreibung des organisatorischen Rahmens in Abschnitt 2, wird in Abschnitt 3 das Vorgehen und die Umsetzung beschrieben und in Abschnitt 4 kritisch reflektiert. Ein Fazit und Ausblick in Abschnitt 5 beschreibt weitere Fragen zur Untersuchung.

1 Hochschule Osnabrück, Fakultät I+I, Albrechtstr. 30, 49076 Osnabrück, Deutschland k.morisse@hs-osnabrueck.de

2 Hochschule Osnabrück, Fakultät I+I, Albrechtstr. 30, 49076 Osnabrück, Deutschland christian.heidemann@hs-osnabrueck.de

2 Organisatorischer Rahmen

Das Pflichtmodul A+D der beiden Informatik-Studienprogramme *Medieninformatik* und *Technische Informatik* wird laut Studienverlaufsplan im zweiten Fachsemester angeboten und besteht aus Vorlesung (3 SWS, Großgruppe im Hörsaal) und Praktikum (1 SWS, Kleingruppen im Rechnerpool). Die Inhalte haben hohe berufspraktische Relevanz und gehören zum Grundkanon der Informatik. Im Sommersemester 2020 (Corona Pandemie) wurde der Vorlesungsanteil von einem der Autoren bereits teilweise in das Inverted Classroom Modell überführt. Inhaltlich werden die theoretischen Grundlagen (Beurteilung und Analyse von Algorithmen und Datenstrukturen, Entwurfsparadigmen) sowie einige ausgewählte Algorithmen behandelt (ausgewählte Inhalte aus [Co09], [SW14]). Der praktische Teil der Veranstaltung wird als Programmierprojekt in der Programmiersprache Java durchgeführt.

Bezogen auf das Kompetenzmodell des HQR [Ku17] kommt man dabei über den Kompetenzbereich Fachkompetenz (Wissen + Verstehen) auf Ebene der Wissensverbreiterung nicht hinaus. Weitergehende Kompetenzziele im Bereich der Fachkompetenz, Methodenkompetenz und Personale Kompentenz werden häufig nicht erreicht. Der praktische Teil der Veranstaltung wird durch Formulierung papierbasierter Aufgaben *(Zeigen Sie eine spezielle Eigenschaft eines Graphen oder Führen Sie den Algorithmus von Kruskal von Hand aus, Entwerfen Sie einen Algorithmus für...)* oder von Programmieraufgaben *(Implementieren Sie den Algorithmus von Dijkstra)* durchgeführt. Das Feedback zur Bearbeitung der Aufgaben wird in der persönlichen Betreuung gegeben. Sowohl der theoretische Teil, wie auch der praktische Teil stellt für die Studierenden mit noch wenig Programmierkenntnissen regelmäßig eine Herausforderung dar.

3 Vorgehen und Umsetzung

Für den Durchlauf im Sommersemester 2021 wurden die bereitgestellten Materialien (Texte, Bilder, selbstproduzierte Videos) in einer auf Open edX basierenden Lernplattform [MH21] gesammelt und mit elektronischen Übungen zur freiwilligen Selbstreflexion gebündelt. Die organisatorische Unterstützung und Kommunikation zwischen Studierenden und Lehrenden wurde auf Basis von MS Teams durchgeführt. Für die Durchführung der Veranstaltung nach der Scrum-Methodik müssen die aus Scrum bekannten Rollen, Artefakte, Ereignisse und Abläufe in die Hochschullehre übertragen werden. Dabei gilt es den formalen Rahmen der Lehrplanung im Hochschulkontext zu berücksichtigen. Die in Scrum üblichen Daily-Standups müssen in die üblicherweise wöchentlich geplanten Veranstaltungstermine integriert werden. Die Inhalte der Lehrveranstaltung müssen aufgeteilt werden, so dass der Grundgedanke der empirischen Prozesskontrolle tatsächlich wirksam wird.

Mit diesem im Sinne einer Grundlagenveranstaltung ganzheitlichen Ansatz der Themenbearbeitung unterscheidet sich das Vorgehen vom zuvor erwähnten eduScrum, bei dem typischerweise einzelne Projektthemen bearbeitet werden. Im Falle der vorliegenden Veranstaltung wurden die Inhalte des Semesters vom Lehrenden in 3 Themenblöcke oder Lernsprints eingeteilt, die jeweils aus theoretischen Inhalten und praktischen Aufgaben bestehen. Kern des Scrum-Ansatzes ist das zyklische Vorgehen in 3 - 4 wöchigen Sprints bestehend aus **Sprint-Planung** *(Was muss erarbeitet werden?)*, **Sprint-Durchführung** *(Eigenständige Erarbeitung der Inhalte)*, **Review** *(Reflexion der Inhalte, z. B. unterstützt durch Selbstlern-Quizze in der Lernplattform)* und **Retrospektive** *(Wie haben wir gearbeitet und was können wir verbessern?)* [DL18]. Mit dem Ansatz werden beide Teile der Veranstaltung - Theorie und Praxis - in konsequente Selbstlernarrangements überführt. Die freiwerdende Präsenzzeit wird für interaktive Formate (z.B. Diskussion, Klärung von Fragen) sowie individuelle Unterstützung verwendet. Der grundsätzliche Semesterablauf ist in Abb. 1 dargestellt.

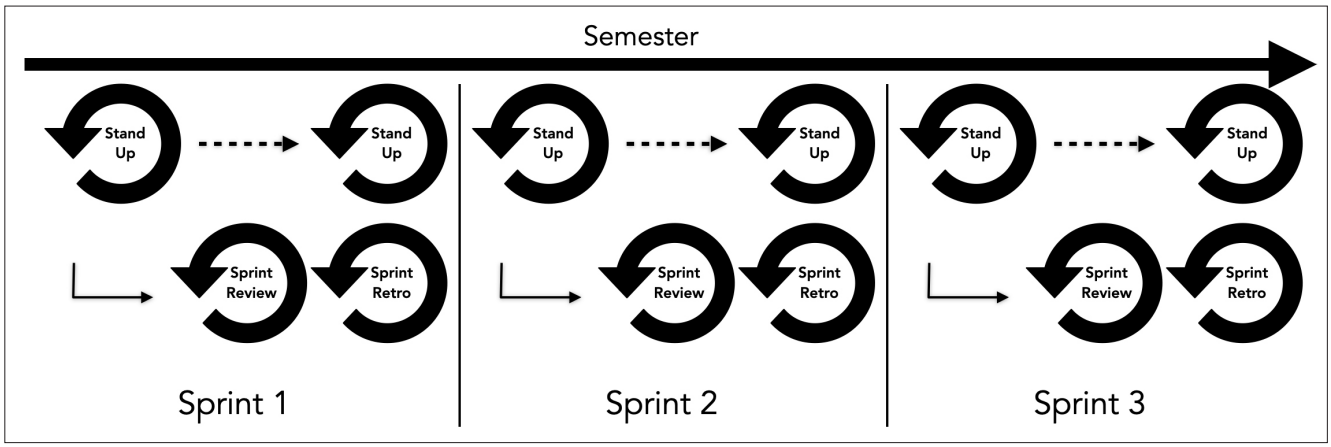

Abb. 1: Lernsprints im Semesterablauf

Zu Beginn des Semesters wurden die Studierende in Lernteams zu 3-4 Personen eingeteilt, um die Lernsprints ganz im Sinne von Scrum gemeinsam als Scrum-Team zu bearbeiten. Der Lehrende gestaltet als *Product Owner* den inhaltlichen Rahmen der einzelnen Sprints, legt die Lernziele fest und definiert auf diese Weise das *Lern-Backlog* (in Analogie zum Produkt-Backlog bei Scrum). Ein Studierender übernimmt mit Unterstützung der Lehrperson die Rolle des *Scrum-Masters*, um den Scrum-Ablauf einzuhalten, ist aber selbst auch Teil des Scrum-Teams, welches sich nach eigenem Ermessen die Arbeit der Theorie- und Praxiselemente aufteilt. Zur Unterstützung der kollaborativen Arbeit der Teams wird ein elektronisches Whiteboard (Nutzung von Miro) eingesetzt. Während der Sprint-Planung wird ein Planungsboard (siehe Abb. 2 für ein Beispiel) nach dem Kanban-Prinzip in Miro entwickelt und kontinuierlich gepflegt. Der Aufbau orientiert sich an dem Planungsflap aus eduScrum.

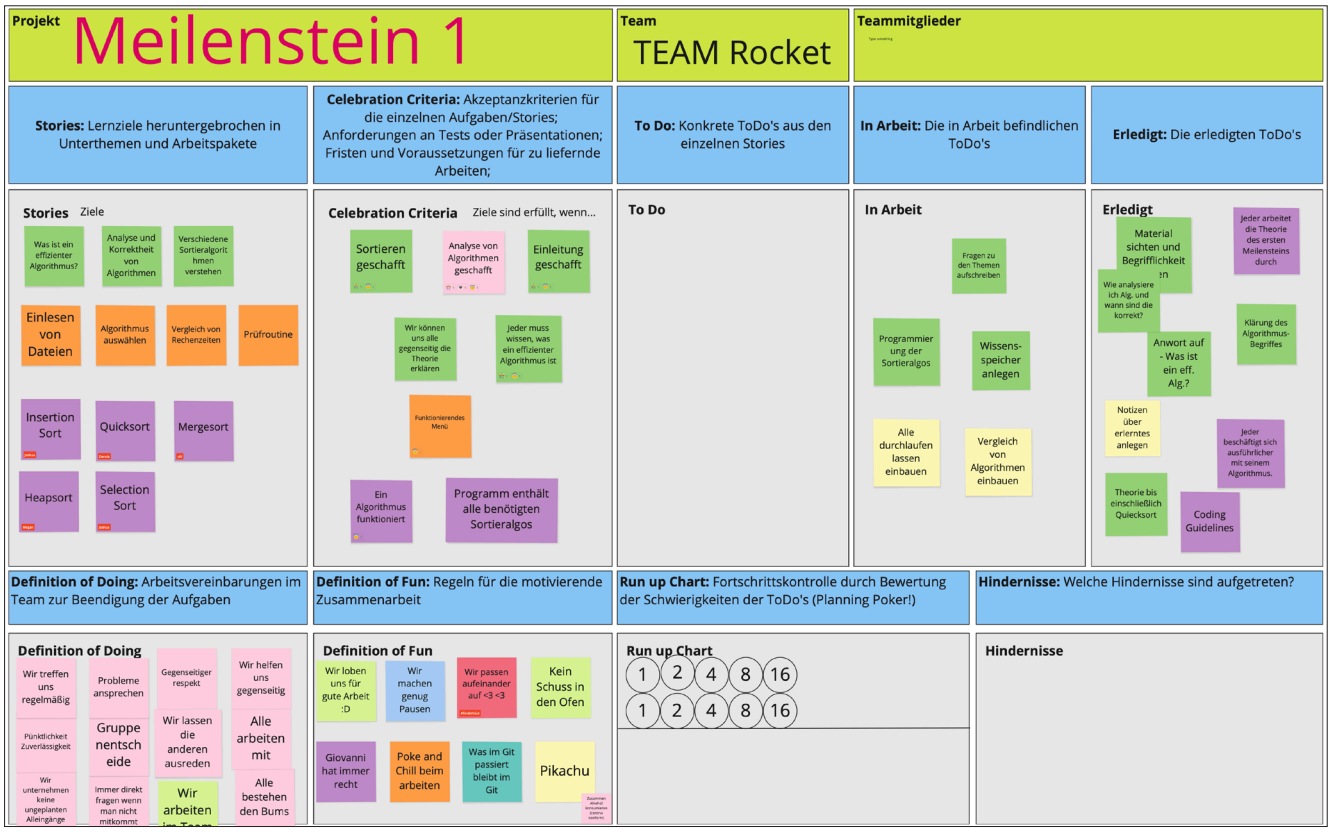

Abb. 2: Planungsboard eines Teams

Die groben Lernziele eines Sprints werden zunächst als *Stories* formuliert. Diese werden dann von den Studierenden-Teams in konkrete Aufgaben aufgeteilt und hinsichtlich des Aufwandes bewertet. Der Lernfortschritt wird durch das Pflegen eines Burn-Down-Charts auf Basis der geschätzten Aufwände beobachtet. Mit dieser Komponente werden die Studierenden an das Thema *Aufwandsschätzung* herangeführt, welche auf Basis von Planning-Poker durchgeführt wird. Eingerahmt wird dieser fachliche Aspekt der Sprint-Planung durch Regeln für die Zusammenarbeit des Teams *(Definition of Fun)* sowie Kriterien für den Abschluss des Sprints *(Definition of Doing, Celebration Criteria)*.

Der praktische Teil eines Sprints (Programmieraufgaben in Java) wird mit Hilfe von IntelliJ, GitLab und dem Build-Tool Gradle unterstützt. Ziel ist hierbei, die Studierenden zu einem frühen Zeitpunkt im Studium mit professionellen Entwicklungswerkzeugen und -methodiken wie Continuous Integration und Unit Testing zu konfrontieren. Auch diese praktischen Aufgaben im Rahmen eines Sprints werden auf dem Planungsboard festgehalten.

Die bei Scrum verwendeten Daily-Standups lassen sich im Hochschulkontext nur schwer realisieren, da Lehrveranstaltungen üblicherweise wöchentlich geplant werden. Sie werden daher im Rahmen der regulär eingeplanten Lehrveranstaltung zu Beginn einer Sitzung durchgeführt. Neben der Abstimmung im Team werden Fragen vom Team gesammelt, die dann in der anschliessenden Sitzung mit dem Lehrenden diskutiert werden.

Zum Ende eines Lernsprints werden das praktisch entwickelte Programmierprojekt im Rahmen eines Sprint-Reviews (*Was wurde erreicht?*) begutachtet und der gewählte Lösungsansatz mit den Studierenden diskutiert. Die anschließende Retrospektive (*Wie haben wir zusammengearbeitet und was können wir verbessern?*) dient zur kritischen Selbstreflexion der Arbeit im Team. Diese

beiden wichtigen Schritte im Scum-Vorgehen erfolgen individuell mit jedem Entwicklungsteam. Im Sinne der empirischen Prozesskontrolle werden die in der Retrospektive identifizierten Verbesserungspotentiale in die Planung des nachfolgenden Sprints übernommen. In Abb. 3 ist das Ergebnis einer Team-Retrospektive dargestellt.

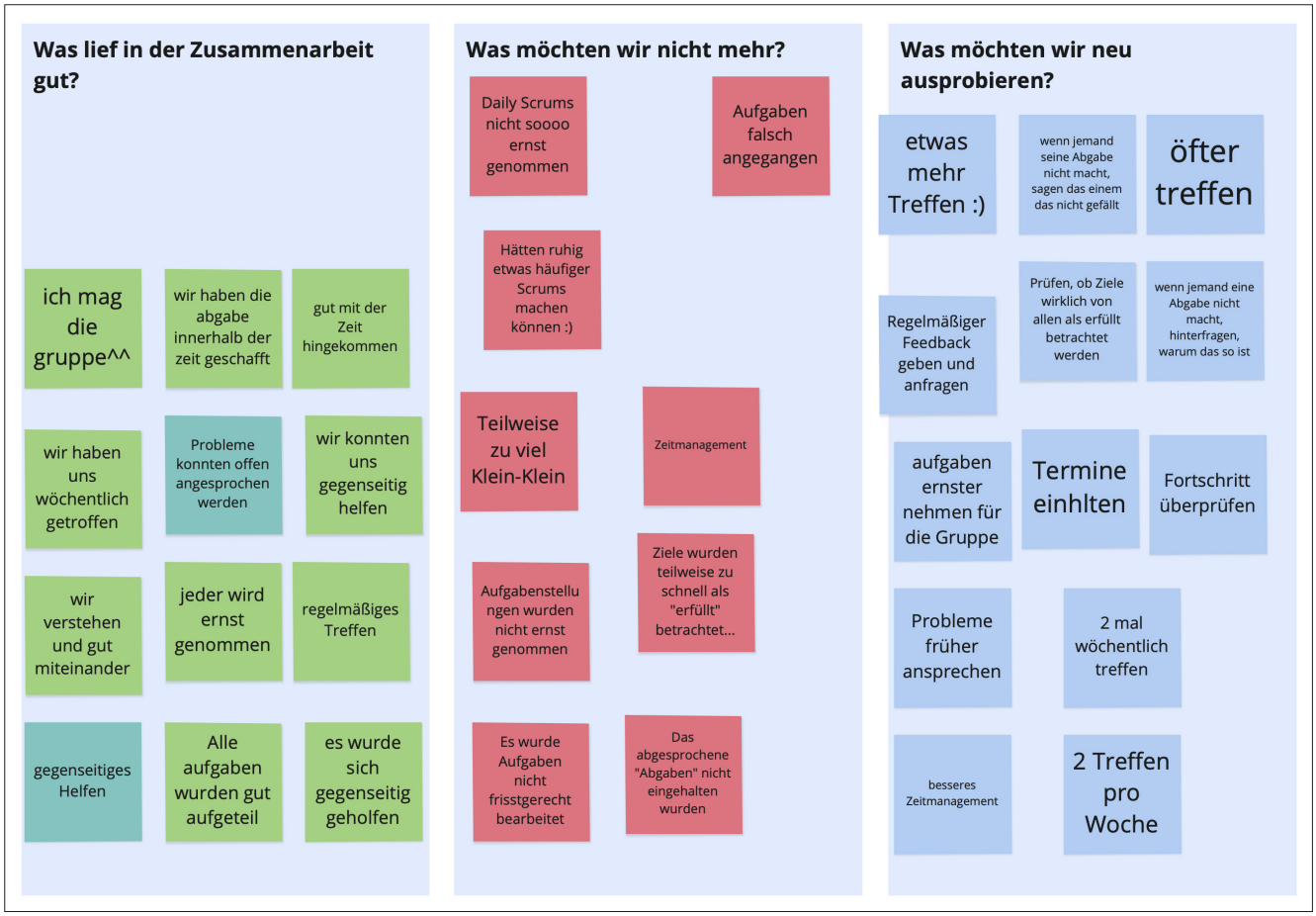

Abb. 3: Ergebnis einer Team-Retrospektive

4 Kritische Reflexion

Für das beschriebene Lehrveranstaltungskonzept können wir noch keine umfassenden empirischen Daten vorweisen. Stattdessen wollen wir die gesammelten Erfahrungen aus Sicht der durchführenden Lehrenden und in einer an Teaching Analysis Poll [Ha16] angelegten Befragung und Diskussion mit den Studierenden reflektieren.

Aus Sicht der Lehrenden bedeutete die Zusammenstellung des Materials für die beiden Teile Theorie und Praxis einen erheblichen Aufwand. Auch wenn das Videomaterial in Form von segmentierten Vorlesungsaufzeichnungen vorhergehender Semester weitestgehend vorlag (ca. 120 selbstproduzierte Videos auf YouTube), war die Zusammenstellung in der Lernplattform in Kombination von Text- & Bildmaterial sowie elektronischer Übungen eine hohe zeitliche Belastung. Gleiches gilt für den praktischen Teil der Veranstaltung. Die Studierenden mussten an eine neue Entwicklungsumgebung (IntelliJ) herangeführt werden, die Nutzung eines Versionierungssystems (GitLab) war ebenso Neuland wie der Umgang mit einem Build-Tool (Gradle).Weil die Studierenden außerdem einen Großteil der Informationen durch bereitgestellte Materialien erhalten und nicht direkt von den Lehrenden, können technische Probleme mit dem Material erst auf Rückfrage behandelt werden. Dementsprechend bedarf es schnellen und direkten Kommunikationswegen zwischen Lernenden und Lehrenden. Dies konnte durch die Kommunikationsplattform MS Teams sehr gut abgebildet werden. In wiederholten Durchläufen können Probleme dieser Art allerdings durch inkrementelle Verbesserung deutlich reduziert werden. Die initial investierte Zeit lässt sich durch die Wiederverwendbarkeit der Materialien ausgleichen.

Die freie, selbstständige Gestaltung der Lernphase ist für die Studierenden eine neue Erfahrung. Das erste Semester besteht aus sehr stark instruktionsorientierter Lehre, so dass die klare Kommunikation des Veranstaltungsablaufes in den ersten Veranstaltungswochen eine besonders wichtige Bedeutung bekommt. Der Zweck der im Zusammenspiel eingesetzten und meist neuen Werk-

zeuge muss wiederholt deutlich gemacht werden. Wenn dies aber von den Studierenden verstanden wird, wird ein deutlicher Mehrwert in der kollaborativen Arbeit und auch mit Blick auf zukünftige Semester von den Studierenden gesehen. Die intensive Zusammenarbeit innerhalb der Studierendenteams sowohl für den theoretischen Teil der Veranstaltung wie für die praktischen Entwicklungsarbeiten führt zu einer Kompetenzförderung über die basale Fachkompetenz hinaus. Insbesondere der teaminterne Dialog befördert die fachliche Kommunikationskompetenz. Zudem trägt das Prinzip des Docendo discimus zum fachlichen Austausch innerhalb des Teams zur Festigung der erarbeiteten Lehrinhalte bei.

Sowohl in den individuellen Retrospektiven der Sprints als auch in der Befragung und Diskussion mit der gesamten Semestergruppe haben die Studierenden sehr positive Rückmeldungen gegeben. Zwar wurde von einigen Studierenden eine erhöhte zeitliche Belastung empfunden, allerdings wurden dafür Punkte wie gute Organisation und ein hoher Teil an praktischen Aufgaben positiv herausgestellt. Die freigewordene Zeit während der regulären Vorlesungszeit wird in den individuellen Lernsupport investiert. Dies wurde von den Studierenden erkannt und sehr deutlich positiv hervorgehoben. Die Mischung aus eigenverantwortlich zu gestaltenden Selbstlerneinheiten in der Gruppe gekoppelt mit einer klaren und straffen Rahmenorganisation (Klare zeitliche Lernsprints, Motivation der kontinuierlichen Arbeit) ist für die Studierenden von wichtiger Bedeutung.

5 Fazit und Ausblick

In zunehmendem Maße werden von Hochschulabsolvent*Innen über die eigentliche Fachkenntnisse hinausgehende Kompetenzen gefordert. Diese werden in der aktuellen Diskussion häufig als Future Skills [Hi19] bezeichnet. Mit der Erfüllung und Bedienung kurzfristiger Anforderungen aus dem Arbeitsmarkt tun sich Hochschulen als in Teilen behäbige Organisationen sehr schwer. In der Informatik, die als Fachdisziplin mit teilweise sehr schnellen Entwicklungen konfrontiert ist bzw. diese gestaltet, ist die Aktualität eines Studienganges eine große Herausforderung. Neben all den fachbezogenen Kompetenzen zusätzlich die geforderten Zukunftskompetenzen zu adressieren, ist nahezu unmöglich. Hier kann das vorgestellte integrierte, agile Lehrkonzept einen Beitrag leisten. Die Umstellung der eher theorielastigen Veranstaltung A+D auf ein Format, welches neben den Fachinhalten die geforderten Future Skill adressiert, ist für Lehrende und Studierende eine große Herausforderung. Die ersten Rückmeldungen von Studierenden und Diskussionen mit anderen Lehrenden zeigen aber, dass von allen Beteiligten großes Potential in diesem Format gesehen wird.

Ob eine direkte Messbarkeit im Sinne von Lernerfolg kurzfristig im Rahmen der Semesterabschlussprüfung erkennbar sein wird, bleibt abzuwarten. Viel wichtiger erscheint aber die nachhaltige Wirkung im Rahmen der zukünftigen Lehrveranstaltungen. Dies gilt es zu beobachten.

Literatur

[Br20] Brandhofer, G.; Buchner, J.; Freisleben-Teutscher, C.; Tengler, K., Hrsg.: Tagungsband zur Tagung Inverted Classroom and beyond 2020, 2020.

[BS18] Biemann, S.; Schmidt, R.: Kreativitätsförderung 4.0 Kombination der Inverted-Classroom-Methode mit dem Einsatz von Tablets in kunstdidaktischen Seminaren. In (Buchner, J.; Freisleben-Teutscher, C.; Haag, J.; Rauscher, E., Hrsg.): Inverted Classroom - Vielfältiges Lernen. FH St. Pölten, S. 39–46, Feb. 2018.

[Bu18] Buchner, J.; Freisleben-Teutscher, C.; Haag, J.; Rauscher, E., Hrsg.: Inverted Classroom - Vielfältiges Lernen, FH St. Pölten, Feb. 2018.

[BW18] Bernhofer, A.; Wieland, E.: Unterrichtsmodelle des Flipped Classroom für die Sing- und Musizierpraxis im Musikunterricht. In (Buchner, J.; Freisleben-Teutscher, C.; Haag, J.; Rauscher, E., Hrsg.): Inverted Classroom - Vielfältiges Lernen. FH St. Pölten, S. 29–38, Feb. 2018.

[Co09] Cormen, T.; Leiserson, C.; Rivest, R.; Stein, C.: Introduction to Algorithms. MIT Press, 2009. [DL18] Derby, E.; Larsen, D.: Agile Retrospektiven. Franz Vahlen GmbH, 2018.

[DL18] Derby, E.; Larsen, D.: Agile Retrospektiven. Franz Vahlen GmbH, 2018.

[Ha16] Hawelka, B.: Handreichung zur Kodierung qualitativer Evaluationsdaten aus Teaching Analysis Poll, Universität Regensburg: Zentrum für Hochschul- und Wissenschaftsdidaktik, 2016.

[Hi19] Hirchherr, J.; Klier, J.; Lehmann-Bruns, C.; Winde, M.: Future Skills: Welche Kompetenzen in Deutschland fehlen, Diskussionspapier 1, Essen: Stifterverband für die Deutsche Wissenschaft e.V., 2019.

[HSB19] Hosser, D.; Schröder, J. M.; Beller, J.: LiteraTUs: ein Lehr-Lern-Konzept zum wissenschaftlichen Arbeiten und Schreiben. In (Kauffeld, S.; Othmer, J., Hrsg.): Handbuch Innovative Lehre. Springer Fachmedien Wiesbaden, Wiesbaden, S. 115–126, 2019.

[KO19] Kauffeld, S.; Othmer, J., Hrsg.: Handbuch Innovative Lehre.Wiesbaden: Springer Fachmedien Wiesbaden, 2019, 470 S.

[Ku17] Kultusministerkonferenz: Qualifikationsrahmen für deutsche Hochschulabschlüsse, Dez. 2017.

[MH21] Morisse, K.; Heidemann, C.: Algorithmen und Datenstrukturen, 2021, url: https://edu.hs-osnabrueck.de/courses/course-v1:HSOS+AuD21_MI_KM+AuD21_km/about.

[Mo19] Morisse, K.: Inverted Classroom in der Informatik: ein Ansatz zum Erwerb überfachlicher Kompetenzen. In (Kauffeld, S.; Othmer, J., Hrsg.): Handbuch Innovative Lehre. Springer Fachmedien Wiesbaden, Wiesbaden, S. 99–114, 2019.

[SS21] Sutherland, J.; Schwaber, K.: Scrum Guide, 2021, url: https://scrumguides.org.

[SW14] Sedgewick, R.; Wayne, K.: Algorithmen. Pearson Deutschland GmbH, 2014.

[TN86] Takeuchi, H.; Nonaka, I.: The New Product Development Game. Harvard Business Review 64/1, S. 137–146, Jan. 1986.

[WS19] Wijnands, W.; Stolze, A.: Transforming Education with eduScrum. In (Parsons, D.; MacCallum, K., Hrsg.): Agile and Lean Concepts for Teaching and Learning. Springer Nature Singapore, 2019.

E-Portfolios in der Informatik-Lehrkräftebildung: Studierende bloggen über Internet-of-Things Projekte

Anatolij Fandrich[1], Nils Pancratz[1], Ira Diethelm[1]

Abstract:
Über die Vision des Internet-of-Things (IoT) lässt sich in der Hochschullehre eine Vielzahl von Informatikinhalten im Kontext eines aktuellen Themas aus Industrie und Forschung kombinieren und vermitteln. Um dieses inhaltliche Potential zu nutzen, lernen angehende Informatiklehrkräfte an der Universität Oldenburg im Rahmen eines Seminars IoT-Technologien und deren Einsatz kennen. Zu diesem Seminar gehört auch die Entwicklung und Dokumentation eines eigenen digitalen Artefakts, welches die Inhalte der Lehrveranstaltung in einem praktischen Kontext aufgreift. Die Dokumentation des Abschlussprojektes erfolgt online in einem öffentlich zugänglichen Web-Blog, in dem die Studierenden ihre Motivation für die Entwicklung, den technischen Aufbau ihres Systems und mögliche Anknüpfungspunkte zum Informatikunterricht beschreiben. Zudem wird der eigene Lernfortschritt regelmäßig in einem Lerntagebuch reflektiert. In diesem Beitragwerden die didaktischen Überlegungen zum Aufbau dieses Seminars beschrieben und erste praktischen Erfahrungen mit WordPress als Portfolio-Plattform dargestellt.

Keywords:
Internet-of-Things, E-Portfolio, Lehrkräftebildung, Physical Computing, Digital Fabrication

1 Einleitung und Related Work

Das *Internet-of-Things* (IoT), welches u. a. die zunehmende Vernetzung zwischen smarten Objekten innerhalb und außerhalb des Internets beschreibt, findet vor allem durch den Themenkomplex *Smart-Home* immer mehr Einzug in die Lebenswelt der Schülerinnen und Schüler. Das IoT ist somit als *relevanter Kontext* für den heutigen Informatikunterricht zu sehen [DBW10] - sowohl in der Schule, als auch in der akademischen Informatiklehrkräftebildung.

Als Querschnittsthema ermöglicht IoT einen spannenden und aktuellen Kontext für projektbasierten Unterricht, um verschiedene Informatikinhalte zu vermitteln [Bu18]. Daher wurden zwei Lehrveranstaltungen *(Internet-of-Things und Smart-Home im schulischen Kontext und Physical Computing und Digital Fabrication im Informatikunterricht)* im Umfang von je drei Kreditpunkten an der Universität Oldenburg entwickelt. Angehende Informatiklehrkräfte (Lehramt für Gymnasien und Wirtschaftspädagogik) müssen im Rahmen ihres Master-Studiums eine der beiden Veranstaltungen als Pflichtmodul belegen. Seit dem Wintersemester 2018/2019 haben bislang insgesamt 14 Master-Studierende die Lehrveranstaltung absolviert und ihre Prüfungsleistungen über der Erstellung von *E-Portfolios* erbracht. Im Rahmen der Lehrveranstaltung sammeln die Studierenden praktische Erfahrungen mit aktuellen Technologien, wie z. B. 3D-Druck, parametrischem Design, hardwarenaher Systementwicklung und Lasercutting. In E-Portfolios dokumentieren und präsentieren die Studierenden ihre Abschlussprojekte und reflektieren ihre Lernprozesse während der Entwicklung. Gerade in Zeiten von Social Distancing und Home-Schooling gewinnen digitale Formen der Prüfungsleistung immer mehr an Bedeutung. Deshalb stellen wir mit diesem Beitrag unsere Erfahrungen aus drei Iterationen vor, in denen wir die Lehrveranstaltungen angeboten und Studierende E-Portfolios erstellt haben.

Bei der Entwicklung des in diesem Beitrag vorgestellten Lehrkonzepts konnte u. a. auf den Erfahrungen von Mäenpää et al. [Mä17] aufgebaut werden, die von Möglichkeiten berichten, einen IoT-Kurs im Sinne problembasierter Szenarien aufzubauen und dabei Lernziele zu verfolgen. Sie fassen ihre Erfahrungen aus drei Jahren Lehre mit IoT-Prototypen zusammen. Ein besonderer Fokus liegt auf einer praktischen und personalisierten Lernumgebung. Silvis-Cividjian [Si19] beschreibt ebenfalls die Struktur ihres IoT-Kurses, der einen starken Fokus auf einen industriellen und nicht auf einen persönlichen Kontext hat. Die Projektphasen sind deutlich länger geplant als beispielsweise bei Mäenpää et al. und als Ergebnis wurde festgehalten, dass der IoT-Kurs ein Erfolg war und sich

1 Universität Oldenburg, Didaktik der Informatik, Uhlhornsweg 84, 26129 Oldenburg, {anatolij.fandrich,nils.pancratz,ira.diethelm}@uni-oldenburg.de

positiv auf die Selbstwirksamkeitserwartung ausgewirkt hat, da sich die Studierenden nach der Lehrveranstaltung zutrauen, weitere reale Probleme lösen zu können. Im Gegensatz zu den anderen Beiträgen, in denen der Aufbau von IoT-Lehrveranstaltungen beschrieben wird, verwenden Barkmin und Brinda [BB18] zusätzlich ein E-Portfolio als Prüfungsleistung. Dies kommt unserem eigenen Ansatz am nächsten. Als Plattform wurde Mahara gewählt und auch hier sind die Erfahrungsberichte der Studierenden positiv.

Safran [Sa08], sowie Tur und Marin [TM13] haben zwar keinen Bezug zum IoT in der Bildung, aber ihre Forschung bestätigt, dass Studierende eine positive Einstellung gegenüber Technologie in der Lehre und E-Portfolios zur Dokumentation von Lernprozessen haben. Darüber hinaus reflektieren Studierende, die ihre aktuelle Arbeit bloggen, aktiver und können so ihre praktische Arbeiten verbessern [Sa08]. Die Vorteile von 3D-Modellierung und parametrischem Design zum Verständnis ingenieurwissenschaftlicher Konstruktionsprinzipien haben Chytas et al. [CDT18] in ihrer Forschung aufgezeigt.

Im Folgenden werden daher zwei Lehrveranstaltungen beschrieben, die Vorzüge des parametrischen Designs, der thematischen Vielfalt des IoT und die positiven Aspekte von E-Portfolios als Prüfungsleistung in der Informatiklehrkräftebildung vereinen.

2 Inhalt der Lehrveranstaltungen

Im Folgenden werden die Lernziele und der Aufbau der Seminare dargestellt. Bei dem Seminar handelt es sich um ein Pflichtmodul für angehende Informatiklehrkräfte und die Lerngruppen sind mit vier bis fünf Studierenden pro Semester relativ klein. Die meisten Studierende haben keine Vorerfahrung mit Physical Computing, Digital Fabrication oder speziell Internet-of-Things. Zudem liegen die letzten praktischen Programmierübungen in den Hochschulbiografien der Teilnehmenden einige Semester zurück.

Daher sind folgende Lernziele für die Veranstaltung formuliert: Die Studierenden ..

.. erkunden den kreativen Umgang mit modernen Informatiktechnologien in einem praktischen Kontext.

.. planen und entwickeln eigene digitale Artefakte.

.. wenden fachliche Inhalte aus der theoretischen, praktischen, angewandten und technischen Informatik an.

.. beschreiben und bewerten ihren Lernfortschritt.

Ein Teil der Prüfungsleistung ist die Dokumentation und Präsentation eines Abschlussprojektes, in dessen Rahmen die theoretischen Inhalte des Semesters in einem praktischen Kontext erprobt werden, indem die Studierende ein (IoT)-Projekt ihrer Wahl umsetzen. Die einzige Einschränkung ist der Einsatz von netzwerkfähigen Mikrocontrollern oder Einplatinencomputern. In späteren Seminaren wurde zudem ein selbst entworfenes Artefakt aus dem 3D-Drucker oder Laser-Cutter gefordert, um die Elektronik einbetten zu können. Webseiten, wie z. B. instructables.com oder thingiverse.com konnten als Inspirationen und Vorlage für die Projekte genutzt werden. Zudem wurden diverse Ausgaben der Zeitschrift *Make:* verliehen.

2.1 Internet-of-Things und Smart-Home im schulischen Kontext

Vor einer Überarbeitung des Lehrkonzepts haben alle Studierenden ein identisches Abschlussprojekt (Bluetooth-fähiger LED-Controller) angefertigt und eine Unterrichtseinheit in einem Langentwurf beschrieben. Dies wurde verworfen, um den Studierenden neue Eindrücke über die Vielseitigkeit der Informatik zu zeigen und Raum für das persönliche Interesse und der kreativen Entfaltung zu schaffen. Daher wurde IoT als Querschnittsthema gewählt und der Langentwurf als Prüfungsleistung abgeschafft. Stattdessen wurden die Projekte auf einem Poster dokumentiert und in einer Poster-Session den Mitzustudierenden präsentiert. Zur Vermittlung wissenschaftlicher Kommunikationskompetenzen wurden die Projekte und Poster Schülerinnen und Schülern im Rahmen eines jährlich fakultätsweit angebotenen Informatik-Hochschulinformationstages präsentiert und zum Ende des Semesters in den Räumlichkeiten des Departments aufgehangen. Die Inhalte des Seminars ist in Tabelle 1 zusammengefasst.

Zu Beginn haben die Studierenden diverse IoT-Funkstandards miteinander verglichen, um zum einen die technischen Möglichkeiten der drahtlosen Kommunikation kennenzulernen und zum anderen Standards für den schulgerechten Einsatz zu identifizieren. In der nächsten Sitzung wurden (physikalische) Kenngrößen (u. a. Stromstärke, Spannung und Widerstand), sowie eine Auswahl passiver und aktiver Bauteile vorgestellt, damit die Studierenden später bei der Recherche nach eigenen Projekte einfache Schaltpläne aus dem Internet verstehen und nachbauen können. In einer anschließenden Seminarsitzung zum Thema Löten und Arbeits-

Tab. 1: Inhalte der Veranstaltung *IoT und Smart-Home im schulischen Kontext*

Sitzung	Thema	Inhalt
1	Einführung	Kurze Vorstellung des Seminars und Lerninhalte. Details zur Prüfungsleistung und Bewertung. Hardwarevorstellung und Verwendungsmöglichkeiten.
2	Mikrocontroller und drahtlose Kommunikation	Mikrocontroller Einführung und Vergleich diverser Funkstandards für den Einsatz in der Schule, wie z.B. BLE, Mobilfunk, Zigbee, Z-Wave, LoRaWAN und WiFi. Verglichen wurden u.a. Reichweite, Einsatzbereiche, Kosten und Tauglichkeit im Klassenzimmer.
3	Einführung Elektrotechnik	Einführung der Kenngrößen Spannung, Strom und Widerstand und Vorstellung gängiger passiver und aktiver Bauteile. Schaltplan lesen und Ströme berechnen. Die Sitzung endete mit einer ersten praktischen Übung.
4	Löten	Arbeitssicherheit, bewährte Verfahren und Lötübungen auf einer Lochrasterplatine und Widerständen. Zum Ende der Sitzung haben die Studierenden eine Platine mit einer blinkenden LED gefertigt und nach Hause genommen.
5	Mikrocontroller Programmierung	Einführung des ESP8266 Entwicklungsboards NodeMCU und der Entwicklungsumgebung Arduino IDE. Grundfunktionen des Arduino Frameworks erprobt und einfach Übungen durchgeführt.
6	Server Client Kommunikation	Mikrocontroller in WLAN einbinden und TCP Pakete an einen anderen Mikrocontroller senden. Ein Board ist der Client und der andere ist der Server. Ziel der Einheit ist die Steuerung einer LED über das Netzwerk.
7	REST	Vorstellung des REST Paradigma und praktische Übungen zu den Methoden GET, POST und DELETE.
8	MQTT	Einführung in MQTT und praktische Übungen zu veröffentlichen, abonnieren, Qualitätsklassen und Testament. Zudem wurde das Smartphone in die Übung integriert.
9	UML und Blynk	Zustandsdiagramme zum Strukturieren von Programmen und Übersetzen in Code. Zudem eine Einführung in Blynk.
10	Reflexion und Start Semesterprojekt	Reflexion der Seminarinhalte und Einbettung in den Informatikunterricht. Hardware für Semesterprojekte herausgegeben.
11-13	Projektphase	Übersicht zum aktuellen Stand des Projekts, aktuelle Probleme und geplantes Vorgehen und Meilensteine für die kommende Woche.
14	Abschluss	Präsentation der Semesterprojekte (10 Minuten) und Diskussion (5 Minuten). Danach Feedback zur Lehrveranstaltung.

sicherheit erwarben die Studierenden die motorischen Fähigkeiten, die für das selbständige Verlöten von Bauteilen erforderlich sind. Diese Kenntnisse wurden im nächsten Seminar aufgegriffen, um die auf dem WiFi-fähigen Mikrocontroller ESP8266 basierenden Entwicklungsboards NodeMCU für die weiteren Seminare vorzubereiten. Die Wahl fiel auf diesen Mikrocontroller, da dieser kostengünstig ist und vom Arduino-Framework unterstützt wird, wodurch zahlreiche Beispielprojekte zugänglich sind. Die Arduino Entwicklungsumgebung eignet sich aufgrund ihrer schlichten Oberfläche und diversen Bibliotheken gut für den Einstieg in die hardwarenahe Programmierung. In den nächsten Seminaren haben die Studierenden in praktischen Übungen IoT-gängige Kommunikationsprotokolle und Paradigmen (wie z. B. MQTT und REST) kennengelernt und untereinander Daten ausgetauscht. In der letzten praktischen Übung wurden UML-Zustandsdiagramme wiederholt, um den Studierenden später beim Strukturieren ihres Programms zu helfen. Aus den Zustandsdiagrammen wurde zudem Quellcode übersetzt, indem jeder Zustand als while-Schleife modelliert wurde. Die Übung wurde mit einem Blynk Projekt abgeschlossen. Blynk ermöglicht es, mit wenigen Klicks eine App zu erstellen und diese u. a. mit internetfähigen Mikrocontrollern zu verknüpfen. In den folgenden Seminaren wurde der Lernfortschritt reflektiert und die Einsatzmöglichkeiten im Unterricht diskutiert. Für die letzten Sitzungen waren die Studierenden freigestellt - bei Bedarf konnten Fragen vor Ort geklärt werden.

Links in der Abbildung 1 ist ein Projekt eines Studenten dargestellt, welcher mit dem oben beschriebenen Lehrkonzept unterrichtet wurde. Der Student wollte für den Unterricht in seinem Zweitfach Chemie ein smartes pH-Meter entwickeln, welches es ermöglicht, eine Titrationskurve in Echtzeit auf die Endgeräte von Schülerinnen und Schülern zu übertragen. Das pH-Meter eröffnet einen Access Point im Klassenzimmer, mit dem sich die Kinder und Jugendlichen via WiFi verbinden können. Im Webbrowser wird dann die Kurve angezeigt. Ein anderer Student hat sich beklagt, dass er öfter vergisst zu prüfen, ob er seine Kaffeemaschine ausgeschaltet hat. Zur Bewältigung dieses lebensweltlichen Anwendungsbezugs entwickelte er im Rahmen des Seminars eine smarte Steckdose, die automatisch eine Warnung an das Smartphone sendet, sobald ein Verbraucher mit der Steckdose verbunden ist und das

Smartphone einen gewissen Radius um den Wohnort verlässt. Per Remote kann das Gerät dann ausgeschaltet und mit einer Web-App der Radius angepasst werden. In der Fußnote[2] sind Fotos der Tafel, Folien, Übungsaufgaben, Beispielprogramme aus dem Semester und die studentischen Poster verlinkt.

Nach dem zweiten Semester mit diesem Lehrkonzept wurden die Poster als Prüfungsleistung durch einen öffentlichen Web-Blog ersetzt. Dies hat den Vorteil, dass auch „Nicht-InformatikerInnen" aus dem sozialen Umfeld der Studierenden (Eltern, Geschwister, Freunde, ..) die Semesterprojekte begutachten und positives Feedback zurückmelden können. IoT und speziell Smart-Home eignen sich durch ihren Alltagsbezug dafür, die Studierenden über ihr Studium sprechfähiger zu machen und so (auch mit Hilfe der Blogs) gute Lehre nach außen zu kommunizieren. Der Web-Blog beschreibt die Entwicklung des Projektes, mögliche Anknüpfungspunkte zum Informatikunterricht und eine Anleitung zum Nachbauen. Design und Layout der Blogs wurden den Studierenden überlassen.

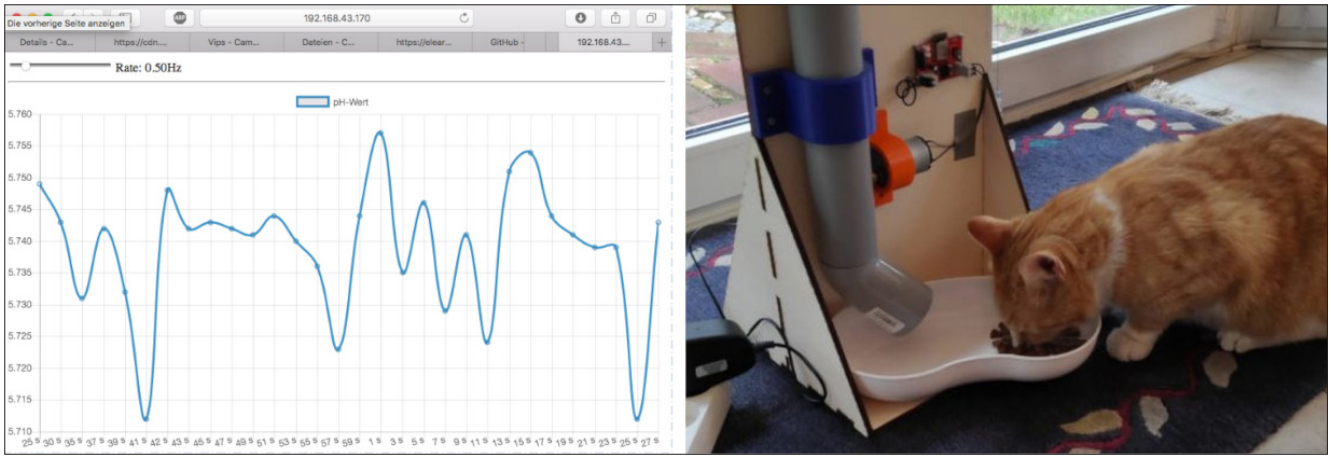

Abb. 1: Beispiele für Abschlussprojekte. Linkes Bild: Screenshot von einem Webbrowser. Das smarte pH-Meter sendet eine Titrationskurve in Echtzeit an das verbundene Endgerät. Rechtes Bild: DIY Katzenfütterungsautomat basierend auf einer Blynk-App und dem ESP8266 Mikrocontroller

Ein Beispielprojekt aus diesem Semester ist rechts in der Abbildung 1 dargestellt. Der Student hat sich zu Beginn des Semesters vorgenommen, einen Katzenfütterungsautomaten zu entwickeln. Die Futterzeitpunkte und Futtermenge lassen sich dabei über eine App einstellen. Auch ein manuelles, ferngesteuertes Füttern ist möglich. Die Hardware basiert auf einer Platine, die gemeinsam mit den Lehrenden entwickelt wurde, einem ESP-12F Modul und einem MOSFET zur Steuerung des DC-Motors. Die Halterung wurde mit einem Lasercutter und 3D-Drucker gefertigt. Weitere Projekte waren z. B. der Prototyp einer Garagensteuerung via App, ein Smart-Mirror oder eine App-gesteuerte Bewässerungsanlage mit integriertem Feuchtigkeitssensor. Sämtliche Projekte wurden auf einem Poster zusammengefasst und mit QR-Codes verlinkt. Auch dieses Poster wurde im Flur des Departments für weitere Studierende zugänglich gemacht. Die Blogs und die Arbeitsmaterialien der Lehrveranstaltung sind in der Fußnote[3] verlinkt.

2.2 Digital Fabrication und Physical Computing im Informatikunterricht

Nach zwei weiteren Semestern wurde der Fokus von IoT auf Physical Computing und Digital Fabrication verlagert, da die Studierenden (auch aus vorherigen Lehrveranstaltungen[4;5]) Schwierigkeiten hatten, Werkstücke aus dem 3D-Drucker oder Laser-Cutter selbstständig zu entwerfen und zu fertigen. Bis zu diesem Zeitpunkt haben die Lehrenden dieWerkstücke nach Spezifikation der Studierenden entworfen und hergestellt. Da das eigene Kompetenzerleben ein übergeordnetes Lernziel des Modulkonzepts ist, wurden die Fähigkeiten zur Fertigung dieser Artefakte im Rahmen des Seminars vermittelt. Daher wurde mehr Zeit zur digitalen Fertigung investiert und Werkzeuge, die sich für den Einstieg in die 3D-Modellierung gut eignen, vorgestellt. Dazu zählen TinkerCAD, BlocksCAD, openSCAD und Fusion 360. IoT-Technologien konnten aus zeitlichen Gründen nur noch in einem kurzen Exkurs behandelt werden. Eine Übersicht der Inhalte dieses Lernkonzepts ist in Tabelle 2 dargestellt. Das Material aus der Lehrveranstaltung und die studentischen Blogs sind ebenfalls auf GitHub einsehbar[6].

2 https://github.com/esdkrwl/Lehre/tree/main/3_2018.19_WiSe

3 https://github.com/esdkrwl/Lehre/tree/main/4_2019_SoSe

4 https://github.com/esdkrwl/Lehre/tree/main/1_2017.18_WiSe

5 https://github.com/esdkrwl/Lehre/tree/main/2_2018_SoSe

6 https://github.com/esdkrwl/Lehre/tree/main/5_2019.20_WiSe

Tab. 2: Inhalte der Veranstaltung *Digital Fabrication und Physical Computing im Informatikunterricht*

Sitzung	Thema	Inhalt
1	Digital Fabrication: TinkerCAD und 3D-Druck	Kurze Vorstellung des Seminars und der bisherigen Semesterprojekte. Einführung in TinkerCAD und des Arbeitsschritte von der Modellierung bis zum 3D-Druck eines Modells. Das allererste Projekt war ein Würfel.
2	blockSCAD und openSCAD	Kurze Einführung in die parametrische Konstruktion. Das Seminarziel war ein parametrischer Stifthalter.
3	Fusion 360	Einführung in Fusion 360, Holzbearbeitung mit dem Lasercutter und parametrisches Design. Am Ende wurde eine Eiswürfelform entworfen und gedruckt.
4	Physical Computing und Arduino Framework	Kleine Programmieraufgaben wie in den Semestern zuvor.
5	MicroBit, Calliope Mini und Löten	Kurze Einführung zur Hardware und den zugehörigen Entwicklungsumgebungen. Zudem Arbeitssicherheit und Lötübungen an einer Lochrasterplatine. Vorlöten der Wemos D1 mini Boards und LEDs für die nächsten Sitzungen.
6	ESP8266 and ESP32	Einführung zu diesen Mikrocontrollern und MQTT.
7	Exkurs Internet-of-Things	Rollenspiel: Wie funktioniert das Internet und der Sprachassistent Alexa? Kleine Übungen mit Wemos Boards, WS2812b LEDs und Openweathermap API.
8	Physical Computing und Digital Fabrication?	Diskussion und Reflexion über die Eignung und Einsatzmöglichkeiten von Physical Computing und Digital Fabrication im Informatikunterricht.
9-12	Projektphase	Übersicht zum aktuellen Stand des Projekts, aktuelle Probleme und geplantes Vorgehen und Meilensteine für die kommende Woche
13	Abschluss	Präsentation der Semesterprojekte (10 Minuten) und Diskussion (5 Minuten). Danach Feedback zur Lehrveranstaltung.

Die Projektdokumentation erfolgt in einem WordPress-Blog, den die Studierenden auf Wunsch nicht-öffentlich schalten konnten. WordPress wurde als CMS-Werkzeug gewählt, da dies nativ in der e-Learning Plattform der Universität Oldenburg integriert ist und sich das Erstellen und Zuordnen der Blogs somit unkompliziert gestaltet. Die Blogs ermöglichen interessierten Personen, die Projekte selbst nachzubauen. Um die Studierenden bei der Strukturierung der Blogs zu unterstützen, wurden mögliche Inhalte vorgeschlagen:

- Alltagsbezug für Studierende oder SuS. Löst das Projekt ein Problem?
- Welche Bereiche der Informatik vernetzt das Projekt oder welche Kompetenzen aus dem Kerncurriculum sind notwendig, um das Projekt zu verstehen und nachbauen zu können?
- Eine Materialliste mit Preisen und Bezugsquellen. Dazu eine Beschreibung der Hardware und eine Begründung der Auswahl.
- Beschreibung der verwendeten 3D-Modelle.
- Fotos und Videos während der Entwicklung und des finalen Artefakts. Zusätzlich eine Beschreibung der Inbetriebnahme.
- Dokumentation der Software und der Werkzeuge.
- Ein Entwicklerblog, in welchem der Projekt- und Lernfortschritt dokumentiert und reflektiert wird. Dieser Blog sollte mindestens fünf Einträge umfassen.
- Kontaktdaten für mögliche Rückfragen.

2.3 Bewertung

Die Prüfungsleistung setzt sich aus drei Teilleistungen zusammen: Abschlussprojekt (Anteil an der Gesamtbewertung: 20%), E-Portfolio inklusive Projektdokumentation (60%) und Abschlusspräsentation (20%). Für Abschlusspräsentationen wurde insgesamt eine Dauer von 15 Minuten festgelegt, wobei fünf Minuten für eine abschließende Diskussionsrunde vorgesehen waren. Das Projekt wurde bewusst geringer gewichtet, damit Studierende die Möglichkeit haben, Risiken einzugehen und Neues zu Probieren, ohne dass sich Fehler oder nur zum Teil erreichte Ziele zu stark auf die Gesamtnote auswirken. Stattdessen wurde Wert auf eine gute Dokumentation und die Reflexion der Lernprozesse gelegt. Studierende, die sich besonders schwierige oder anspruchsvolle Projekte ausgesucht haben, erhalten die bessere Note, sofern sich diese rechnerisch zwischen zwei Noten befindet. Neben der Gesamtnote wurde den Studierenden in schriftlicher Form Feedback zum Projekt und zum Vortrag ausgeteilt.

3 Fazit und Lessons Learned

Wie so häufig steht und fällt das Engagement von Lehrenden, digitale Methoden in ihren Unterricht zu integrieren, mit den strukturellen Begebenheiten. Glücklicherweise funktioniert die Integration von WordPress, welches die Studierenden für die Anfertigung ihrer E-Portfolios verwenden, in die Arbeitsumgebung, die an unserer Universität zur Verwaltung von Lehrveranstaltungen genutzt wird (Stud.IP), sehr gut. So lassen sich schnell und einfach entsprechende Instanzen für die Studierenden einrichten. Der einzige Nachteil, den wir hinsichtlich der technischen Gegebenheiten erfahren haben, ist der Umstand, dass keine Benachrichtigungen erfolgen, sobald die Studierenden ihre Arbeiten aktualisieren. Zwar ist einsehbar, wann die letzte Seite neu angelegt wurde, jedoch fehlt eine Funktion, die eine Änderung der Inhalte auf den Seiten anzeigt. So sind Lehrende, die den Fortschritt ihrer Studierenden fortwährend verfolgen möchten, gezwungen, sich regelmäßig manuell einen Einblick zu verschaffen. Ein weiterer Nachteil ist die eingeschränkte Möglichkeit zum Export der Blogs, der nur ohne eine Einbettung der verwendeten Medien möglich ist.

Aus *organisatorisch-didaktischer* Sicht sind Unsicherheiten auf Seiten der Studierenden bezüglich angemessener Zitierweisen und der Notwendigkeit der Verfassung eines Impressums aufgefallen. Schließlich soll der Blog nach Abschluss der Arbeit öffentlich zugänglich gemacht werden. Unserer Erfahrung nach sollten Lehrkräfte demnach entsprechende Fragestellungen also frühzeitig und einheitlich mit ihren Studierenden klären. Projekte aus vorangegangenen Durchläufen können hierzu als Anhaltspunkte dienen.

Zur Ausstellung vergangener Projekte und zum Hinweis auf die vorherigen Blogs wurden die Semesterergebnisse von unseren Studierenden in den letzten Jahren auch analog auf Postern festgehalten. QR-Codes zum Link zu ergänzenden Informationen in den Blogs haben sich hierbei bewährt. Außerdem dienten die Poster in den vergangen Jahren als Gesprächsanlass zur abschließenden gegenseitigen Vorstellung der Projekte in dem Kurs. Zur Inspiration zukünftiger Studierender und zur nachhaltigen Würdigung vergangener Projektarbeiten soll diese ergänzende Prüfungsleistung auch weiterhin von den Studierenden eingefordert werden.

3.1 Erfahrungsbericht

Unserer Erfahrung nach ist die Motivation der Studierenden höher, da sie neben den schriftlichen Hausarbeiten, die sie häufig noch aus anderen Kursen kennen, in der Anfertigung eines E-Portfolios eine willkommene Abwechslung sehen. Auch für uns Lehrende ist die Bewertung von E-Portfolios eine angenehme Alternative, da den Studierenden durch die Möglichkeiten zur multimedialen individuellen Entfaltung mehr Raum gegeben wird, positiv zu bewertende Inhalte zu produzieren.

Aufgrund der kleinen Lerngruppe konnte die universitätsweit durchgeführten Lehrevaluation nicht genutzt wurden. Stattdessen wurde zum Ende des Semesters das Feedback von allen Teilnehmenden mündlich verfasst und mitgeschrieben.Vor allem wurde der hohe Praxisanteil der Lehrveranstaltung positiv bewertet: Schließlich haben die Studierenden im Rahmen von Modulen oft keine Möglichkeit, die in den Basismodulen erlernten theoretischen Fähigkeiten in einem praktischen Szenario zu erproben[7]. Gerade gegen Ende der universitären Phase der Lehrkräfteausbildung bietet dieses Lehrveranstaltungskonzept eine gute Möglichkeit, informatische Grundlagen kontext- und anwendungsbezogen zu wiederholen und gleichzeitig den Fokus auf schulpraktische Anwendungsmöglichkeiten zu legen. Dies wird von den Studierenden besonders geschätzt. Studierende gaben im Feedback zur Veranstaltung an, dass sie mit den Wahl- und Gestaltungsmöglichkeiten des E-Portfolios und des Semesterprojektes teilweise überfordert waren und so z. B. zu viel Zeit in das Design des Blogs investierten. Daher wurde der Wunsch nach konkreten Vorlagen und Projekten geäußert. Für andere Studierende war es schwierig, den Zeitaufwand für das eigene Projekt abzuschätzen und geeignete Hardware für den Einkauf zu bestimmen. Obwohl für die Bearbeitung des Projektes Präsenzzeiten vorgesehen waren

7 Dies gilt – bedingt durch die Studienverlaufspläne üblicherweise – insbesondere für Lehramtsstudierende und fachbezogene Module.

und keine Leistung in den Semesterferien gefordert wurde, haben einige Studierende angemerkt, dass der Arbeitsaufwand für drei Kreditpunkte zu hoch sei. Trotzdem äußerten sie, Spaß an der Projektarbeit gehabt zu haben.

In Zukunft werden wir den Anteil an Digital Fabrication wieder reduzieren, da sonst für das Physical Computing zu wenig Zeit bleibt. Die Studierenden gaben an, dass die Phase zur 3DModellierung recht lang war und dass ihnen manchmal konkrete Anwendungsmöglichkeiten im eigenen Unterricht fehlen. Eine kurze Einführung in BlocksCAD und OpenSCAD sollte für angehende Informatiklehrkräfte im Bezug auf 3D-Modellierung ausreichen.

Es ist zudem wichtig, genug Zeit für den Einkauf zu planen und mögliche Kosten im Auge zu behalten. In der Praxis hat sich eine Vorlaufzeit von vier Wochen zwischen der Bestellung der Hardware und dem Start des Semesterprojektes bewährt. Das mit Abstand teuerste Projekte (Smarter Spiegel) kostet insgesamt über 200 €. Andere Projekte, wie z.B. der Katzenfütterungsautomat oder die Garagensteuerung beliefen sich auf unter 5€, da kaum Teile nachbestellt werden mussten. Alle Kosten wurden aus Haushaltsmitteln gedeckt. Alternativ hätten auch Studienqualitätsmittel beantragt werden können. Falls eine solche Veranstaltung mit größeren Lerngruppen durchgeführt wird, sollte aber ein festes Budget festgelegt werden. Eine Alternative ist die konkrete Vorgabe bestimmter Hardware.

Insgesamt sind das e-Portfolio in Kombination mit einem Semesterprojekt ein spannendes Lehrkonzept, welches über die Jahre verfeinert wurde. Die Konzeption basiert auf einer theoretischen Grundlage und die Lehramtsstudierenden genießen die praktische Erfahrung am Ende des Studiums, bevor sie das Referendariat an einer Schule beginnen. Aus unserer Sicht haben sich WordPress-Blogs als E-Portfolio in vielerlei Hinsicht bewährt - die Studierenden identifizieren sich mit ihren Blogs und Projekten, sie können die Ergebnisse ihrer Arbeit mit ihrem (Nicht-Informatik)-Umfeld teilen und erhalten Anerkennung für ihre Fähigkeiten. Damit tragen sie nicht nur zu einem positiven Bild der Informatik bei, sondern auch dazu, wie innovative Lehre aussehen kann. Dennoch gibt es Verbesserungspotentiale in der Feinabstimmung der Inhalte und dem empfundenen Arbeitsaufwand.

3.2 Ausblick

Im sozialen Umfeld erfahren die Studierenden nach dem Besuch dieses Seminars und der Fertigstellung des Blogs Anerkennung als Spezialistinnen und Spezialisten, die ihre Fähigkeiten erfolgreich in einer projektbasierten Arbeit unter Beweis gestellt haben. Empirische Forschungen haben in den vergangen Jahren belegt [z. B. Th18], dass Schülerinnen und Schüler ihre Selbstwirksamkeitserwartung steigern, wenn sie positive Erfahrungen in Physical-Computing-Projekten sammeln. Es kann daher davon ausgegangen werden, dass sich auch für die Studierenden, die an unserem Kurs-Konzept teilgenommen haben, ähnliche Effekte einstellen. Empirische Begleitstudien zur Untersuchung dieser Effekte werden aktuell in einer Veranstaltung, in der die positiven Erfahrungen aus der Informatik-Lehramtsausbildung auf die Hochschullehre in den grundständigen Informatik-Studiengängen übertragen werden, mit mehr als einhundert Erstsemester-Studierenden der Studiengänge Informatik und Wirtschaftsinformatik durchgeführt.

Literatur

[BB18] Barkmin, M.; Brinda, T.: Exploring and Evaluating Computing Systems for Use in Learning Scenarios by Creating an E-Portfolio: Course Design and First Experiences. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education. WiPSCE '18, Association for Computing Machinery, Potsdam, Germany, 2018, isbn: 9781450365888, url: https://doi.org/10.1145/3265757.3265792.

[Bu18] Burd, B.; Barker, L.; Perez, F. A. F.; Russell, I.; Siever, B.; Tudor, L.; Mc-Carthy, M.; Pollock, I.: The Internet of Things in Undergraduate Computer and Information Science Education: Exploring Curricula and Pedagogy. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE 2018 Companion, Association for Computing Machinery, Larnaca, Cyprus, S. 200–216, 2018, isbn: 9781450362238, url: https://doi.org/10.1145/3293881.3295784.

[CDT18] Chytas, C.; Diethelm, I.; Tsilingiris, A.: Learning programming through design: An analysis of parametric design projects in digital fabrication labs and an online makerspace. In: 2018 IEEE Global Engineering Education Conference (EDUCON). S. 1978–1987, 2018.

[DBW10] Diethelm, I.; Borowski, C.; Weber, T.: Identifying Relevant CS Contexts Using the Miracle Question. In: Proceedings of the 10th Koli Calling International Conference on Computing EducationResearch.Koli Calling '10, Association for Computing Machinery, Koli, Finland, S. 74–75, 2010, sbn: 9781450305204, url: https://doi.org/10.1145/1930464.1930477.

[Mä17] Mäenpää, H.; Varjonen, S.; Hellas, A.; Tarkoma, S.; Männistö, T.: Assessing IOT Projects in University Education: A Framework for Problem-Based Learning. In: Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track. ICSE-SEET '17, IEEE Press, Buenos Aires, Argentina, S. 37–46, 2017, isbn: 9781538626719, url: https: //doi.org/10.1109/ICSE-SEET.2017.6.

[Sa08] Safran, C.: Blogging in Higher Education Programming Lectures: An Empirical Study. In: Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era. MindTrek '08, Association for Computing Machinery, Tampere, Finland, S. 131–135, 2008, isbn: 9781605581972, url: https://doi.org/10.1145/1457199.1457228.

[Si19] Silvis-Cividjian, N.: Teaching Internet of Things (IoT) Literacy: A Systems Engineering Approach. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSESEET). S. 50–61, 2019.

[Th18] Theodoropoulos, A.; Leon, P.; Antoniou, A.; Lepouras, G.: Computing in the Physical World Engages Students: Impact on Their Attitudes and Self-Efficacy towards Computer Science through Robotic Activities. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education. WiPSCE '18, Association for Computing Machinery, Potsdam, Germany, 2018, isbn: 9781450365888, url: https://doi.org/10.1145/3265757.3265770.

[TM13] Tur, G.; Marin, V. I.: Student Teachers' Attitude towards EPortfolios and Technology in Education. In: Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality. TEEM '13, Association for Computing Machinery, Salamanca, Spain, S. 435–439, 2013, isbn: 9781450323451, url: https://doi.org/10.1145/2536536.2536603.

Peer-Review als Motor für die tiefere Auseinandersetzung

Karsten Weicker[1]

Abstract:
Peer-Reviews werden seit geraumer Zeit in unterschiedlichen Lehrszenarien eingesetzt. In diesem Paper wird untersucht, inwieweit das Peer-Review die Auseinandersetzung mit den Inhalten eines Grundlagenmoduls in einem präsenzfreien Lehrszenario befördern kann. Dabei scheint in den Ergebnissen die Qualität der selbst erstellten Reviews der wichtigste Einflussfaktor für den Lernerfolg zu sein, während Experten-Feedback und weitere Faktoren deutlich untergeordnet erscheinen.

Keywords:
Distanzlehre; Feedback; Diskussionskultur; Peer-Review; Lernerfolg

1 Einleitung

In der ersten Welle der Covid-19-Pandemie im Frühjahr 2020 wurden die Hochschullehrer und Dozenten unmittelbar in Szenarien der Fernlehre katapultiert, was eine ganze Reihe an gewagten Lehrexperimenten nach sich gezogen hat. Dieser Beitrag beschreibt eines dieser Experimente und die daraus gezogenen Lehren.

Der Vorlesungsanteil einer Lehrveranstaltung stellt bei dem Übergang zu Distanzformaten weniger eine Problem dar, können doch mit relativ geringen Bordmitteln Screencasts produziert und so publiziert werden, dass diese asynchron den Studierenden zur Verfügung stehen. Dies kann als ein Schritt in Richtung Flipped-Classroom [We21] sogar mit einem didaktischen Mehrwert einhergehen.

Allerdings hat das begleitende Seminar zu einer Lehrveranstaltung häufig die Funktion der studentischen Reflexion und des Abgleichs des eigenen Verständnisses von Konzepten, Aufgaben und Lösungen mit dem der Kommilitonen. Dies kann in unterschiedlichen Ausprägungen passieren – etwa durch Votieren und Vorrechnen oder durch kollaborative Lerngruppen [We07,We20] –, aber in jedem Fall mit einem hohen Grad an Diskussionskultur. Ausgehend von der infrastrukturell begründeten Befürchtung, dass sich dies nicht so kurzfristig in die Welt der virtuellen Konferenzschaltungen übertragen lässt, wurde ein Alternativkonzept entworfen, in welchem Peer-Reviews die Rolle der Seminare übernehmen.

Dabei versuchen wir aussagekräftige Argumente bezüglich der folgenden Thesen zu bekommen:

F1 Durch den Einsatz von Peer-Reviews kann bei Studierenden eine tiefe Auseinandersetzung mit Inhalten und damit auch gute Prüfungsergebnisse erreicht werden.

F2 Experten-Feedback der Dozenten auf studentische Abgaben ist essentiell für den Lernprozess und die erfolgreiche Prüfungsvorbereitung.

F3 Werden die schwierigeren, spät im Semester gestellten Aufgaben bearbeitet und abgegeben, wirkt sich dies positiv auf die Prüfungsergebnisse aus.

F4 Gute Programmierkompetenzen im Sinne des *computational thinking* wirken positiv auf den Lernprozesse im betrachteten Fach "Algorithmen und Datenstrukturen".

1 HTWK Leipzig, Fakultät Informatik und Medien, Gustav-Freytag-Str. 42A, 04277 Leipzig, karsten.weicker@htwk-leipzig.de

2 Peer-Reviews in der Lehre

Peer-Reviews wurden verschiedentlich bereits in Lehrveranstaltungen der Informatik eingesetzt. Ein starker Fokus liegt dabei auf der Programmierausbildung, wobei ganze Programme [RFT09, Li11] oder auch Concept-Maps für den Entwurf objektorientierter Programme [Tu10] begutachtet werden. Turner et. al. [Tu10] berichten, dass Peer-Reviews das Lernen von Konzepten in größeren Zusammenhängen verbessern. Reily et. al. [RFT09] zeigen, dass das Peer-Review von den Studierenden angenommen wird und dass die Tätigkeit des Begutachtens die Lernleistung der Gutachter signifikant verbessert. Auch Li et. al. [Li11] berichten von verbesserten Ergebnissen bei den Studierenden, die am Peer-Review teilnehmen.

Allgemein eignet sich das Peer-Review auch in der Informatik für schriftliche Arbeiten, beispielsweise in Oberseminaren [We07]. Gehringer [Ge01] hat dies in acht verschiedenen Modulen der Informatik eingesetzt, in welchen die Studierenden das Konzept als hilfreich eingeschätzt haben. Liu et.al. [Li01] ließen HTML-Dokumente in einem mehrstufigen Prozess begutachten und überarbeiten; sie berichten zwar von einer hohen Akzeptanz seitens der Studierenden, schließen allerdings aus ihren Daten, dass effektive Reviews nicht ausreichten, um eine gute Note in der Prüfung zu erhalten.

Machanick [Ma05] hat das Peer-Review für die Ergebnisse von vorlesungsbegleitenden Tests eingesetzt und konnte ebenfalls nur eine schwache Korrelation zur Abschlussprüfung identifizieren.

Während manche der vorstehenden Ansätze im Unterricht wie auch als Hausaufgabe eingesetzt werden können, seien hier noch Peer-Reviews erwähnt, die als Mittel der Interaktion in der Präsenzlehre dienen [NNP19] und daher keinen Bezug zu den Ergebnissen dieses Papers haben.

Unabhängig von der Informatik gibt es zahlreiche Studien und Veröffentlichungen. Topping [To98] berichtet, dass die Mehrheit der von ihm betrachteten Studien dem Peer-Review eine hohe Zuverlässigkeit attestiert. Der Literaturreview von Dochy et. al. [DSS99] fasst gar 63 Studien hinsichtlich Fairness, inkonsistente Gutachten und positive Effekte für den Lernprozess zusammen. Unter den jüngeren Veröffentlichungen haben Nicol et. al. [NTB14] den Einsatz der Technik in einem Ingenieursmodul mit Fragebögen begleitet und herausgefunden, dass das Peer-Review die Studierenden auf vielfache Weise aktiviert. Mulder et. al. [MPB14] haben noch tiefgreifender durch mehrere Fragebögen untersucht, wie sich die Einschätzungen der Studierenden durch das Peer Review verändern, wobei die Sinnhaftigkeit des Peer-Reviews in den vier betrachteten Fachgebieten teilweise radikal unterschiedlich beurteilt wurde.

3 Peer-Review statt Seminar

Die Rahmenbedingung in der Lehrveranstaltung “Algorithmen und Datenstrukturen” waren im Sommersemester 2020 wie folgt:

- 4 SWS Vorlesung wurden als 46 kleinteilige, thematisch abgeschlossene, vorproduzierte Screencasts aufbereitet,
- die Prüfungsvorleistung ist in der Prüfungsordnung vage als “Beleg und Präsentation” formuliert und konnte somit flexibel ausgelegt werden,
- es gibt 12 Übungsblätter mit jeweils 4–5Aufgaben (vor allem das händische Anwenden von Algorithmen, aber auch Laufzeitbetrachtungen, Analysen zu Sonderfällen und Transferaufgaben) und
- es gibt 3 Programmieraufgaben, von denen jede Studentin und jeder Student mindestens eine so lösen muss, dass die unbekannten Testfälle erfüllt werden.

Anstatt des bisherigen Seminarkonzepts (Votieren für mindestens 70% der Übungsaufgaben, mindestens einmaliges Vorrechnen an der Tafel) wurde nun über die Technik des Peer- Reviews die Auseinandersetzung mit unterschiedlichen Lösungen für die Aufgaben ebenfalls asynchron organisiert.

Die Studierenden mussten dabei für jedes Übungsblatt ihre Lösung entweder direkt am PC erstellen oder die handschriftliche Lösung per Scan/Foto digitalisieren und in Opal, dem Lernmanagementsystem des Bildungsportals Sachsen, hochladen. Nach der Abgabefrist müssen die Studierenden die Lösungen von drei Kommilitonen innerhalb 72 Stunden einsehen und kommentieren. Hierfür steht ein eher prototypischer Peer-Review-Baustein zur Verfügung, in dem immer nur eine Abgabe eines Kommilitonen zugeordnet wird – erst nach deren Begutachtung kann man eine weitere Abgabe einsehen.

Jede Aufgabe eines Übungsblatts, die ernsthaft bearbeitet wurde, ergab – ungeachtet der Korrektheit des Lösungsversuchs – einen Punkt für die Prüfungszulassung, wenn für das Übungsblatt drei Reviews für Abgaben von Mitstudierenden erstellt wurden. Es mussten insgesamt 34 Punkte über alle 12 Übungsblätter hinweg erreicht werden. Letztlich zählte zu diesen Punkten auch das Ergebnis einer Probeklausur, welche hier allerdings nicht berücksichtigt wird, da sie keinen Aufschluss über das betrachtete didaktische Konzept der Peer-Reviews gibt und eine hohe Korrelation zwischen Probeklausur und Prüfung das Ergebnis verfälscht.

Neben den studentischen Peer-Reviews gab es auch stichprobenartige Expertenreviews der Dozenten. Wegen der unterschiedlich hohen anderweitigen Lehrbelastung des involvierten Lehrpersonals hat die Hälfte der Studierenden Expertenreviews zu einem Großteil der Aufgaben erhalten, während die andere Hälfte nur die ursprünglichen geplanten stichprobenartigen Rückmeldungen erhielt.

Für Rückfragen und einen Austausch stand ein Forum und alle ca. 3 Wochen eine synchrone Fragestunde per Videokonferenz zur Verfügung. Die Prüfung fand als Klausur in Präsenz statt.

4 Studentische Einschätzung

In der studentischen Lehrevaluation wurden die Peer-Reviews neun Mal in den Kategorien "Was hat Ihren Lernerfolg negativ beeinflusst?" und "Was hätte in dieser Lehrveranstaltung besser gemacht werden können?" erwähnt. Demgegenüber stehen fünf Nennungen bei "Was hat Ihren Lernerfolg positiv beeinflusst?" und "Was hat Ihnen an dieser Lehrveranstaltung besonders gut gefallen?".

Häufig wird der hohe Arbeitsaufwand angeführt (4x). Andere Teilnehmer bezeichnen die Reviews als "nicht hilfreich, eher nervig", hinterfragen den Lerneffekt ("Was mir bis jetzt unverständlich geblieben ist, ist der Lerneffekt dabei", "hat mir beim Lernen nicht wirklich geholfen", "wozu die studentische Kontrolle?") und zeigten Unzufriedenheit mit den erhaltenen Reviews ("Die Reviews fand ich bis zum Schluss eher mittelmäßig, da die Bearbeitung meist eher schlecht ausfiel und man als Leistungsstärkerer selten selber etwas lernen konnte", "Reviews hatten wahrscheinlich nicht genau den Effekt, den sie haben hätten sollten. Viele schrieben nur 'habe alles genauso', obwohl eine Aufgabe komplett fehlte").

Den hohen Aufwand und den Nutzen wägten zwei Studierende in ihren Kommentaren ab ("mit den Reviews weiß ich nicht recht, gab zwar schon Fälle, wo ich es hilfreich fand, aber oft war es dann auch noch mehr Arbeit" vs. "Viel Arbeit, aber auch wirklich effektiv und wirkungsvoll").

Zwei positive Anmerkungen zum Peer-Review würdigten die Auseinandersetzung mit anderen Abgaben ("Die Peer-Reviews sind sehr hilfreich, da man sich noch einmal mit den Aufgaben beschäftigt und Lösungen von anderen sehen kann", "das Peer-Review hat bei mir manchen Aha-Effekt ausgelöst").

Bezüglich der Durchschnittswerte der Lehrevaluation bewegt sich die Lehrveranstaltung in demselben Bereich der beiden vorherigen Evaluationen (Tabelle 1). Es haben sich 42 von 106 Teilnehmern beteiligt. In der aktuellen Evaluation gab ferner die Mehrheit an, dass in dieser Lehrveranstaltung der Lernerfolg durch die Umstellung auf das digitale Lehrformat positiv beeinflusst wurde (20% "sehr positiv", 37,1% "eher positiv"). Nur 14,3% sahen die Umstellung "eher negativ", die restlichen 28,6% sahen keine Auswirkung auf den Lernerfolg.

Tab. 1: Lehrveranstaltung (2020) mit Peer-Reviews im Vergleich der studentischen Lehrevaluation

	2016	2019	2020
Insgesamt bewerte ich diese Vorlesung mit der Note	1,8	1,5	1,5
Diese Lehrveranstaltung fördert mein Interesse an dem Thema (1=stimme voll zu, 5=stimme gar nicht zu)	2,0	1,9	1,9
Ich habe in dieser Vorlesung ein tiefes Verständnis für den Stoff gewonnen (1=stimme voll zu, 5=stimme gar nicht zu)	2,0	1,8	1,9

5 Erhobene Daten

An der ersten Prüfung nach der Lehrveranstaltung haben insgesamt 91 der Studentinnen und Studenten teilgenommen, die direkt im Sommersemester 2020 die Prüfungszulassung erworben haben. Weitere 26 Prüflinge mit älterer Prüfungszulassung bleiben unberücksichtigt.

Für jeden Teilnehmer wurden dabei die folgenden Daten erhoben:

P: Punktzahl der Prüfungszulassung, die durch Übungsaufgaben und Reviews erreicht wurde [5 – 58, Median: 37]

ΔB: Spanne der bearbeiteten Übungsblätter $\#_{letztes} - \#_{erstes} + 1$ [1 – 12, Median: 10]

#oR: Anzahl der selbst bearbeiteten Blätter, für die keine Reviews anderer Abgaben erstellt wurde [0 – 5, Median: 1]

#R: Anzahl der erstellten Reviews [3 – 58, Median: 26.5]

QR: Durchschnittswert der Qualität der erstellten Reviews – diese wurden manuell im Nachhinein in drei Klassen eingeteilt (1 =oberflächlich, 2 =akzeptabel, 3 =spezifisch/detailliert) [1 – 2:424, Median: 1.418]

E: Teilnehmer war in der Gruppe mit Experten-Feedback [0/1]

Pr✓: Nummer der ersten erfolgreich bewältigten Programmieraufgabe [1 – 3, Median: 1]

#Pr: Anzahl der bearbeiteten Programmieraufgaben [1 – 3, Median: 1]

Z: Erhalt der Prüfungszulassung [0/1]

N: Note in der Prüfung, falls teilgenommen [1:0; 1:3; 1:7; : : : ; 4:0; 5:0, Median: 2.0]

Für ein besseres Verständnis der Datenbasis wird sie im Weiteren zunächst über Korrelationen, Clustering und eine Hauptkomponentenanalyse untersucht.

5.1 Korrelationen

Tabelle 2 zeigt die Korrelationswerte zwischen den verschiedenen Rohdaten. So besteht eine moderate Korrelation zwischen der Anzahl der bearbeiteten Programmieraufgaben und der ersten erfolgreichen Abgabe.

Tab. 2: Korrelationswerte zwischen den Attributen der erfassten Rohdaten.

	N	E	Z	#R	QR	P	ΔB	#oR	Pr✓	#Pr
N	1									
E	0,036	1								
Z	Div/0	0,141	1							
#R	-0,195	-0,085	**0,530**	1						
QR	**-0,419**	-0,166	0,141	0,246	1					
P	**-0,446**	-0,039	**0,620**	**0,874**	0,366	1				
ΔB	-0,086	0,143	**0,589**	**0,680**	0,134	**0,734**	1			
#oR	0,177	0,236	0,018	**-0,500**	-0,354	-0,397	0,076	1		
Pr✓	0,149	-0,001	Div/0	-0,029	-0,281	-0,165	0,043	0,052	1	
#Pr	0,003	0,140	0,115	0,220	-0,065	0,176	0,198	-0,029	**0,617**	1

Ein großes Korrelations-Cluster mit vornehmlich moderaten Korrelationen spiegelt die enge Verzahnung der Anzahl der Reviews #R, den spät bearbeiteten Übungsblättern ΔB, der erreichten Gesamtpunktzahl P und der Zulassung Z. Aus diesen Attributen weist allerdings einzig die Punktzahl eine schwache (negative) Korrelation zur Note in der Prüfung auf. Die einzige weitere nennenswerte, ebenfalls schwache negative Korrelation besteht zwischen der Note und der Qualität der Reviews – dies stützt schwach die These F1.

5.2 Daten-Cluster

Um stärker das Zusammenwirken aller erfassten Attribute zu berücksichtigen, haben wir mit dem Expectation-Maximization-(EM-) Algorithmus [RN20, S. 737ff] alle Studierenden in Cluster eingeteilt. Dies soll einen Einblick in typische Vorgehensweisen der Studierenden sowie die resultierenden Noten in der Prüfung geben. Tabelle 3 zeigt die sieben ermittelten Cluster mit ihren Zentroiden bzw. der Notenverteilung.

Die meisten Noten "sehr gut" sind in Cluster 4 und gehen einher mit vielen bearbeiteten Übungsblättern, Punkten und Reviews; die Reviews sind hochwertig und werden zuverlässig erbracht; auch die Programmierfertigkeit ist gut.

Cluster 2 und 3 repräsentieren vornehmlich Studierende mit der Note "gut". Beide Cluster haben ein extrem ähnliches Profil mit hohen Punktzahl – der einzige ausgesprochen markante Unterschied ist das Experten-Feedback. Dies ist ein starker Hinweis darauf, dass F2 falsch sein könnte.

Cluster 7 vereinigt Studierende, für welche die Programmieraufgabe eine große Herausforderung darstellte, die aber dennoch bei den Reviews Zuverlässigkeit bewiesen haben – zumeist mit der Prüfungsnote "befriedigend". Cluster 1 weist keine herausragenden Eigenschaften auf, sondern hat mehr den Charakter eines Gemischtwarenkorbs mit der vorherrschenden Note "ausreichend".

Tab. 3: Cluster der Rohdaten als Ergebnis des EM-Algorithmus. Auffallend große (und im unteren Teil auch kleine) Werte sind in jeder Zeile markiert.

	1	2	3	4	5	6	7
Note: sehr gut	1.186	1.803	2.620	**12.381**	1.185	**6.387**	2.437
gut	1.094	**15.219**	**14.923**	2.368	**4.113**	2.022	3.261
befriedigend	1.554	1.069	1.243	1.847	**5.999**	1.076	**8.214**
ausreichend	**8.267**	1.080	2.042	1.241	2.584	**3.287**	1.501
ungenügend	1.089	1.040	2.183	1.114	**4.028**	**3.013**	1.534
Anteil E	0.165	**0.919**	**0.097**	0.229	0.884	**0.906**	0.514
#R	27.159	28.401	26.393	**30.425**	24.436	**20.287**	27.672
QR	1.362	1.601	1.640	**1.858**	**1.222**	**1.250**	1.368
P	34.527	**43.917**	39.289	**47.308**	34.339	**32.576**	40.690
ΔB	9.743	10.277	**9.139**	**10.789**	9.532	10.055	9.810
#oR	1.622	0.674	0.649	**0.288**	1.058	**2.917**	**0.389**
Pr✓	2.165	1.334	1.549	**1.173**	**1.076**	**2.356**	**2.881**
#Pr	1.432	1.660	1.616	1.352	**1.000**	**2.284**	**2.723**

Spannend ist das Cluster 6, dessen Studierende in der Programmieraufgabe eine Herausforderung sahen, die viel Experten-Feedback erhalten haben, aber nur das Notwendigste bzgl. Punkte, Reviews und Reviewqualität geleistet haben. Die resultierenden Noten sind entweder "sehr gut" oder "ausreichend"/"ungenügend".

Studierende in Cluster 5 erledigten die erste Programmieraufgabe und lieferten eher oberflächlich Reviews; dafür erzielten sie Noten im Spektrum "gut" bis "ungenügend".

5.3 Hauptkomponentenanalyse

Schon bei der Betrachtung der Korrelationen hat sich angedeutet, dass verschiedene Attribute einen starken Zusammenhang aufweisen. Daher wird hier nochmals tiefgehender untersucht, welche kombinierten Vektoren die Punktwolke ohne Betrachtung der Prüfungsnote aufspannen. Tabelle 4 zeigt die Ergebnisse.

So können über die fünf wichtigsten Hauptkomponenten definierende Verhaltensweisen in der Menge der Studierenden mit abnehmender Bedeutung aufgezeigt werden:

Schmalspurverhalten (PC1):
wenig Punkte, wenig Reviews und viele fehlende Reviews

Mängel im computational thinking (PC2):
Erfolg bei später Programmieraufgabe, mehrere Programmieraufgaben

Wertschätzung von Experten-Feedback (PC3):
regelmäßiges Expertenfeedback, auch späte Blätter bearbeitet

Reviews ohne Tiefgang (PC4):
niedrige Reviewqualität, wenig fehlende Reviews, wenig Blätter

Suchende (PC5):
kein regelmäßiges Expertenfeedback, geringe Qualität, auch späte Blätter

Erstaunlicherweise ist die Reviewqualität im Gegensatz zur hohen Korrelation zur Note ein untergeordneter bestimmender Faktor in den restlichen Daten (erst in PC4 und PC5).

Tab. 4: Fünf wichtigste Hauptkomponenten in den Daten ohne Note und Zulassung.

	PC1	PC2	PC3	PC4	PC5
Experten-Feedback E	0.1511	0.1951	**0.6538**	0.2657	**-0.6247**
Anzahl Reviews #R	**-0.5265**	0.0882	0.0606	0.2506	0.2008
Qualitaet Reviews QR	-0.3089	-0.2742	-0.1825	**-0.6563**	**-0.5484**
Punkte P	**-0.5511**	0.0118	0.1525	0.0418	0.0029
Spanne Blätter ΔB	-0.2886	0.1972	**0.5668**	**-0.4287**	**0.3593**
Blätter ohne Review #oR	**0.4497**	0.0335	0.2763	**-0.4702**	0.2236
erfolgr. Prog. Pr✓	0.0711	**0.6487**	-0.2768	-0.1406	0.0917
Anzahl Prog. #Pr	-0.1013	**0.6465**	-0.1952	-0.0973	-0.2848
Wichtigkeit (sdv)	1.6822	1.3379	1.1274	0.8906	0.8105

6 Vorhersagekraft des Peer-Reviews

Die reine Datenanalyse liefert bereits einige Hinweise, dass F1 positiv beantwortet werden kann. Um insbesondere auch für F2, F3 und F4 komplexere Abhängigkeiten der Note von den verschiedenen Eingangsdaten zu analysieren, soll in diesem Abschnitt über zwei Machine-Learning-Modelle erfasst werden, wie genau eine wenigstens gute Note (≤ 2.3) prognostiziert werden kann und welche Abhängigkeiten die Modelle dafür benutzen.

Abbildung 1 zeigt einen J48graft pruned Entscheidungsbaum,welcher 82 der 91 Studierenden richtig einordnet. Die Prognosefähigkeit kann durch die 10-fache Kreuzvalidierung ermessen werden, welche mit 10 ähnlichen Modellen eine korrekte Klassifikation von 68.13% erreicht.

Das Modell kann durch Reviewqualität und die frühe Bearbeitung der Programmieraufgabe bereits 10 schlechte und 46 gute Noten richtig zuordnen. Im linken Teilbaum unterhalb der zweiten Ebene können über die Bedingungen "wenig Übungsblätter, aber

fehlende Reviews" sowie "viele Übungsblätter bei wenig Punkten" noch sechs der insgesamt 12 schlechten Noten im Teilbaum richtig klassifiziert werden. Der rechte Teilbaum unterhalb der zweiten Ebene ist schwerlich interpretierbar, lässt aber bezüglich des Experten-Feedbacks vermuten, dass gewissenhafte Studierende (Ebene 3) mit Schwierigkeiten im computational thinking (Ebene 2) von Experten-Feedback ggf. mehr profitieren können als von eigenen Reviews.

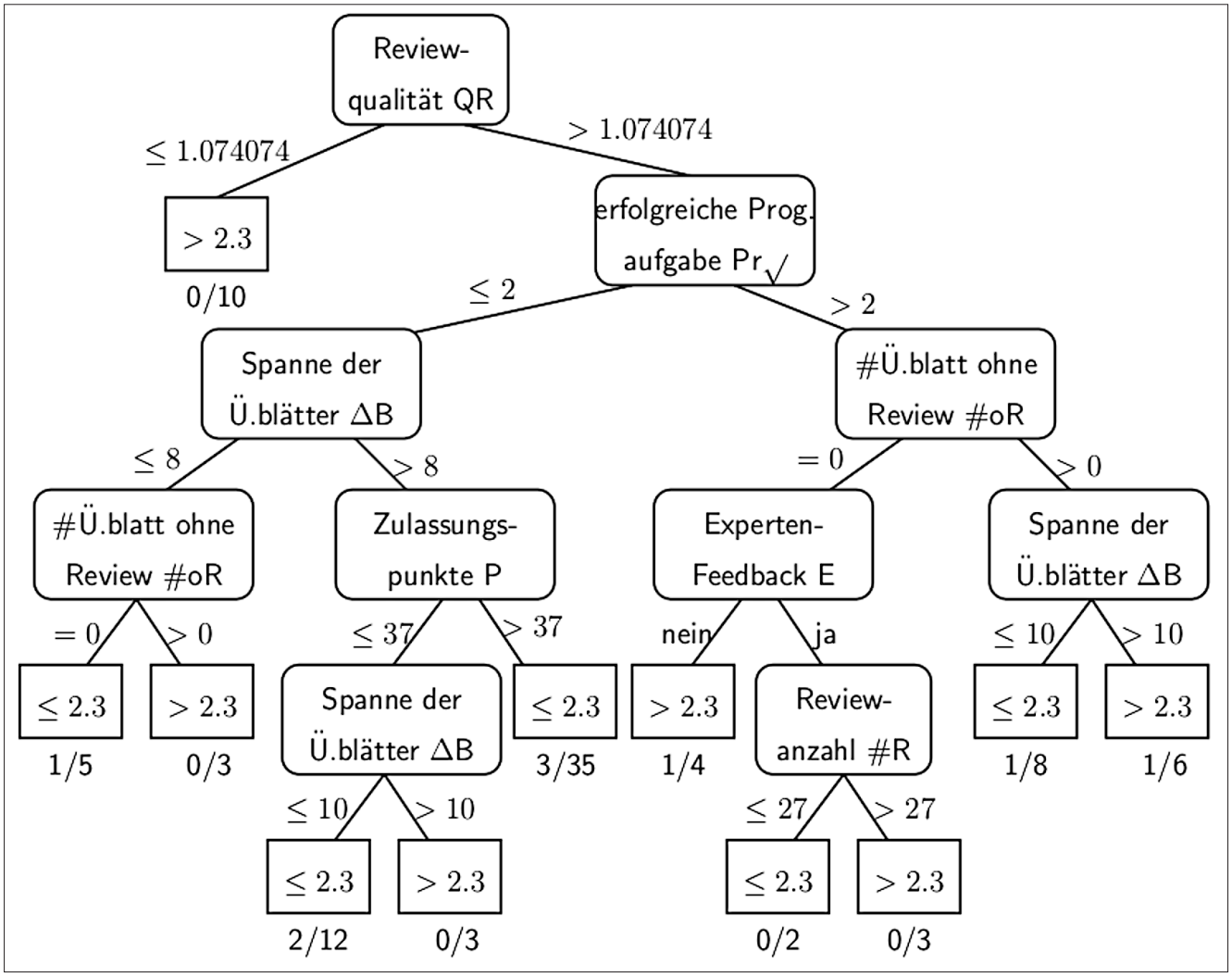

Abb. 1: Entscheidungsbaum (J48graft pruned) zur Prognose, ob wenigstens eine Note 2,3 erreicht wird. An jedem Blatt ist die #Fehlklassifikationen/#Gesamtklassifikationen annotiert.

Ein mit dem AD-Tree erstelltes Modell in Abb. 2 kann zwar lediglich 76 der 91 Klausurergebnisse richtig einordnen, hat aber in der Kreuzvalidierung mit 69.23% Trefferquote eine leicht bessere Vorhersagekraft. Bemerkenswert ist an diesem Modell, dass es ausschließlich die Qualität der Reviews, die Anzahl der Reviews, die erreichten Punkte bei der Zulassung und die erste erfolgreich bewältigte Programmieraufgabe benutzt. Dies unterstreicht den Einfluss, den eine gute Auseinandersetzung mit anderen Lösungen auf den Erfolg haben kann.

Das Ergebnis des Modells ist in tabellarischer Form in Tabelle 5 aufbereitet. Hier lässt sich gut ablesen, dass eine durchweg schlechte wie auch eine durchweg überragende Reviewqualität nahezu direkt die Note bestimmen. In den mittleren Bereichen ist dies nicht so leicht ablesbar, aber ein gutes computational thinking gibt verschiedentlich den Ausschlag zu guten Ergebnissen.

7 Ergebnisse

Die These **F1**, dass der Peer-Review in der Lehre ein Mechanismus für die tiefe Auseinandersetzung mit Inhalten sein kann, wird verschiedentlich bestärkt: die Qualität der Reviews (QR) ist die einzige nicht-triviale Korrelation zur Note, die Notenverteilung in den drei Clustern mit den qualitativ hochwertigsten Reviews weist überdurchschnittlich viele Noten "gut" und "sehr gut" auf und die Prognosemodelle zeigen, dass durchgängig schlechte Reviewqualität mit schlechten Noten einhergeht. Während die Hauptkomponentenanalyse zeigt, dass die Qualität der Reviews wenig mit anderen Merkmalen zusammenhängt, zeigt das Modell des ADTrees, dass QR kein alleiniger Indikator sein kann, da sie bei vielen Reviews verknüpft mit wenig Punkten eher negativ zu wirken scheint. Die These **F2**, dass Experten-Feedback (E) auf studentische Abgaben sich positiv auswirkt, bleibt ungeklärt. So zeigt zwar das Prognosemodell J48, dass es Situationen gibt, in denen Studierende davon profitieren können, der Faktor spielt jedoch keine Rolle im ADTree-Modell und den Korrelationen. Auch bei der Clusteranalyse belegen die Cluster 2 und 3 sowie Cluster 6 einen höchstens untergeordneten Einfluss von (E).

```
(1)reviewquality < 1.079: 1.315
(1)reviewquality >= 1.079: -0.175
|  (9)reviewquality < 1.225: -0.71
|  (9)reviewquality >= 1.225: 0.12
|  |  (10)reviewquality < 1.289: 0.728
|  |  (10)reviewquality >= 1.289: -0.129
(2)zulassungspunkte < 37.5: 0.426
(2)zulassungspunkte >= 37.5: -0.331
|  (5)reviewzahl < 27.5: -0.591
|  |  (7)reviewquality < 1.155: 0.703
|  |  (7)reviewquality >= 1.155: -0.836
|  (5)reviewzahl >= 27.5: 0.155
|  |  (6)zulassungspunkte < 43.5: 0.901
|  |  (6)zulassungspunkte >= 43.5:-0.363
(3)ersteerfolgreicheproga < 2.5: -0.26
(3)ersteerfolgreicheproga >= 2.5: 0.477
|  (8)reviewquality < 1.328: 0.539
|  (8)reviewquality >= 1.328: -0.26
(4)reviewquality < 1.877: 0.135
(4)reviewquality >= 1.877: -0.868
```

Abb. 2: ADtree für die Vorhersage der Note "≤ 2,3". Summen kleiner als Schwellwert -0.253 entsprechen einer besseren Note.

Der positive Einfluss von der Bearbeitung später Übungsaufgaben (ΔB, These **F3**) scheint schwach gegeben zu sein: so zeigen die Cluster mit späten Übungsaufgaben (2, 4, 6) einen hohen Anteil sehr guter und guter Noten, auch wenn das J48-Modell eine starke Abhängigkeit von anderen Faktoren nahe legt. Die Korrelationen können diesen Einfluss zwar nicht direkt belegen, aber ΔB ist mit P stark korreliert und P hat im ADTree-Modell einen hohen Einfluss bei ordentlicher Reviewqualität QR. Selbst in einem Fach wie "Algorithmen und Datenstrukturen" scheint der Einfluss von Vorkenntnissen im *computational thinking* (Pr✓, These **F4**) schwächer ausgeprägt zu sein, als üblicherweise angenommen: es liegt keine Korrelation zur Note vor und auch Cluster 5 zeigt, dass die Studierenden mit gutem computational thinking und oberflächlichen Reviews (QR) vornehmlich Noten im mittelmäßigen Bereich und den höchsten Anteil an Durchfallern aufweisen. Die unterschiedliche Anzahl der + im linken und rechten Block von Tabelle 5 zeigt hingegen deutlich, dass es einen Einfluss gibt, aber die Struktur des J48-Baums mit Pr✓ in der zweiten Ebene zeigt, dass der Faktor stark mit anderen Faktoren interagiert.

Tab. 5: Einordnung der Studierenden im ADTree, wobei + für die Noten sehr gut und gut steht und – für eine schlechtere Note. Bei den Zulassungspunkten unterscheiden wir die Bereiche hoch (↑), mittel (→) und niedrig (↓), bei der Anzahl der Reviews niedrig (⇓) und hoch (⇑).

	Prog.aufgabe	früh				spät			
	Punkte	↑	→	↑ / →	↓	↑	→	↑ / →	↓
	#Reviews	⇑	⇑	⇓	⇑/⇓	⇑	⇑	⇓	⇑/⇓
Reviewqualität	< 1.079	–	–	–	–	–	–	–	–
	1.079 – 1.155	+	+	+	+	+	–	–	–
	1.155 – 1.225	+	+	+	+	+	–	+	–
	1.225 – 1.289	–	–	+	–	–	–	–	–
	1.289 – 1.328	+	–	+	–	–	–	+	–
	1.328 – 1.877	+	–	+	–	+	–	+	–
	≥ 1.877	+	+	+	+	+	–	+	+

8 Fazit und Ausblick

Jegliche Analyse dieser Art, die Post-mortem durchgeführt wird, krankt daran, dass Kausalität bezüglich bestimmter Faktoren nicht abgeleitet werden kann – gehen die Fähigkeit zur tieferen Auseinandersetzung mit den Lösungen anderer grundsätzlich mit einer Tendenz zu besseren Noten einher oder hat die Ausübung des Reviews einen positiven Effekt auf die Note?

Bezüglich der möglichen Einflussfaktoren legt unsere Analyse den größten Einfluss eines tiefgründig ausgeführten Peer-Reviews nahe, wobei Experten-Feedback und Bearbeitung von Übungsaufgaben gegen Ende des Semesters untergeordnet zu sein scheinen.

Im Falle der Lehrveranstaltung "Algorithmen und Datenstrukturen" darf allerdings auch das *computational thinking* nicht vernachlässigt werden. Inwieweit die Technik des Peer-Reviews generell auch in anderen Lehrsituationen hilfreich sein kann, muss in weiteren Untersuchungen gezeigt werden.

Die Analyse wurde mit R [R 17] und WEKA [FHW16] durchgeführt.

Literaturverzeichnis

[DSS99] Dochy, Filip; Segers, Mien; Sluijsmans, Dominique: The use of self-, peer and coassessment in higher education: A review. Studies in Higher Education, 24(3):331–350, 1999.

[FHW16] Frank, Eibe; Hall, Mark A.; Witten, Ian H.: . The WEKA Workbench, 2016. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[Ge01] Gehringer, Edward F.: Electronic peer review and peer grading in computer-science courses. In (Walker, Henry MacKay; McCauley, Renée A.; Gersting, Judith L.; Russell, Ingrid, Hrsg.): Proc. of the 32rd SIGCSE Technical Symposium on Computer Science Education. ACM, S. 139–143, 2001.

[Li01] Liu, Eric Zhi-Feng; Lin, Sunny S. J.; Chiu, Chi-Huang; Yuan, Shyan-Ming: Web-based peer review: the learner as both adapter and reviewer. IEEE Trans. Educ., 44(3):246–251, 2001.

[Li11] Li, Cen; Dong, Zhijiang; Untch, Roland H.; Chasteen, Michael; Reale, Nathan: PeerSpace - An Online Collaborative Learning Environment for Computer Science Students. In: ICALT 2011, 11th IEEE International Conference on Advanced Learning Technologies, Athens, Georgia, USA, 6-8 July 2011. IEEE Computer Society, S. 409–411, 2011.

[Ma05] Machanick, Philip: Peer Assessment for Action Learning of Data Structures and Algorithms. In (Young, Alison; Tolhurst, Denise, Hrsg.): Seventh Australasian Computing Education Conf. (ACE 2005). Jgg. 42 in CRPIT. Australian Computer Society, S. 73–82, 2005.

[MPB14] Mulder, Raoul A.; Pearce, Jon M.; Baik, Chi: Peer review in higher education: Student perceptions before and after participation. Active Learning in Higher Education, 15(2):157–171, 2014.

[NNP19] Nazir, Sajid; Naicken, Stephen; Paterson, James H.: Teaching Data Structures through Group Based Collaborative Peer Interactions. In (Rahimi, Ebrahim; Stikkolorum, Dave R., Hrsg.): CSERC '19: The 8th Computer Science Education Research Conf. ACM, S. 98–103, 2019.

[NTB14] Nicol, David; Thomson, Avril; Breslin, Caroline: Rethinking feedback practices in higher education: a peer review perspective. Assessment & Evaluation in Higher Education, 39(1):102–122, 2014.

[R 17] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[RFT09] Reily, Ken; Finnerty, Pam Ludford; Terveen, Loren G.: Two peers are better than one: aggregating peer reviews for computing assignments is surprisingly accurate. In (Teasley, Stephanie D.; Havn, Erling C.; Prinz, Wolfgang; Lutters, Wayne G., Hrsg.): Proc. of the 2009 Int. ACM SIGGROUP Conf. on Supporting Group Work, GROUP 2009. ACM, S. 115–124, 2009.

[RN20] Russell, Stuart J.; Norvig, Peter: Artificial Intelligence: A Modern Approach. Pearson, Upper Saddle River, NJ, USA, 4. Auflage, 2020.

[To98] Topping, Keith: Peer Assessment Between Students in Colleges and Universities. Review of Educational Research, 68(3):249–276, 1998.

[Tu10] Turner, Scott A.; Pérez-Quiñones, Manuel A.; Edwards, Stephen H.; Chase, Joseph: Peer review in CS2: conceptual learning. In (Lewandowski, Gary;Wolfman, Steven A.; Cortina, Thomas J.; Walker, Ellen Lowenfeld, Hrsg.): Proc. of the 41st ACM technical symposium on Computer science education, SIGCSE 2010. ACM, S. 331–335, 2010.

[We07] Weicker, Nicole: Zielorientierte Didaktik der Informatik – Kompetenzvermittlung bei engen Zeitvorgaben. In (Schubert, Sigrid E., Hrsg.): Didaktik der Informatik in Theorie und Praxis. INFOS 2007. GI, Bonn, S. 337–348, 2007.

[We20] Weicker, Karsten: Teaching cooperative problem solving. In (Mottock, Jürgen, Hrsg.): ECSEE '20: European Conference on Software Engineering Education. ACM, S. 6–11, 2020.

[We21] Werner, Julia; Ebel, Christian; Spannagel, Christian; Bayer, Stephan, Hrsg. Flipped Classroom – Zeit für deinen Unterricht. Verlag Bertelsmann Stiftung, Gütersloh, 3. Auflage, 2021.

Fähigkeiten und Kenntnisse bei Studienanfänger*innen in der Informatik: Was erwarten die Dozent*innen?

Ergebnisse einer deutschlandweiten Umfrage unter Informatik-Hochschuldozent*innen

Esther Bender[1], Helena Barbas[2], Fabian Hamann[3], Marcus Soll[4], Daniel Sitzmann[5]

Abstract:
Viele Studieneingangs- und Eignungstests haben zum Ziel, für den entsprechenden Studiengang geeignete Studierende zu finden, die das Studium erfolgreich beenden. Gerade in der Informatik ist aber häufig unklar, welche Eigenschaften geeignete Studierende haben sollten - auch, weil der Erwartungshorizont der Dozierenden an Studienanfänger*innen nicht einheitlich ist und in der Vergangenheit auch nicht untersucht wurde. Um die Erwartungen von Dozent*innen an Studienanfänger*innen im Fach Informatik an deutschen Hochschulen zu analysieren, hat das Projekt MINTFIT im Sommer 2019 eine deutschlandweite Online-Befragung durchgeführt, an der 588 Hochschuldozent*innen aus allen Bundesländern teilnahmen. Die Umfrage hat gezeigt, dass überwiegend allgemeine Fähigkeiten, wie z.B. Motivation und logisches Denkvermögen und nur wenig fachliches Vorwissen, wie Programmieren oder Formale Sprache erwartet wird. Nach Einschätzung der Dozent*innen sind die problembehafteten Bereiche überwiegend in der theoretischen Informatik und in formellen Aspekten, (wie z.B: Formale Sprache) zu finden. Obwohl Tendenzen erkennbar sind, zeigt die Umfrage, dass bei Anwendung strenger Akzeptanzkriterien keine Fähigkeiten und Kenntnisse explizit vorausgesetzt werden, was darauf hindeutet, dass noch kein deutschlandweiter Konsens unter den Lehrenden vorhanden ist.

Keywords:
Informatikstudium, Studienanfänger*innen, Vorkenntnisse, Studieneingangsphase, Umfrage

1 Einführung

Viele Studierende beenden ihr Informatikstudium nicht mit einem Abschluss, die Studienabbruchquote variiert dabei je nach Umfrage zwischen 30% und 45% [He10]. Dabei werden als Abbruchgrund unter anderem hohe Leistungsanforderungen genannt, dies ist laut Heublein et al. [He10] mit 25% der häufigste Abbruchgrund. Zusätzlich werden nach Heublein et al. [He10] Studierende oftmals von den hohen Leistungsanforderungen am Anfang des Studiums überrascht. Dies wirft die Frage auf, ob die Erwartungen der Dozent*innen in der Informatik an die fachlichen Vorkenntnisse und Fähigkeiten dem Niveau der Studienanfänger*innen entsprechen. In anderen Fächern existieren Erhebungen dazu, welche Erwartungen Dozent*innen an Studienanfänger*innen haben, wie zum Beispiel von Neumann et al. [NPH17] im Fach Mathematik. Für den Bereich der Informatik gibt es bislang nur wenig Forschung zur Erwartung der Dozent*innen an Studienanfänger*innen und auch keinen bundesweit einheitlichen Mindestanforderungskatalog, wie es z.B. im Bereich Mathematik der Mindestanforderungskatalog der Cooperation Schule-Hochschule (kurz „cosh") [Co14] ist. Diese zwischen Hochschulen, Berufsschulen und Schulen in Baden-Württemberg ausgehandelten und mittlerweile an vielen Hochschulen Deutschlands akzeptierten Mindestanforderungen sind eine Übereinkunft darüber, auf welches in der Schule erworbene Vorwissen Hochschuldozent*innen aufbauen können. Ein vergleichbarer Mindestanforderungskatalog fehlt im Bereich der Informatik bislang. Zwar gibt es Empfehlungen für den Informatik-Schulunterricht der Gesellschaft für Infor-

1 Hochschule für Angewandte Wissenschaften Hamburg, Fakultät Technik und Informatik, Berliner Tor 7, 20099 Hamburg, Deutschland esther.bender1@haw-hamburg.de, ORCID: https://orcid.org/0000-0002-4912-3955

2 HafenCity Universität Hamburg, Geomathematik, -informatik & Physik, Henning-Voscherau-Platz 1, 20457 Hamburg, Deutschland helena.barbas@hcu-hamburg.de, ORCID: https://orcid.org/0000-0002-2384-8042

3 Technische Universität Hamburg, Institut für Mathematik, Am Schwarzenberg-Campus 3, 21073 Hamburg, Deutschland fabian.hamann@tuhh.de, ORCID: https://orcid.org/0000-0001-7166-1437

4 Marcus Soll, Autal 20, 22880 Wedel, Deutschland research@msoll.eu, ORCID: https://orcid.org/0000-0002-6845-9825

5 Technische Universität Hamburg, Arbeitsstelle MINTFIT Hamburg (AMH), Technische Universität Hamburg, Schloßmühlendamm 30, 21073 Hamburg, Deutschland daniel.sitzmann@uhh.de, ORCID: https://orcid.org/0000-0003-2295-3083

matik e.V. ([Pu08], [Rö16]), diese sind jedoch nicht in allen Bundesländern umgesetzt. Es ist daher sowohl unklar, welches Wissen Schüler*innen aus der Schule mitbringen, als auch was Lehrende von Studienanfänger*innen erwarten. Um diese Forschungslücke zu schließen, wurde im Rahmen des Projekts MINTFIT [MSZ18] eine bundesweite Onlineumfrage unter Lehrenden an Hochschulen zu dem Thema durchgeführt, welche Erwartungen Hochschuldozent*innen in der Informatik an Studienanfänger*innen stellen. Studien aus dem englischsprachigen Raum legen nahe, dass Studierende der Informatik häufig an Kompetenzen wie *Problemlösekompetenz* [BLH01] oder *Auge für Details* [CPV05] scheitern. Diese dort genannten Kompetenzen wurden in die Umfrage integriert. Die Ergebnisse dieser Studie, an der 588 Personen teilnahmen, werden im Folgenden vorgestellt. Mithilfe der gewonnenen Umfrageergebnisse konnte nicht nur ein Informatik-Eingangstest im Projekt MINTFIT fundiert entwickelt werden [Si19], sondern auch ein Beitrag für die Informatik-Didaktik allgemein geleistet werden.

2 Durchführung der Umfrage

Zur Ermittlung der Erwartungen (Kenntnisse und allgemeine Fähigkeiten) von Hochschul-Informatiklehrenden an Erstsemester-Studierende wurde eine Online-Umfrage durchgeführt. Der Link zu der Umfrage wurde an ca. 6.000 Lehrende an Universitäten und anderen Hochschulen in ganz Deutschland per E-Mail verschickt. Es wurden alle ermittelbaren Personen, die im Bereich Informatik einer öffentlichen Bildungseinrichtung laut institutseigener Internetseite beschäftigt waren und deren E-Mail-Adressen frei zugänglich waren, angeschrieben. Der Umfragezeitraum lief über ca. 5 Wochen im Juli 2019. Die Umfrage bestand aus 14 Fragen: vier Fragen mit Likert-Skalen, vier offene Textfragen und sechs Single-Choice-Fragen. Die Bearbeitungsdauer zur Beantwortung aller Fragen lag durchschnittlich bei 19 Minuten. Als Vorbild für die Umfrage diente die Studie von Neumann et al. [NPH17]. Eine dreistufige Delphi-Studie durchzuführen, war im Rahmen des MINTFIT-Projektes leider nicht möglich. Die offene erste Frage-Runde wurde durch die Expertise verschiedener Hamburger Informatik-Dozent*innen und der Projekt-Mitarbeiter*innen ersetzt, und im Gegensatz zu der Studie aus 2017 wurde nur eine Umfrage-Runde durchgeführt. Zur statistischen Auswertung wurden folgende vier Merkmale der Teilnehmer*innen und der Institution, an der sie beschäftigt sind, abgefragt:

- Art der Hochschule (Universität, Technische Universität, Hochschule für Angewandte Wissenschaften/Fachhochschule)
- Berufliche Position (Professor*in, Wissenschaftliche*r Mitarbeiter*in, Lehrbeauftragte*r)
- Lehrtätigkeit im 1. Semester (Ja/Nein)
- Bundesland, in dem die arbeitgebende Hochschule ansässig ist

Um die Erwartungen an Studienanfänger*innen genauer zu bestimmen, wurden sowohl Kompetenzen als auch loser definierte Fähigkeiten und Wissenstände abgefragt. Diese werden im Folgenden gesammelt als Fähigkeiten und Kenntnisse bezeichnet. Die unterschiedlichen Items wurden in folgende Kategorien eingeteilt: allgemeine Fähigkeiten und Kenntnisse, fachliche Fähigkeiten und Kenntnisse und problembehaftete Bereiche. „Problembehaftete Bereiche" bedeutet in diesem Fall, dass nach Ansicht der Lehrenden Studienanfänger*innen mit diesen Themengebieten in den ersten Semestern Probleme haben (es ihnen also schwer fällt, diese Fähigkeiten zu erwerben) - wie sich diese auf den Studienerfolg auswirken, wurde nicht erfragt. In der Umfrage sollten Teilnehmer*innen der Umfrage zuerst per Likert-Skala vorgegebene Fähigkeiten und Kenntnisse danach bewerten, ob und in welchem Grad sie diese erwarten. Anschließend hatten sie die Möglichkeit in Freitextfeldern Fähigkeiten und Kenntnisse zu ergänzen, die in den aufgeführten Antwort-Items nicht berücksichtigt worden waren. Die vorgegebenen Items für die Frage nach erwarteten fachlichen Fähigkeiten und Kenntnissen und problembehafteten Bereichen waren identisch. Bei der Frage nach den allgemeinen Fähigkeiten und Kenntnissen wurden andere Items vorgegeben. Es wurde eine Likert-Skala verwendet, um eine genauere Gewichtung der einzelnen Items zu ermöglichen. Die abgefragten allgemeinen Fähigkeiten und Kenntnisse wurden aus persönlichen Gesprächen mit Dozent*innen sowie verschiedener Fachliteratur ([BLH01], [CPV05]) abgeleitet. Bei der Formulierung der Items wurde darauf geachtet, dass diese möglichst eindeutig zu verstehen sind. Folgende allgemeine Fähigkeiten und Kenntnisse wurden in der Umfrage abgefragt:

Abstraktionsvermögen, Algorithmisches Denkvermögen, Analytische Fähigkeiten, Auge für Details, Durchhaltevermögen/Frustrationstoleranz, Fähigkeit zum schnellen Erfassen von mathematischen Sachverhalten, Interesse an Informatik, Kommunikationsfähigkeit, Konzentriertes und sorgfältiges Arbeiten, Leseverständnis von allgemeinen deutschen Texten, Logisches Denkvermögen, Mathematikkenntnisse auf Abiturniveau (Grundkurs), Motivation, Problemlösekompetenz, Strukturiertes Denken und Handeln, Visualisierung von Ergebnissen (PowerPoint etc.)

Die abgefragten fachlichen Fähigkeiten und Kenntnisse ergaben sich ebenfalls aus persönlichen Gesprächen sowie vorhandener Literatur: Dazu wurden insbesondere die Lehrpläne der 16 Bundesländer mit den Curricula von vier Hamburger Hochschulen (HafenCity Universität Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Technische Universität Hamburg, Universität Hamburg) verglichen [Si19]. Auch hier wurde auf eine eindeutige Formulierung geachtet. Folgende fachliche Fähigkeiten und Kenntnisse wurden in der Umfrage abgefragt: *Automatentheorie, Binärsystem, Datenbanken, Datenanalysesprachen (z.B. R, Matlab), Dokumentenbeschreibungssprachen (z.B. LaTeX, HTML, Markdown), Programmiersprachen (z.B. Java, Python, .net), Formale Logik/logische Operatoren, Formale Sprachen (Sprachentheorie), Kenntnisse über den Zusammenhang zwischen Informatik und Gesellschaft, Kenntnisse über elementare Algorithmen, Mengenlehre, Modellierung von Problemen, Rechneraufbau/Hardware, Rechnernetzwerke, Umgang mit handelsüblicher Software, Umgang mit (mehreren) Betriebssystemen, Umgang mit Logarithmen, Lesen von formalisierter Schreibweise, Schreiben von formalisierter Schreibweise.*

Die Reihenfolge der Items im Fragebogen war prinzipiell zufällig, nur zusammengehörige Gruppen (wie z.B. das Lesen/Schreiben formalisierter Schreibweise) wurden als solche betrachtet und stets hintereinander abgefragt. Nach dem Hauptteil wurde als letzte Frage auch noch die Zustimmung zu dem Satz „Unterricht im Fach Informatik in der Schule ist notwendig für ein erfolgreiches Informatikstudium." abgefragt, um die These, dass ein Informatikstudium auch ohne fachspezifische Vorkenntnisse möglich ist, zu validieren. Die weiteren drei Fragen bezogen sich auf die geplante Entwicklung eines Informatik-Eingangstests im Projekt MINTFIT [Si19]. Auf diese wird im Folgenden nicht näher eingegangen.

3 Ergebnisse

Insgesamt nahmen 588 Personen an der Umfrage teil; alle angefangenen Fragebögen wurden bis zum Ende bearbeitet und abgegeben. Bei rund 6000 angeschriebenen Personen entspricht dies einer Rücklaufquote von ca. 10%, was für eine Umfrage dieser Art und mit der gewählten Verteilungsstrategie „per E-Mail" als verhältnismäßig solide eingeschätzt werden kann. Unter den Teilnehmenden waren nach eigenen Angaben 222 Professor*innen, 328 wissenschaftliche Mitarbeiter*innen sowie 38 Personen, die ihren Status als „Sonstige" angaben. Da aufgrund von Datenschutzbestimmungen keine Informationen über die Angeschriebenen systematisch gespeichert werden durften, ist nicht festzustellen, ob dies verhältnismäßig der angeschriebenen Personenverteilung entspricht. Die Tatsache, dass generell mehr wissenschaftliche Mitarbeiter*innen an Hochschulen beschäftigt sind als Professor*innen, kann diese Verteilung aber erklären. Die meisten Teilnehmenden lehren oder lehrten an Universitäten (282) oder Fachhochschulen (221), aber auch Lehrende an Technischen Universitäten (84) nahmen an der Umfrage teil. Personen aus allen Bundesländern nahmen an der Umfrage teil, wobei jeweils der Ort der Lehrtätigkeit (also Sitz der Hochschule) und nicht der Wohnort erfragt wurde. Den zahlenmäßig größten Rücklauf gab es aus Bayern (108) und Nordrhein-Westfalen (83), die sowohl die bevölkerungsreichsten Bundesländer, als auch die Länder mit den meisten Hochschulen sind. Aus Hamburg gab es verhältnismäßig viele Rückmeldungen (26), was hauptsächlich auf persönliche Kontakte der durchführenden Projektgruppe zurückzuführen ist. Aus den östlichen Bundesländern gab es wenige Rückmeldungen (Brandenburg (11), Mecklenburg-Vorpommern (12)). Die Mehrheit der Teilnehmenden war aktiv in die Lehre des 1. Semesters eingebunden, davon 173 in mehreren Kursen, 176 in nur einem Kurs. 239 der Teilnehmenden gaben keine Lehre im 1. Semester.

Die Auswertung erfolgte in mehreren Abschnitten nach folgendem Schema: Zuerst wurden die Antworten zu den vorgestellten Fähigkeiten und Kenntnissen ausgewertet, danach (falls vorhanden) die Freitextfelder in aggregierter Form dargestellt und abschließend – entsprechend der Studie von Neumann et al. [NPH17] – festgehalten, welche Fähigkeiten und Kenntnisse die Dozent*innen der Informatik voraussetzen. Als Akzeptanz-Kriterien gelten hier (angelehnt an [NPH17]):

- Ein Item wird als vorausgesetzt angesehen, wenn 2/3 aller Befragten und 1/2 der Befragten pro Hochschulart (Universität, Technische Universität, Fachhochschule bzw. Hochschule für Angewandte Wissenschaften etc.) diese als vorausgesetzt ansehen.

- Ein Item wird als explizit nicht vorausgesetzt angesehen, wenn 3/4 aller Befragten und 2/3 der Befragten pro Hochschulart (Universität, Technische Universität, Fachhochschule bzw. Hochschule für Angewandte Wissenschaften etc.) diese als nicht vorausgesetzt ansehen.

3.1 Allgemeine Fähigkeiten und Kenntnisse

Die Ergebnisse der Erhebung bezüglich der allgemeinen Fähigkeiten und Kenntnisse sind in Abbildung 1 dargestellt. Fast alle Fähigkeiten und Kenntnisse werden von mindestens der Hälfte aller befragten Dozent*innen vorausgesetzt. *Interesse an Informatik, logisches Denkvermögen* und *Motivation* erhielten die größte Zustimmung. Lediglich *Auge für Details und Visualisierung von Ergebnissen* werden von nur ca. 40% der Teilnehmenden vorausgesetzt.

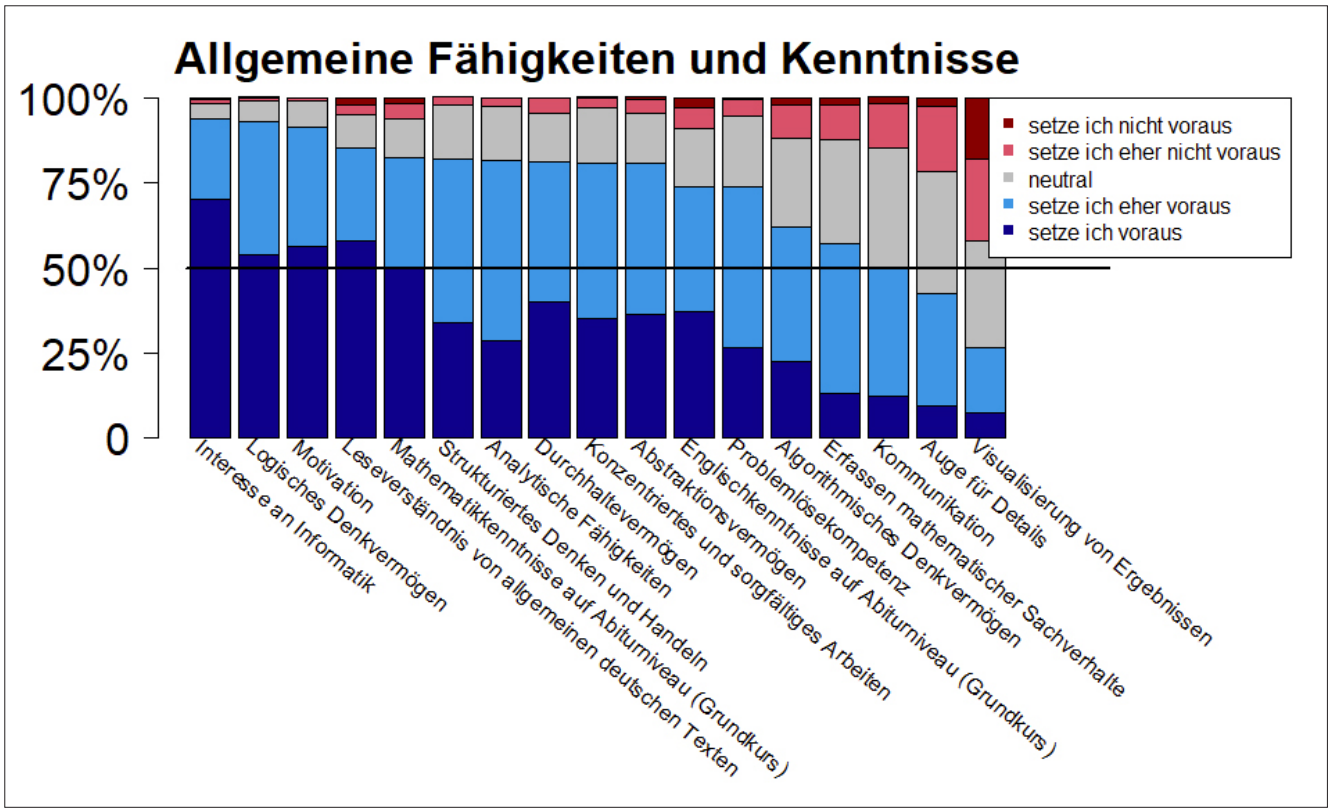

Abb. 1: Von Dozent*innen vorausgesetzte allgemeine Fähigkeiten und Kenntnisse, sortiert nach der Zustimmung (dabei wurden die Antworten „setze ich voraus“ und „setze ich eher voraus“ als Zustimmung gewertet).

In der dazugehörigen Freitextfrage wurden folgende weitere allgemeine Fähigkeiten und Kenntnisse genannt (fett gesetzte Items sind neu): **Selbstständigkeit** (34-mal genannt), **Teamfähigkeit** (25), **Neugierde**/*Motivation* (23), **Sozialkompetenz** (16), **Medienkompetenz** (11), **Disziplin** (10), *Problemlösekompetenz* (9), *Mathematik* (8), **Kritisches Denken** (6), **Ehrgeiz** (6), **Recherchekompetenz** (6), *Englischkenntnisse* (5), **Interdisziplinarität** (5).

Die Auswertung anhand der Akzeptanz-Kriterien ergibt, dass folgende allgemeine Fähigkeiten und Kenntnisse von den Dozent*innen der Informatik vorausgesetzt werden: *Motivation, Frustrationstoleranz/Durchhaltevermögen, Interesse an Informatik, Mathematik auf Abiturniveau (Grundkurs), konzentriertes und sorgfältiges Arbeiten, logisches Denkvermögen, Englischkenntnisse auf Abiturniveau (Grundkurs), Problemlösekompetenz, Leseverständnis von allgemeinen deutschen Texten, analytische Fähigkeiten, strukturiertes Denken und Handeln.* Keines der abgefragten Items wird explizit nicht vorausgesetzt.

3.2 Fachliche Fähigkeiten und Kenntnisse

Abbildung 2 zeigt die Ergebnisse für die fachlichen Fähigkeiten und Kenntnisse. Die einzigen zwei fachlichen Fähigkeiten und Kenntnisse, die von mehr als der Hälfte der Dozent*innen vorausgesetzt werden, sind Umgang mit *handelsüblicher Software* und *Mengenlehre.* Außer diesen beiden sowie *Umgang mit Logarithmen*, *Binärsystem* und Lesen von *formalisierter Schreibweise* werden alle fachlichen Fähigkeiten und Kenntnisse von der Mehrheit der Dozent*innen der Informatik nicht vorausgesetzt.

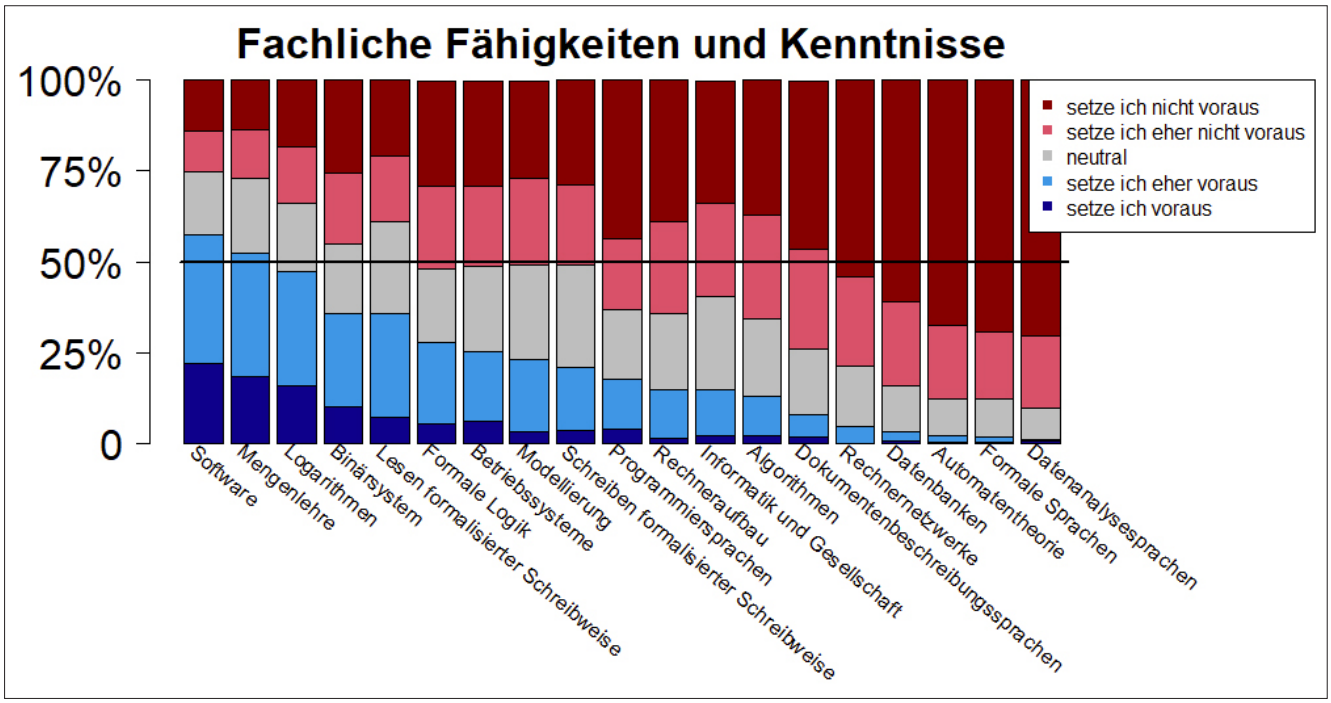

Abb. 2: Von Dozent*innen vorausgesetztes fachliches Vorwissen, sortiert nach der Zustimmung.

In der dazugehörigen Freitextfrage wurden die folgenden weiteren Fähigkeiten und Kenntnisse genannt (einige davon wurden schon unter „allgemeine Fähigkeiten und Kenntnisse" abgefragt, fett gesetzte Items sind neu)): *Mathematik* (19-mal genannt), *Computeranwendungen* (16), **Kommunikation** (7), **Deutsch** (6), *Programmierung* (5), **Physik/Elektrotechnik** (3), **Grundlagen Informatik** (3).

Das Akzeptanz-Kriterium ergibt, dass keine fachlichen Fähigkeiten und Kenntnisse explizit vorausgesetzt werden. Nach dem Akzeptanz-Kriterium werden folgende fachliche Fähigkeiten und Kenntnisse explizit nicht vorausgesetzt: *Rechnernetzwerke, Datenbanken, Automatentheorie, Formale Sprachen (Sprachtheorie), Datenanalysesprachen.*

3.3 Problembehaftete Bereiche

Die Ergebnisse zu der Frage, welche Bereiche die Informatik-Dozent*innen als problematisch für die Studierenden ansehen, sind in Abbildung 3 dargestellt. Dabei werden die Bereiche *Schreiben von formalisierter Schreibweise, Modellierung von Problemen, Lesen von formalisierter Schreibweise, Formale Sprachen (Sprachtheorie), Formale Logik/logische Operatoren, Automatentheorie, Programmiersprachen, Umgang mit Logarithmen* von mehr als der Hälfte der Informatik-Dozent*innen als problematisch angesehen. Im Gegensatz dazu werden die folgenden Bereiche von den Dozent*innen in der Informatik mehrheitlich als nicht problematisch bewertet: *Dokumentenbeschreibungssprachen, Binärsystem, Kenntnisse Zusammenhang Informatik und Gesellschaft, Umgang mit handelsüblicher Software.*

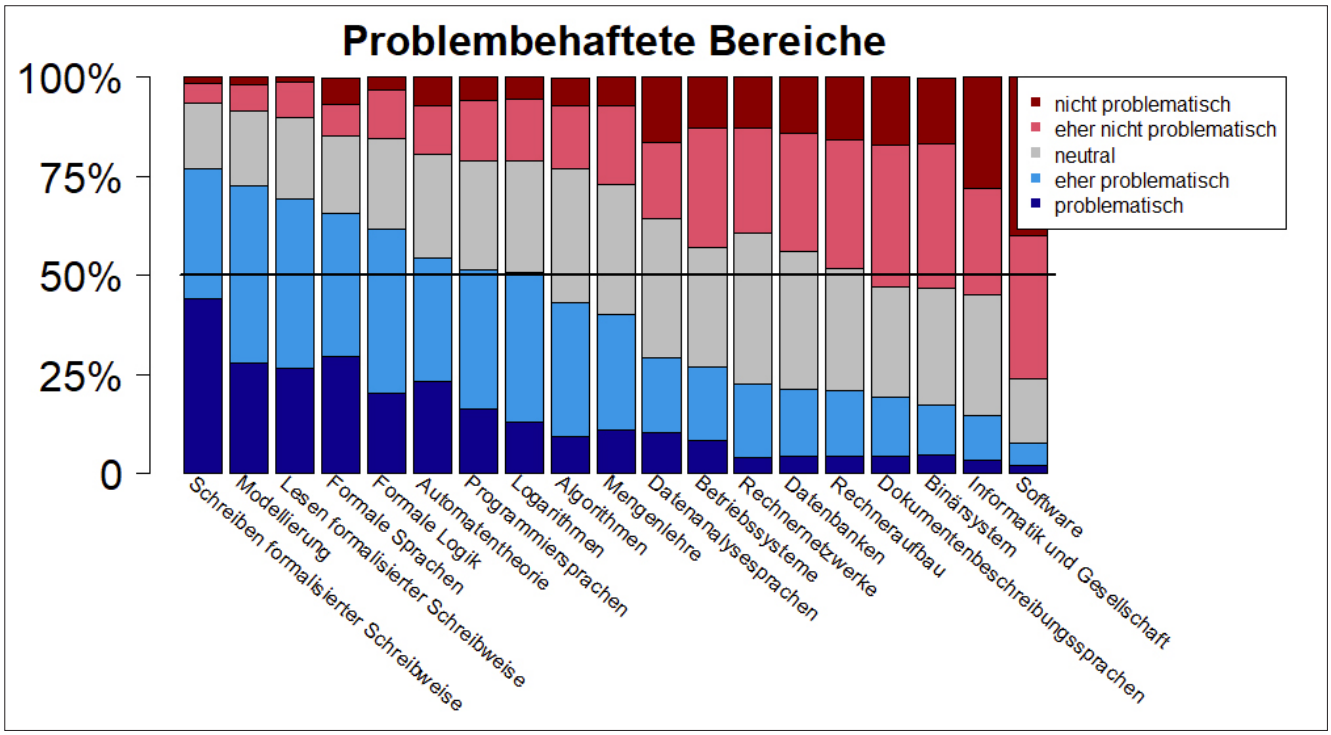

Abb. 3: Nach Ansicht der Informatik-Dozent*innen für Studierende problembehaftete Bereiche, sortiert nach der Zustimmung.

In der dazugehörigen Freitextfrage wurden folgende weitere problembehaftete Bereiche genannt (fett gesetzte Items sind neu): *Programmieren* (7-mal genannt), **Kommunikationsfähigkeit** (5), **Selbstständigkeit** (4), *Modellierung* (3), **Universitätsalltag** (3) und *Logik* (2).

Für die problembehafteten Bereiche wurde keine Überprüfung anhand des Akzeptanzkriteriums vorgenommen, da eine solche Auswertung bei der Frage nach problembehafteten Bereichen nicht sinnvoll erscheint.

3.4 Informatik in der Schule

Der Aussage „Unterricht im Fach Informatik in der Schule ist notwendig für ein erfolgreiches Informatikstudium" stimmten über die Hälfte der Dozent*innen nicht zu. 29% stimmen der Aussage dabei überhaupt nicht zu, 27% stimmen der Aussage eher nicht zu. Ungefähr ein Viertel (19%) vertritt die Ansicht, dass Informatik in der Schule eher notwendig für ein erfolgreiches Studium ist und 8% sind der Meinung, dass Informatik in der Schule notwendig für ein erfolgreiches Studium ist. 17% verhalten sich in dieser Frage neutral.

3.5 Sonstige Ergebnisse

Es konnte bei den Erwartungen der Dozent*innen kein Unterschied zwischen den verschiedenen Bundesländern, Statusgruppen bzw. Hochschularten festgestellt werden. Einige Freitext-Antworten brachten die Frustration der Dozent*innen bezüglich der Fähigkeiten und Kenntnisse der Studienanfänger*innen der letzten Jahre deutlich zum Ausdruck. Insbesondere wird dort ein Mangel an Grundkenntnissen (wie z.B. Lesekompetenz, Deutschkenntnisse, Mathematik auf Abitur-Niveau) hervorgehoben. Die Auswertung der Umfrage hat gezeigt, dass die zusätzliche Differenzierung durch die Likert-Skala keinen großen Mehrwert gegenüber einer einfachen Ja/Nein-Skala gebracht hat.

3.6 Faktorenanalyse

Zur genaueren Auswertung der Umfrage und um zu untersuchen, ob sich die erwarteten fachlichen und allgemeinen Fähigkeiten und Kenntnisse bestimmten Gruppen zuordnen lassen, wurde für die Fragen nach den allgemeinen erwarteten Fähigkeiten und Kenntnissen (F1), fachlichen Fähigkeiten und Kenntnissen (F2) und den problembehafteten Bereichen (F3) eine explorative Fakto-

renanalyse [WB10] durchgeführt. Alle statistischen Analysen wurden mit der R-Software[6] durchgeführt. Mithilfe des Kaiser-Meyer-Olkin-Kriteriums wurde überprüft, ob der Datensatz dafür grundsätzlich geeignet ist. Die Werte lagen für die Fragen zwischen 0,84 und 0,91, die der einzelnen Faktoren lagen alle über 0,6, womit die Datensätze grundsätzlich für eine Faktorenanalyse geeignet sind. Für alle Fragen wurde eine Faktorenanalyse nach dem Maximum-Likelihood-Kriterium mit einer Rotation durchgeführt. Die Anzahl der optimalen Faktoren wurde durch Ausprobieren bestimmt. Für die Frage nach den allgemeinen erwarteten Fähigkeiten und Kenntnissen (F1) wurden zwei Faktoren ausfindig gemacht. Der erste Faktor enthält mathematische Fähigkeiten und Kenntnisse wie *Fähigkeit zum schnellen Erfassen von mathematischen Sachverhalten, Abstraktionsvermögen, logisches Denkvermögen, Problemlösekompetenz, algorithmisches Denkvermögen, analytische Fähigkeiten, strukturiertes Denken und Handeln.* Der zweite Faktor enthält die folgenden motivationalen Aspekte und kommunikativen Fähigkeiten: *Motivation, Durchhaltevermögen, Interesse an Informatik, konzentriertes und sorgfältiges Arbeiten, Leseverständnis von allgemeinen deutschen Texten, Auge für Details, Kommunikation.* Die drei Items *Mathematik auf Abiturniveau, Visualisierung von Ergebnissen*, *Englischkenntnisse* mussten nachträglich aus der Analyse entfernt werden. Die 17 abgefragten Items lassen sich also in zwei Faktoren unterteilen - einerseits mathematische Fähigkeiten, andererseits motivationale Aspekte und kommunikative Fähigkeiten. Vergleicht man die Faktorensummen der beiden Faktoren, so zeigt sich, dass die sozialen und kommunikativen Fähigkeiten eher von den Dozent*innen erwartet werden als die mathematischen. Die motivationalen Aspekte und kommunikativen Fähigkeiten werden zu 40% vorausgesetzt und zu 34% eher vorausgesetzt, die mathematischen zu 30% vorausgesetzt und zu 45% eher vorausgesetzt. Für die Frage nach den erwarteten fachlichen Fähigkeiten und Kenntnissen wurden vier Faktoren identifiziert: Mathematische Kenntnisse (Faktor Aufbauwissen Mathematik), Fähigkeiten und Kenntnisse, die sich mit Aufbau und Benutzung des Computers beschäftigen (Faktor Computer), Fähigkeiten, die das Gebiet der theoretischen Informatik betreffen (Faktor Informatik) und formalisierte Schreibweise. Die einzelnen Items sind dabei wie folgt auf die Faktoren verteilt:

- Faktor Aufbauwissen Mathematik: *Binärsystem, Formale Logik, Modellierung, Logarithmen*
- Faktor Computer: *Rechneraufbau, Software, Betriebssysteme, Dokumentenbeschreibungssprachen, Programmiersprachen*
- Faktor Informatik: *Rechnernetzwerke, Datenbanken, Automatentheorie, Formale Sprache, Datenanalysesprache, Algorithmen*
- Faktor Formalisierte Schreibweise: *Lesen formalisierter Schreibweise, Schreiben formalisierter Schreibweise*

Der Vergleich der Faktorensummen zeigt, dass die Fähigkeiten und Kenntnisse im Bereich Aufbauwissen Mathematik am ehesten erwartet werden – 10% setzen diese Faktoren voraus, 25% setzen sie eher voraus. Die Items im Bereich Informatik werden am wenigsten erwartet, diese werden nur zu 0,6% vorausgesetzt und nur zu 3% eher vorausgesetzt. Bei der Faktorenanalyse für die problembehafteten Bereiche stellte sich heraus, dass sich keine Faktoren eindeutig identifizieren lassen. Die Faktorenanalyse zeigt einerseits, dass sich die unterschiedlichen Items in verschiedene Gruppen einteilen lassen. Bei den allgemeinen Fähigkeiten in motivationale Aspekte (die eher allgemeine Fähigkeiten beinhalten) und mathematische Kenntnisse, und für das fachliche Vorwissen in die oben genannten Faktoren. Diese Gruppen werden unterschiedlich stark von den Dozent*innen vorausgesetzt, wobei die allgemeineren Fähigkeiten jeweils häufiger vorausgesetzt werden. Die Faktorenanalyse verifiziert also die Ergebnisse aus Abschnitt 3.1.-3.3., dass überwiegende allgemeine Fähigkeiten und wenig fachliche Fähigkeiten und Kenntnisse erwartet werden.

4 Diskussion

Insgesamt lässt sich feststellen, dass die Dozent*innen in der Informatik überwiegend allgemeine Fähigkeiten und Kenntnisse bei Studienanfänger*innen voraussetzen. An fachlichen Fähigkeiten und Kenntnissen werden vor allem mathematische Fähigkeiten und formales Vorgehen erwartet. Die Auswertung mit Hilfe des Akzeptanz-Kriteriums bestätigt dies. Auch die Faktorenanalyse zeigt, dass die motivationalen und kommunikativen Soft Skills mit etwas höheren Prozentzahlen erwartet werden als mathematische Fähigkeiten. Auch bei den fachlichen Vorkenntnissen werden überwiegend mathematische Fähigkeiten und Denkweisen erwartet - aus Bereichen, die dem Faktor Informatik zugeordnet werden, hingegen quasi gar keine. Laut Ansicht der Dozent*innen haben Studierende mit *Schreiben formalisierter Schreibweise, Lesen formalisierter Schreibweise und Modellierung* die meisten Probleme - also mit Fähigkeiten und Kenntnisse aus Bereichen, die die Dozent*innen nicht voraussetzen. Da es keine Übereinstimmung zwischen erwarteten fachlichen Fähigkeiten und Kenntnissen und problembehafteten Bereichen gibt, deutet die Umfrage darauf hin, dass fehlendes fachliches Vorwissen - insbesondere in den ersten Semestern - wahrscheinlich nicht die Ursache dieser Probleme ist. Der laut Heublein [He10] am häufigsten genannte Grund für den Studienabbruch - hohe Leistungsanforderung - kann mit Blick auf die Daten der hier vorgestellten Studie nicht mit einer Diskrepanz zwischen den erwarteten fachlichen Vorkenntnissen und

6 The R Project for Statistical Computing, https://www.r-project.org/, Version 3.0.5.

Fähigkeiten und den tatsächlich vorhandenen erklärt werden, da oftmals keine oder nur wenige Informatikkenntnisse vorausgesetzt werden. Offen bleibt dabei die Frage, ob die Vorstellung, was genau unter die fachlichen Fähigkeiten und Kenntnisse fällt (also was z.B. genau der Bereich *Kenntnisse über elementare Algorithmen* umfasst), zwischen den verschiedenen Gruppen identisch ist. Die Beantwortung dieser Frage ist aber außerhalb des Umfangs dieser Umfrage. Die Ursachen für einen Studienabbruch lassen sich also nicht einfach mit zu hohen fachlichen Anforderungen begründen und sollten weiter untersucht werden. Die Auswertung anhand der Akzeptanz-Kriterien ist uneindeutig, da sich für die meisten (insbesondere die fachlichen) Fähigkeiten und Kenntnisse keine klare Tendenz über die Erwartungen der Dozent*innen erkennen lässt (also „vorausgesetzt" bzw. „explizit nicht vorausgesetzt"). Dies könnte darauf hindeuten, dass - obwohl eindeutige Trends erkennbar sind - in vielen Fällen noch kein Konsens zwischen den Dozent*innen in der Informatik darüber besteht, welches Vorwissen Studienanfänger*innen haben sollten. Leider kann der Umfrage nicht entnommen werden, inwieweit die Unterschiede in der Erwartungshaltung mit unterschiedlichen Lehrerfahrungen (z.B. Bundesland, Hochschulart) zusammenhängen. Das Ergebnis könnte auch auf Schwachstellen in der Umfrage (z.B. Likert-Skala, unklare Formulierung der Items) hindeuten. Auch in den problembehafteten Bereichen ist kein Konsens erkennbar. Zwar gibt es einige Bereiche (wie z.B. *Formale Sprache*) die laut den Dozent*innen in der Informatik den Studienanfänger*innen eher Probleme bereiten und andere, bei denen dies nicht der Fall ist. Allerdings lässt sich für viele Bereiche keine eindeutige Aussage treffen. Auch hier lässt die Datenlage keinen Schluss zu, ob dies mit unterschiedlichen Lehrerfahrungen (z.B. Bundesland, Hochschulart) zusammenhängt. Ebenfalls könnte, wie oben beschrieben, die Ursache auch im Umfragedesign begründet sein.

Die Mehrheit der Dozent*innen in der Informatik ist der Meinung, dass Informatik in der Schule für ein erfolgreiches Informatikstudium nicht notwendig ist. Dies spiegelt die momentane Lage des Informatik-Schulunterrichts in Deutschland wider. Informatik ist bislang bundesweit kein Pflichtfach und wird dementsprechend nicht von allen Schüler*innen in der Schule belegt. Inwieweit Vorkenntnisse in Informatik - insbesondere durch Schulunterricht - ein Indikator für Studienerfolg sind, wurde in der Umfrage nicht untersucht.

5 Zusammenfassung

Welche Erwartungen haben Dozent*innen der Informatik an Studienanfänger*innen? Zur Beantwortung dieser Frage wurde eine deutschlandweite Umfrage durchgeführt, an der 588 Informatik-Hochschuldozent*innen teilgenommen haben. Die Umfrage ergab, dass hauptsächlich allgemeine Fähigkeiten und Kenntnisse (hier insbesondere *Interesse an Informatik, logisches Denkvermögen* und *Motivation*) vorausgesetzt werden. Bei den fachlichen Fähigkeiten und Kenntnissen überwiegen vor allem mathematische Fähigkeiten bzw. formales Vorgehen. Dadurch liegt die Vermutung nahe, dass die hohe Zahl an Abbrüchen beim Informatikstudium nicht an der Diskrepanz zwischen den von den Dozent*innen erwarteten und tatsächlich vorhandenen Fähigkeiten und Kenntnissen fachlicher und allgemeiner Natur liegt. Die Umfrage hat gezeigt, dass es keine erwarteten Fähigkeiten gibt, die dem Akzeptanzkriterium genügen. Dies deutet darauf hin, dass ein solcher Konsens unter den Dozent*innen in Deutschland (beispielsweise durch unterschiedliche Lehrerfahrungen) noch nicht gegeben ist.

Literaturverzeichnis

[BLH01] Beaubouef, T.; Lucas, R.; Howatt, J.: The UNLOCK system: Enhancing problem solving skills in CS-1 students. ACM SIGCSE Bulletin, 33(2):43–46, 2001.

[Co14] Cooperation Schule-Hochschule, cosh: Mindestanforderungskatalog Mathematik (Version 2.0) der Hochschulen Baden-Württembergs für ein Studium von WiMINT-Fächern (Wirtschaft, Mathematik, Informatik, Naturwissenschaft und Technik). Online erreichbar unter: https://lehrerfortbildung-bw.de/u_matnatech/mathematik/bs/bk/cosh/katalog/makv2.pdf, 2014.

[CPV05] Cappel, J.; Prybutok, V.; Varghese, B.: A closer look at attention to detail. Communications of the ACM, 48(7):87–92, 2005.

[He10] Heublein, U.; Hutzsch, C.; Schreiber, J.; Sommer, D.; Besuch, G.: Ursachen des Studienabbruchs in Bachelor- und in herkömmlichen Studiengängen: Ergebnisse einer bundesweiten Befragung von Exmatrikulierten des Studienjahres 2007/08. Forum Hochschule, 2010(2), Hannover: Deutsches Zentrum für Hochschul- und Wissenschaftsforschung (DZHW)., 2010.

[MSZ18] Müller, U. C.; Sitzmann, D.; Zimmermann, S.: Naturwissenschaftliche Bildung als Grundlage für berufliche und gesellschaftliche Teilhabe. C. Maurer, 2018. 39, 918-921, Kiel: Gesellschaft für Didaktik der Chemie und Physik.

[NPH17] Neumann, I.; Pigge, C.; Heinze, A.:Welche mathematischen Lernvoraussetzungen erwarten Hochschullehrende für ein MINT-Studium. Eine Delphi-Studie. Kiel: IPN, 2017.

[Pu08] Puhlmann, H.: Bildungsstandards Informatik für die Sekundarstufe I (Januar 2008). 2008.

[Rö16] Röhner, G.; Brinda, T.; Denke, V.; Hellmig, L.; Heußer, T.; Pasternak, A.; Schwill, A.; Seiffert, M.: Bildungsstandards Informatik für die Sekundarstufe II. Beilage zu LOG IN, 183(184):88, 2016.

[Si19] Sitzmann, D.; Soll, M.; Barbas, H.; Hamann, F.; Bender, E.: Entwicklung eines Informatik-Onlinetests zur Studienvorbereitung im Projekt MINTFIT Hamburg. In B. Meissner, C. Walter, B. Zinger, J. Haubner, F. Waldherr (Hrsg.), Tagungsband zum 4. Symposium zur Hochschullehre in den MINT-Fächern, S. 277–285, 2019.

[WB10] Wolff, H.-G.; Bacher, J.: Hauptkomponentenanalyse und explorative Faktorenanalyse. In: Handbuch der sozialwissenschaftlichen Datenanalyse, S. 333–365. Springer, 2010.

Automatisches kompetenzbasiertes Feedback in der Informatiklehre

Jana Schmitz[1]

Abstract:
Diese Arbeit untersucht den Forschungsstand beim automatischem, kompetenzbasiertem Feedback in der Informatiklehre. Es wurde ein systematisches Literatur Review durchgeführt. Es konnten acht relevante und zwölf teilweise relevante Veröffentlichungen gefunden werden. Die Veröffentlichungen wurden hinsichtlich der zwei Forschungsfragen analysiert: Erstens ob Kompetenzmodelle verwendet und konkrete Kompetenzen genannt werden. Zweitens welche Gestalt bzw. Form das beschriebene Feedback hat. Insgesamt wurde festgestellt, dass nur wenige Arbeiten Kompetenzdefinitionen verwenden und die verwendeten Definitionen sehr divers sind. Auch wird in keiner der Arbeiten ein konkretes Kompetenzmodell verwendet.

Keywords:
Informatik; Literatur Review; Feedback; Kompetenzen; Assessment

1 Einleitung

Feedback ist ein wichtiger Bestandteil von Lernprozessen. Teilweise wird es sogar als einer der wichtigsten Faktoren für einen guten Lernerfolg beschrieben. Feedback zu geben ist jedoch kein trivialer Vorgang. Effizientes Feedback soll Lernenden dabei helfen die Kluft zwischen dem aktuellen Lernstand und dem angestrebten Lernziel zu überbrücken. Lehrende müssen somit die individuellen Bedürfnisse und den Lernstand eines Lernenden adressieren, um den Lernerfolg gezielt zu fördern. Insbesondere bei größeren Gruppen von Lernenden ist es zudem schwierig auf einzelne Personen speziell einzugehen. Lernenden ein gutes und individuelles Feedback zu geben, kann so eine didaktische und organisatorische Herausforderung sein. [HT07] Eine individuelle Betreuung durch Lehrpersonen ist bei den aktuell steigenden Studierendenzahlen nicht oder nur eingeschränkt möglich. Abhilfe schaffen hier seit einigen Jahren automatische E-Learning- bzw. Assessmentsysteme. Die Generierung von automatischem Feedback durch Tools ist jedoch eine weitere Herausforderung. Die Evaluation, ob eine Aufgabe korrekt ist, ist dabei durch automatische Systeme verhältnismäßig einfach umzusetzen und teilweise sogar einfacher als eine manuelle Korrektur. Beispiele hierfür sind z.B. Multiple-Choice Aufgaben oder das automatische Testen von Programmcode. Binäres Feedback (korrekt/falsch) allein reicht bei komplexeren Aufgaben jedoch nicht aus, um den Lernerfolg zu fördern [La16]. Für komplexe Aufgaben ist Feedback eine wesentlich größere Herausforderung, z.B. automatisches Feedback zu natürlichsprachlichen Texten. Aber auch komplexeres Feedback für einfachere Aufgaben, das über binäres Feedback - oder die Anzeige an welcher Stelle ein Fehler gemacht wurde - hinausgeht, ist schwieriger automatisch zu generieren. Das automatische Feedback sollte die Lernenden unterstützen und bestenfalls - ebenso wie das Feedback durch eine Person - alle Feedbackebenen umfassen und helfen die Diskrepanz zwischen Lernstand und Lernziel zu überwinden. Dazu mussten geeignete Tools entwickelt und Möglichkeiten gefunden werden, automatisches Feedback zu generieren, das den Lernerfolg genauso fördert wie persönliches Feedback. Durch umfassende Forschung in den letzten Jahrzehnten, gibt es bereits eine Vielzahl an erprobten automatischen Feedback- und Assessmentsystemen in diesem Bereich. Einige Tools z.B. zur automatische Bewertung von Programmieraufgaben, werden bereits seit Jahrzehnten erfolgreich in der Lehre eingesetzt [KJH18].

Zusätzlich zur stetigen Erhöhung der Studierendenanzahl, fand auch der kompetenzbasierte Ansatz z.B. durch die Bologna Reform, immer mehr Beachtung in der Lehre. Wird kompetenzbasiert gelehrt und geprüft, muss auch das Feedback an die Lernenden kompetenzbasiert sein. Das heißt auch, dass die automatischen Feedback- und Assessmentsysteme an den kompetenzbasierten Ansatz angepasst werden müssen. Insbesondere das automatisch generierte Feedback sollte dann die Kompetenzen, die erworben werden sollen, adressieren. Das heißt, das Feedback muss die für eine Aufgabe relevanten Kompetenzen ansprechen und angeben, ob und inwieweit diese beherrscht werden. Es mussten Tools entwickelt werden, die Kompetenzen prüfen und gezielt Feedback

1 Jana Schmitz, Universität Duisburg-Essen, Zentrum für Informations- und Mediendienste, Schützenbahn 70, 45127 Essen, Deutschland, Jana.Schmitz@uni-due.de

geben können, um den Kompetenzerwerb zu fördern. Im Rahmen dieser Arbeit wurde ein Literatur Review zum automatischem, kompetenzbasiertem Feedback in der Informatiklehre durchgeführt, um den aktuellen Forschungsstand in diesem Bereich zu erkunden.

1.1 Forschungsfragen

Folgende Forschungsfragen wurden im Rahmen des Literatur Reviews betrachtet: *F1: Wird ein Kompetenzmodell genutzt? Wenn ja, welche Kompetenzen* und *Kompetenzstufen werden genannt?* und *F2: Welche Art(en) von Feedback wird/werden beschrieben bzw. verwendet?* Die erste Forschungsfrage beschäftigt sich damit, ob in den betrachteten Arbeiten Kompetenzmodelle, konkrete Kompetenzen oder Kompetenzstufen beschrieben werden. Mit dieser Forschungsfrage wird untersucht, ob die Autoren formale Aspekte - z.B. ein Kompetenzmodell - in die Entwicklung von automatischem, kompetenzbasiertem Feedback mit einbeziehen. Klieme und Leutner [KL06] sehen Kompetenzmodelle als Ausgangspunkt für Messverfahren zur Kompetenzmessung. Auch Kramer et al. [KHB16] geben an, dass für eine Messung von Kompetenzen Kompetenzmodelle notwendig sind. Somit soll mit dieser Frage die fachliche Basis, auf der das automatische, kompetenzorientierte Feedback entwickelt wurde, analysiert werden. Die zweite Forschungsfrage untersucht welche Arten von Feedback beschrieben werden. Werden im Rahmen des Feedbacks z.B. konkrete Kompetenzen genannt, welche erfüllt/nicht erfüllt sind oder werden Hinweise auf konkrete Fehler gegeben? Zusätzlich wird untersucht, ob zum kompetenzbasiertem Feedback weiteres nicht-kompetenzorientiertes Feedback gegeben wird. Durch diese Forschungsfrage soll die konkrete Umsetzung und die verschiedenen Ansätze, kompetenzbasiertes Feedback zu geben, analysiert werden.

2 Theoretischer Hintergrund

Zunächst werden einige theoretische Konzepte betrachtet, die für das weitere Verständnis wichtig sind.

Feedback.
Hattie und Timperley beschreiben Feedback als Information, die von einem Agenten (Lehrer, Peer, Buch, Eltern, einem selbst, Erfahrungen) bereitgestellt wird und Aspekte des Verständnisses oder Leistung des Feedbackempfängers behandelt [HT07, S. 102]. Es soll die Diskrepanz zwischen dem aktuellen Leistungsstand und dem angestrebten Lernziel reduzieren [HT07]. Feedback ist somit ein wichtiger Baustein des Lernprozesses. Nach Hattie ist Feedback sogar eine der effektivsten Strategien zum Lernen [HMB16]. Die Formulierung von gutem Feedback ist zudem kein trivialer Vorgang. Feedback ist laut Hattie und Timperley immer die Konsequenz aus Leistung [HT07, S. 81]. Es muss zudem an einen konkreten Lernkontext adressiert sein, da es in einem „Vakuum" keinen Effekt habe [HT07, S. 82].

Kompetenzen und Kompetenzmodelle.
Für den Begriff „Kompetenz " gibt es eine Vielzahl an unterschiedlichen Definitionen. Ein Beispiel ist die Definition nach Weinert [We01]: „Kompetenzen sind die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können" [SS11, S. 111]. In der englischsprachigen Literatur wird gelegentlich eine Unterscheidung der Begriffe *competence und competency* vorgenommen. Im Deutschen gibt es diese Unterscheidung nicht, weswegen in dieser Arbeit die Begriffe als synonym verstanden werden. Mithilfe von Kompetenzmodellen können Kompetenzen systematisch geordnet und so die Aspekte, Abstufungen und Entwicklungsverläufe von Kompetenzen dargestellt werden. Kompetenzmodelle sind zudem notwendig, um Kompetenzen messbar zu machen [KHB16]. Klieme et al. [KHR08] beschreiben drei Arten von Kompetenzmodellen: Competency Structure Models (Kompetenzstrukturmodelle), welche die kognitiven Voraussetzungen, die Lernende benötigen, um Aufgaben und Probleme in einem bestimmten Inhalts- oder Anforderungsbereich zu lösen beschreiben. Competency Level Models (Kompetenzstufenmodelle) geben Auskunft über die Stufen bzw. Profile der beschriebenen Kompetenzen und Competency Development Models (Kompetenzentwicklungsmodelle), beschreiben wie sich Kompetenzen über die Zeit entwickeln sollen.

Kompetenzorientiertes Feedback.
Eine allgemeine Definition für kompetenzbasiertes Feedback konnte in der Literatur nicht gefunden werden. In einigen Arbeiten wird im Bezug auf kompetenzbasiertes Feedback beschrieben, dass nicht nur konkrete Fehler aufgezeigt, sondern die Konzepte (Kompetenzen) genannt werden, die zur Lösung der Aufgabe benötigt werden. Zudem werden die fehlenden Kompetenzen genannt, die zu dem Fehler geführt haben könnten. [Kr20] [Um14] In dieser Arbeit wird kompetenzbasiertes Feedback so verstanden, dass die zu erwerbenden oder bereits erworbenen Kompetenzen adressiert werden. Durch Feedback soll angezeigt werden, ob eine Kompetenz beherrscht wird. Zusätzlich wird ggf. quantifiziert zu welchem Grad eine Kompetenz bereits erworben wurde. Das

Ziel von kompetenzbasiertem Feedback ist dabei immer den Kompetenzerwerb zu fördern. Ein einfaches Beispiel für kompetenzbasiertes Feedback wäre z.B., dass bei einer Rechenaufgabe nicht nur angegeben wird, ob das Ergebnis falsch ist (der konkrete Fehler), sondern auch genannt wird, welche Kompetenzen für die Lösung der Aufgabe benötigt werden z.B. „Subtraktion“ und „Division“. Zusätzlich könnte auch die konkrete Kompetenz genannt werden, die (nicht) beherrscht wird.

3 Methode

In diesem Abschnitt wird dargestellt wie bei der Durchführung des Literatur Review vorgegangen wurde.

3.1 Kriterien

Im Rahmen des Reviews wurden nur wissenschaftliche Veröffentlichungen betrachtet, welche in englischer oder deutscher Sprache verfasst wurden, und sich mit kompetenzbasiertem Feedback in der Informatiklehre beschäftigen. Hierbei wurden nicht nur Lehrveranstaltungen, welche direkt in Informatikstudiengängen durchgeführt wurden betrachtet. Es wurden auch Lehrveranstaltungen, welche anderen Studiengängen zuzuordnen sind, die sich mit Informatikthemen beschäftigen untersucht. Auch Informatiklehre im Rahmen der schulischen Ausbildung wurde betrachtet. Das gegebene Feedback sollte automatisch oder zumindest semi-automatisch sein und nicht ausschließlich durch Personen (Lehrpersonen, Tutoren, andere Lernende) gegeben werden. Ebenso sollte das Feedback mindestens in Teilen kompetenzbasiert sein bzw. von den Autoren als kompetenzbasiert bezeichnet werden. Das bedeutet, dass auch Arbeiten betrachten wurden in denen das Feedback und weitere Bestandteile als kompetenzorientiert oder als Kompetenzen von den Autoren bezeichnet werden, es sich aber nach allgemeinem Verständnis (siehe Abschnitt 2) nicht um kompetenzorientiertes Feedback oder Kompetenzen handelt.

3.2 Suchprozess

Search-String Entwicklung
Startpunkt der Search-String Entwicklung war eine erste manuelle Suche mit der Suchmaschine Google Scholar. Dabei konnten fünf möglicherweise relevante Veröffentlichungen gefunden werden [Kr20] [Ga19] [Hu17] [Um14] [Um15]. Anhand dieser Veröffentlichungen wurden Begrifflichkeiten und Schlüsselwörter identifiziert.

Der Search-String wurde für die Datenbank Scopus anhand dieser Begriffe entwickelt. Für die Datenbanken IEEE Xplore und Web of Science wurden die Search-Strings basierend auf dem Ersten erarbeitet. Die verwendeten Search-Strings wurden in Tabelle 1 dargestellt. Bei der Datenbank Scopus wurde mithilfe des Search-Strings das Abstract, der Titel und die Keywords nach den gesuchten Begriffen durchsucht. Bei IEEE Xplore wurden alle Datenfelder und somit alle Metadaten außer dem Volltext einbezogen. Bei der Datenbank Web of Science wurde nur das Abstract durchsucht. Die Unterschiede begründen sich in der unterschiedlichen Syntax und Möglichkeiten der Datenbanken.

Tab. 1: Search-Strings

Datenbank	Search-String
Scopus	TITLE-ABS-KEY ((„ competency based“ OR „ competence based“ OR „ competence-based“ OR „ competence driven“ OR „ Competency Based“ OR Competence Based OR„ Competency-Based“ OR „ Competency-based“ OR competency model OR competence model) AND (feedback OR hint) AND (learning OR teaching OR education OR assessment OR e-learning OR task)) AND (computer science OR programming) AND (EXCLUDE (SUBJAREA, „ MEDI “)) AND (EXCLUDE (DOCTYPE, „cr“)) AND (EXCLUDE (SUBJAREA, „ DENT“))
IEEE Xplore	(„ competency based“ OR „ competence based“ OR „ competence-based“ OR „ competence driven“ OR „ competency model“ OR „ competence model“) AND („ feedback“ OR „ hint“) AND („learning„ OR “ teaching“ OR „ education“ OR „ assessment“ OR „ e-learning“ OR „ task“)
Web of Science	(„ competency based“ OR „ competence based“ OR „ competence-based“ OR „ competence driven“ OR „ competency model“ OR „ competence model“) AND(„ feedback“ OR „ hint“) AND („ learning“ OR „ teaching“ OR „ education“ OR „ assessment“ OR „ e-learning“ OR „ task“)

Die Suche
Die Suche in den Datenbanken wurde im Oktober 2020 durchgeführt. Dabei wurden insgesamt 71 Veröffentlichungen ohne Duplikate gefunden. In der Datenbank Scopus konnten 42, in IEEE Xplore 23 und in der Datenbank Web of Science sechs möglicherweise relevante Veröffentlichungen gefunden werden. Die Veröffentlichungen wurden zunächst anhand des Abstracts und der in den Datenbanken vergebenen Keywords in drei Kategorien einsortiert: *Konkret, Allgemein* und *Ausschuss*. Die Kategorie *Konkret* beinhaltet Veröffentlichungen, die allen Einschlusskriterien entsprechen. Die Kategorie *Allgemein* enthält alle Veröffentlichungen, die grundsätzlich die Themenbereiche, mit denen sich das Review beschäftigt, ansprechen und interessante Aspekte enthalten, aber ggf. nicht alle Einschlusskriterien erfüllen. Dazu gehören Veröffentlichungen bei denen das Feedback zwar kompetenzbasiert, aber nicht automatisch ist oder die sich mit kompetenzbasiertem Feedback außerhalb der Informatiklehre beschäftigen. Diese Unterscheidung wurde vorgenommen, um nicht vorzeitig möglicherweise relevante Arbeiten auszusortieren, aber einen besseren Überblick über die gefundene Literatur zu bekommen. Im Anschluss an die Suche in den Datenbanken wurde eine Schneeballsuche durchgeführt. Hierfür wurden die Quellen der Veröffentlichungen aus den Kategorien *Konkret* und *Allgemein* untersucht und weitere relevante Literatur identifiziert. Es wurden zunächst 18 möglicherweise relevante Veröffentlichungen gefunden. Diese wurden gelesen und in die drei Kategorien eingeordnet. Zwei konkrete und fünf allgemeine Veröffentlichungen konnten gefunden werden. Im Anschluss an die Schneeballsuche wurde die Einordnung in die Kategorien noch einmal anhand der Einschlusskriterien überprüft und Paper von *Konkret* nach *Allgemein* herabgestuft oder aussortiert.

Die Suchergebnisse
Acht Veröffentlichungen wurden der Kategorie *Konkret* und zwölf Veröffentlichungen der Kategorie *Allgemein* zugeordnet. 64 Veröffentlichungen wurden nach dem Lesen als nicht relevant deklariert. Auf insgesamt fünf potentiell relevante Veröffentlichungen konnte nicht zugegriffen werden. Alle relevanten Veröffentlichungen sind in englischer Sprache verfasst. Mithilfe einer Datenextraktionstabelle wurden die Inhalte der Veröffentlichungen strukturiert erfasst und festgehalten. Aufgrund der nur geringen Anzahl an Veröffentlichungen wurden die Arbeiten aus der Kategorie *Allgemein* nicht verworfen. Dieses Paper wird sich im weiteren mit den insgesamt 20 Veröffentlichungen aus den Kategorien *Konkret* und *Allgemein* befassen.

4 Ergebnisse und Diskussion

Insgesamt wurden nur wenige Veröffentlichungen zum Thema automatisches kompetenzbasiertes Feedback in der Informatiklehre gefunden. Keine der Veröffentlichung wurde vor 2009 publiziert. Drei (*Konkret*) bzw. sechs (*Konkret und Allgemein*) der Arbeiten sind aus den Jahren 2020 und 2019. Die meisten Veröffentlichungen stammen aus den Jahren 2013/2014. Die geringe Anzahl an gefundenen Veröffentlichungen, sowie die jungen Veröffentlichungsdaten deuten darauf hin, dass das automatische, kompetenzbasierte Feedback in der Informatiklehre ein relativ neuer Forschungsbereich ist.

Verwendung von Kompetenzdefinitionen
Einige der Veröffentlichungen nennen die Bologna Reform als Grund sich mit der kompetenzbasierten Lehre zu befassen. Obwohl die kompetenzbasierte Lehre durch die Bologna Reform in den Fokus gerückt wurde, scheint es keine einheitlich verwendete Definition des Begriffs *Kompetenz* zu geben. Wie bereits festgestellt, gibt es eine Vielzahl an verschiedenen Definitionen des Konzepts. In fünf (Kategorie *Konkret*) der betrachteten Veröffentlichungen [Al09] [CGH19] [FGF13] [Ku14] [Ta14] wird der Begriff ohne vorherige Definition verwendet. In drei der Veröffentlichungen werden Definition gegeben [Kr20] [Um14] [Um15]. Auch bei den Veröffentlichungen der Kategorie *Allgemein* enthalten nur vier [Hu17] [PCP16] [BS13] [DS14] von zwölf Veröffentlichungen eine Definition des Begriffs. Es werden insgesamt 9 verschiedene Definitionen verwendet, wobei einige Paper mehrere Definitionen nutzen [Hu17] [Kr20] [Um14] [Um15]. Es lässt sich somit feststellen, dass häufig keine Kompetenzdefinition genannt und bei der Verwendung von Definitionen keine einheitliche Definition verwendet wird. Die fehlende (einheitliche) Begriffsbestimmung führt dazu, dass Arbeiten schwerer zu verstehen und unterschiedlich zu interpretieren sind. Teilweise wird der Begriff „Kompetenz" auch sehr unspezifisch für Bereiche verwendet, bei denen es sich nach dem allgemeinen Verständnis nicht um Kompetenzen handelt.

4.1 Forschungsfrage F1

Zur Beantwortung der ersten Forschungsfrage wurde untersucht, ob Kompetenzmodelle angesprochen oder verwendet und konkrete Kompetenzen genannt werden. Drei Veröffentlichungen beschäftigen sich mit Kompetenzmodellen [Al09] [FGF13] [Kr20]. Fünf Veröffentlichungen beschäftigen sich nicht mit Kompetenzmodellen [CGH19] [Um14] [Um15] [Ta14] [Ku14].

Krugel et al. [Kr20] geben in ihrer Arbeit an, dass es in der Informatik nur wenig empirische Forschung zu Kompetenzmodellen gäbe. Sie stellen zudem fest, dass es kein ausreichend detailliertes und empirisch fundiertes Kompetenzmodell im Bereich der objektorientierten Programmierung gibt. Die Autoren nutzen somit kein konkretes Kompetenzmodell in ihrer Arbeit, stützen sich

jedoch auf die beschriebene vorangegangene Forschung. Zusätzlich beschäftigen sich Krugel et al. [Kr20] in ihrer Arbeit mit der empirischen Definition von Kompetenzen bzw. Kompetenzkomponenten im Bereich der objektorientierten Programmierung.

Florian-Gaviria et al. [FGF13] befassen sich mit einer Software Suite die Lehrende unterstützen soll kompetenzbasierte Lehre und den Europäischen Qualifikationsrahmen (EQR) in online und blended Kursen umzusetzen. Der EQR soll die Bildung eines europäischen Bildungsraums ermöglichen und als Übersetzungs- und Vergleichsinstrument von Kompetenzen und Qualifikationen dienen [FGF13] [No15]. Die Autoren beschreiben, dass der EQR nicht in allen gleich detailliert ausgearbeitet wurde. Sie führen hier als Beispiel den Spracherwerb auf, welcher sehr detailliert ausgearbeitet und beschrieben wurde. Für andere Bereiche gibt es eine solche Ausarbeitung nicht. Die Autoren sehen hier die Lehrenden in der Pflicht, Kompetenzmodelle für ihre Kurse, bei Bereichen ohne detaillierte Ausarbeitung, zu erstellen. Die Autoren nennen nur im Rahmen der durchgeführten Case Study konkrete Kompetenzen, in der weiteren Arbeit werden diese nur abstrakt behandelt. Als Beispiel für eine Kompetenz wird z.B. die „Ability to use programming paradigms and languages" genannt. Neben dem EQR an sich wird kein konkretes Kompetenzmodell genannt. Kompetenzstufen ergeben sich daher, dass Kompetenzenden Leveln des EQRs zugeordnet werden. Alisnet et al. [Al09] nutzen den Begriff „competence model" nicht. Die Autoren stellen jedoch einen Ansatz für die Darstellung und Modellierung von Kompetenzen vor. Sie nutzen Ontologien zur Modellierung von Kompetenzen und Abhängigkeiten. Der Ansatz kann als Modellierung eines Kompetenzmodells verstanden werden. Jedoch lässt sich dies, da ein direkter Bezug zu Kompetenzmodellen fehlt und auch keine Definitionen verwendet werden, nicht bestimmen. Die Autoren sprechen Kompetenzen nur abstrakt an und nennen keine Konkreten.

Nur drei der Arbeiten befassen sich im weitesten Sinne mit Kompetenzmodellen. Jedoch nennt oder nutzt keine der Arbeiten ein konkretes Kompetenzmodell. Im Folgenden wird untersucht inwieweit die weiteren Arbeiten konkrete Kompetenzen und Kompetenzstufen ansprechen. Kumar et al. [Ku14] und Umbleja et al. [Um14] befassen sich nicht mit Kompetenzmodellen und nennen keine konkreten Kompetenzen oder Kompetenzbereiche. Auch die folgenden Veröffentlichungen nennen kein Kompetenzmodell, sprechen aber Kompetenzen, Kompetenzbereiche und ähnliche Konzepte an.

Umbleja [Um15] beschreibt die Kompetenzen im Text sehr abstrakt. In einer Beispielgrafik werden konkrete Kompetenzbereiche genannt, z.B. *Hashing*, *Stack* und *Processes*. Die Veröffentlichung beschreibt eine abgestufte Bewertung von Kompetenzen. Diese werden mit einem 128-stufigen System bewertet; ab Stufe 77 gilt eine Kompetenz als erworben. In den weiteren Veröffentlichungen wird nicht konkret auf die verwendeten Bewertungsskalen eingegangen. Es wird lediglich beschrieben, dass diese (mit einer Note) bewertet werden. Da Cruz Alves et al. [CGH19] haben im Rahmen ihrer Arbeit ein Literatur Review zu Assessments bei blockbasierten Programmiersprachen in der schulischen Lehre durchgeführt. Sie untersuchen welche Form von Assessment genutzt werden kann, um „computational thinking" Kompetenzen bewerten zu können. Es werden keine konkreten Kompetenzen genannt sondern „Subkonzepte", zu den Bereichen Programmierung und Algorithmen angesprochen, die von den betrachteten Assessmentsystemen einbezogen werden z.B. *algorithms, variables, code organisation, documantation und aesthetics*.

Tashiro et al. [Ta14] nennen 10 konkrete Kompetenzbereiche z.B. *Information Technology Foundations und Software Engineering and Development*. Jeder Kompetenzbereich wird dann in weitere „Objectives" aufgeteilt und von diesen „Objectives" konkrete „Lessons" abgeleitet. Einzelne konkrete Kompetenzen werden nicht genannt. Nur in drei Arbeiten werden somit konkrete Kompetenzen bzw. Kompetenzbereiche genannt. Davon werden bei zwei Veröffentlichungen die konkreten Kompetenzen und Kompetenzbereiche im Rahmen eines Beispiels erwähnt. Nur Krugel et al. [Kr20] befassen sich explizit mit konkreten Kompetenzen und deren Definition.

4.2 Forschungsfrage F2

Die zweite Forschungsfrage untersucht die Feedback Arten. In allen acht Veröffentlichungen der Kategorie *Konkret* wird automatisches, kompetenzbasiertes Feedback in unterschiedlicher Form angesprochen. Krugel et al. [Kr20] sprechen automatisches, kompetenzbasiertes Feedback nur am Rande an. Die Autoren möchten mit den Ergebnissen ihrer Arbeit das automatische Feedback des Assessmentsystems „JACK" verbessern. In der Arbeit von Da Cruz Alves et al. [CGH19] wird automatisches, kompetenzbasiertes Feedback im Rahmen eines Literatur Reviews diskutiert. Die Autoren untersuchen wie in den betrachteten Arbeiten Feedback gegeben wird. Sie stellen fest, dass einige der Tools nur ein binäres Feedback geben. Andere Tools unterteilten den Code in Bereiche bzw. Kompetenzen und bewerteten diese einzeln. Zusätzlich wird bei manchen Ansätzen eine Gesamtnote aus den einzelnen Noten gebildet. Nur einige wenige Ansätze geben Hinweise um den Lernprozess zu unterstützen. Zusätzlich sei der Fokus stark auf Programmierkompetenzen aus den Kompetenzbereichen *algorithms, variables, control* und *modularity* gelegt worden. Bereiche wie *code organisation, documantation, aesthetics, creativity* und *usability* wurden meist nur bei manuellen Ansätzen bedacht. Die Autoren gehen nicht genau darauf ein, inwieweit die Ansätze kompetenzbasierte Lehre und Feedback einsetzen. Die Arbeiten von Umbleja et al. [Um14] und Umbleja [Um15] werden hier gemeinsam betrachtet, da diese aufeinander aufbauen und das gleiche

E-Learningsystem beschreiben. Umbleja et al. [Um14] stellen einen Algorithmus vor, der analysieren soll an welcher Stelle ein Fehler aufgetreten ist und warum dieser gemacht wurde. Den Studierenden wird dann konkretes Feedback zu den Fehlern und zum Kompetenzerwerb gegeben. Umbleja [Um15] stellt fest, dass der kompetenzbasierte Ansatz im Rahmen eines online Kurses, den Studierenden eine hohe Flexibilität und individuelles Lernen ermöglicht, jedoch es für die Lernenden schwerer ist ihren eigenen Lernfortschritt zu erkennen. Das Paper stellt verschiedene Visualisierungen vor, die den Lernenden ihren Lernfortschritt bezüglich der zu erwerbenden Kompetenzen aufzeigen. Die Lernenden können mit diesen Tools ihren Lernfortschritt bzw. Kompetenzerwerb analysieren und dann eine Kompetenz auswählen, die sie erwerben/erweitern wollen. Automatisch wird dann eine passende Aufgabe ausgewählt mit der diese Kompetenz trainiert werden kann.

Tashiro et al. [Ta14] haben den Electronic Learning Record (ELR) ähnlich dem "Patient Data Electronic Healthcare Record" (EHR) aus dem Gesundheitsbereich entwickelt. Durch diesen erhalten Lernende und Lehrende automatisches Feedback über den Lernfortschritt und den Kompetenzerwerb. Neben dem automatischem Feedback wird jede/r Lernende zusätzlich persönlich von einem/einer Mentor/in betreut. Verschiedene Systeme sammeln automatisch Daten über den Lernfortschritt der Studierenden und analysieren diese. Lehrende können so den Lernfortschritt der Studierenden beobachten. Es wird nicht genau beschrieben welche Daten der ELR enthält und wie das Feedback konkret gestaltet ist. Es wird zudem nicht immer deutlich welche Informationen Studierende und welche nur Lehrenden zur Verfügung gestellt werden.

Ein Teil der AEEA Software Suite, die von Florian-Gaviria et al. [FGF13] vorgestellt wird, ist die SOLAR Anwendung. Mithilfe dieser Anwendung wird die durchgeführte Kompetenzanalyse visualisiert und Lehrenden und Lernenden angezeigt. Lehrenden können die Daten all ihrer Studierenden auf einem interaktiven Dashboard sehen und vergleichen. Es wird hauptsächlich die Lehrendenansicht und nicht die Studierendenansicht beschrieben. Die erworbenen Kompetenzen werden dann z.B. mit verschiedenen Graphen visualisiert. Zusätzlich erhalten Lernende beim Assessment auch direktes Feedback zu den Aufgaben. Es wird nicht beschrieben, welche Form dieses Feedback hat und ob dieses ebenfalls kompetenzbasiert ist. Das SCALE Framework, welches in der Arbeit von Kumar et al. [Ku14] vorgestellt wird, besitzt einen interaktiven Visualisierungslayer. Es gibt ein grafisches Interface mit dessen Hilfe die Kompetenzen in Relation zu den Lernaktivitäten dargestellt werden können. Der Lernfortschritt kann so analysiert werden. Zusätzlich dient das Tool auch als Kommunikationsmittel zwischen Lehrenden, Lernenden und Peers. Es wird nicht beschrieben wie das Feedback bzw. der Visualisierungslayer genau gestaltet ist und welche Informationen Lehrende und Lernende konkret bekommen. Neben dem kompetenzbasiertem Feedback, das auf dem Visualisierungslayer gezeigt wird, wird weiteres nicht kompetenzbasiertes Feedback vom Analyselayer zu den Programmieraufgaben produziert. Es werden fünf verschiedene Feedbacktypen beschrieben: *instructions, error detection feedback, compile-time errors, corrective actions* und *solution code snippets*. Das Feedback soll Lernende dabei unterstützen validen Programmcode zu schreiben.

Alisnet et al. [Al09] haben ein Model entwickelt, um Kompetenzen zu modellieren. Um den Kompetenzerwerb messbar zu machen, werden Indikatoren („learning outcomes") definiert. Jedem Indikator wird dann eine Note zugewiesen. Dieses Tool wurde jedoch nicht von den Autoren implementiert, sondern nur theoretisch behandelt.

In den meisten Arbeiten wird das automatische, kompetenzbasierte Feedback nicht ausführlich beschrieben. Es wird nicht genau erläutert wie das Feedback aussieht. Ebenso wird nicht deutlich was Lernende und Lehrende genau angezeigt wird. Nur Umbleja [Um15] hat einen Fokus auf die Tools und Sichten, die verwendet werden um das Feedback darzustellen. Aber auch bei dieser Arbeit liegt der Fokus auf den Tools nicht auf der genauen Beschreibung des Feedbacks.

5 Fazit und Ausblick

Abschließend lässt sich feststellen, dass es sich beim automatischem, kompetenzbasiertem Feedback in der Informatiklehre um einen jungen Forschungsbereich handelt, zu dem noch viel Forschungsbedarf besteht. Die Untersuchung der Veröffentlichungen wirft die Frage nach der Qualität des beschriebenen kompetenzbasierten Feedbacks auf. Insgesamt liefern nur sieben Veröffentlichungen (drei Kategorie *Konkret*, vier Kategorie *Allgemein*) eine Kompetenzdefinition. 13 Veröffentlichungen (fünf Kategorie *Konkret*, acht Kategorie *Allgemein*) verwenden den Begriff Kompetenz, ohne diesen zuvor zu definieren. Die Veröffentlichungen, die Kompetenzdefinitionen angeben, verwenden größtenteils verschiedene Definitionen, die sich inhaltlich unterscheiden. Die Vergleichbarkeit und das Verständnis der Forschungsarbeiten- und ergebnisse wird so erschwert. Zusätzlich wurde festgestellt, dass nur drei Veröffentlichungen (Kategorie *Konkret*) sich mit Kompetenzmodellen beschäftigen. Keine der Veröffentlichungen nennt und nutzt ein konkretes Kompetenzmodell. Es stellt sich so die Frage, auf welcher Basis und in welcher Qualität das in den Veröffentlichungen beschriebene kompetenzbasierte Feedback generiert wird. Aufgrund der fehlenden oder unterschiedlichen Definitionen, sowie der fehlenden Verwendung von Kompetenzmodellen, ist es unklar und auch nicht überprüfbar inwieweit das beschriebene Feedback wirklich den Kompetenzerwerb fördert. Auch wenn Quellen wie z.B. Krugel et al. [Kr20] in ihrer Arbeit

angeben, dass es kaum Kompetenzmodelle und nur wenig (empirische) Forschung an Kompetenzmodellen in der Informatik gibt, gibt es in dieser Domäne bereits Forschung und ausgearbeitete Kompetenzmodelle [Zu16] [KHB16]. Es sollte somit damit begonnen werden, die bereits bestehenden Erkenntnisse in die Entwicklung von Assessmentsystemen einzubeziehen, um das automatische, kompetenzbasierte Feedback auf Basis dieser Forschung zu gestalten. In diesem Paper wurden zudem nur Arbeiten betrachtet, in denen konkrete Begriffe wie „Kompetenz" oder „Kompetenzorientierung" usw. genannt wurden. Es ist jedoch möglich, dass es weitere Forschung in diesem Bereich gibt, in denen diese Begriffe nicht explizit erwähnt, jedoch ähnliche Konzepte verwendet werden und somit der Forschungsbereich ggf. größer ist als in diesem Paper festgestellt.

Literatur

[Al09] Alsinet, T.; Barroso, D.; Béjar, R.; Planes, J.: A Formal Model of Competence-Based Assessment. In: Frontiers in Artificial Intelligence and Applications. Bd. 202, S. 428–436, Jan. 2009.

[BS13] Baumgartner, I.; Shankararaman, V.: Actively linking learning outcomes and competencies to course design and delivery: Experiences from an undergraduate Information Systems program in Singapore. In: 2013 IEEE Global Engineering Education Conference (EDUCON). S. 238–246, 2013.

[CGH19] da Cruz Alves, N.; Gresse von Wangenheim, C.; Hauck, J.: Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study. Informatics in Education 18/, S. 17–39, Apr. 2019.

[DS14] Ducrot, J.; Shankararaman, V.: Measuring student performance and providing feedback using Competency Framework. In: International Conference on Engineering Education (ICEED). S. 55–60, Dez. 2014.

[FGF13] Florian-Gaviria, B.; Glahn, C.; Fabregat Gesa, R.: A Software Suite for Efficient Use of the European Qualifications Framework in Online and Blended Courses. IEEE Transactions on Learning Technologies 6/3, S. 283–296, 2013.

[Ga19] Galan, D.; Heradio, R.; Vargas, H.; Abad Cardiel, I.; Cerrada, J.: Automated assessment of computer programming practices: the 8-years UNED experience. IEEE Access PP/, S. 1–1, Aug. 2019.

[HMB16] Hattie, J.; Mark, G.; Brooks, C.: Instruction Based on Feedback. In (Mayer, R. E.; Alexander, P. A., Hrsg.): Handbook of Research on Learning and Instruction. Taylor & Francis, Kap. 14, S. 376–417, 2016.

[HT07] Hattie, J.; Timperley, H.: The Power of Feedback. Review of Educational Research 77/1, S. 81–112, 2007.

[Hu17] Hubwieser, P.; Berges, M.; Striewe, M.; Goedicke, M.: Towards competency based testing and feedback: Competency definition and measurement in the field of algorithms data structures. In: 2017 IEEE Global Engineering Education Conference (EDUCON). S. 517–526, 2017.

[KHB16] Kramer, M.; Hubwieser, P.; Brinda, T.: A Competency Structure Model of Object-Oriented Programming. In: 2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE). S. 1–8, 2016.

[KHR08] Klieme, E.; Harting, J.; Rauch, D.: The Concept of Competence in Educational Contexts. In (Hartig, J.; Klieme, E.; Leutner, D., Hrsg.): Assessment of Competencies in Educational Context. Hogrefe & Huber Publisher, Toronto, Kap. 1, S. 3–22, 2008.

[KJH18] Keuning, H.; Jeuring, J.; Heeren, B.: A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Trans. Comput. Educ. 19/1, Sep. 2018.

[KL06] Klieme, E.; Leutner, D.: Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen. Beschreibung eines neu eingrichteten Schwerpunktprogramms der DFG. Zeitschrift für Pädagogik 52/, S. 876–903, Jan. 2006.

[Kr20] Krugel, J.; Hubwieser, P.; Goedicke, M.; Striewe, M.; Talbot, M.; Olbricht, C.; Schypula, M.; Zettler, S.: Automated Measurement of Competencies and Generation of Feedback in Object-Oriented Programming Courses. In: 2020 IEEE Global Engineering Education Conference (EDUCON). S. 329–338, 2020.

[Ku14] Kumar, V.; Boulanger, D.; Seanosky, J.; Kinshuk; Panneerselvam, K.; Somasundaram, T. S.: Competence analytics. Journal of Computers in Education/, S. 251–270, 2014.

[La16] Laengrich, M.: Simple feedback can do the job: Analyzing the effects of simple computer based feedback for fundamental programming tasks. In: 2016 IEEE Frontiers in Education Conference (FIE). S. 1–5, 2016.

[No15] Nollmann, A.: Deutscher und Europäischer Qualifikationsrahmen (DQR und EQR): Die Zukunft der beruflichen Bildung. Diplomica Verlag, 2015, isbn: 9783958508675.

[PCP16] Palmer-Brown, D.; Cai, F. F.; Patel, P.: Competency-Based Feedback for the Improvement of Employment Outcomes for Computing Students. In: 2016 International Conference on Computational Science and Computational Intelligence (CSCI). S. 252–257, 2016.

[SS11] Schubert, S.; Schwill, A.: Didaktik der Informatik. Spektrum Akademischer Verlag, 2011.

[Ta14] Tashiro, J.; Hurst, F.; Brown, A.; Hung, P. C. K.; Martin, M.V.: Personalized-Adaptive Learning – A Model for CIT Curricula. In (Cheung, S. K. S.; Fong, J.; Zhang, J.; Kwan, R.; Kwok, L. F., Hrsg.): Hybrid Learning. Theory and Practice. Springer International Publishing, Cham, S. 266–277, 2014, isbn: 978-3-319-08961-4.

[Um14] Umbleja, K.; Kukk, V.; Jaanus, M.; Udal, A.: New concepts of automatic answer evaluation in competence based learning. In. S. 922–925, Apr. 2014, isbn: 978-1-4799-3191-0.

[Um15] Umbleja, K.: Students' grading control and visualisation in competence-based learning approach. In: 2015 IEEE Global Engineering Education Conference (EDUCON). S. 287–296, 2015.

[We01] Weinert, F. E.: Concept of competence: A conceptual clarification. In (Rychen, D. S.; Salganik, L. H., Hrsg.): Defining and selecting key competencies. Hogrefe & Huber, Seattle, WA, S. 45–65, 2001.

[Zu16] Zukunft, O.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016), 2016.

A Scaffolded, Open-Book, Open-Web Exam for a First Programming Course in Java

Debora Weber-Wulff [1]

Abstract:
During the Corona pandemic, university teachers were suddenly confronted with not being able to evaluate courses by a traditional method, using in-person, proctored exams. Open-Book exams have long been discussed in the pedagogical literature, but seem to have seldom been used in German computer science tertiary instruction. In this paper, an example of a scaffolded, openbook, open-web exam for a first programming course at a University of Applied Sciences will be given along with some feedback from the first group of students taking this exam.

Keywords:
assessment, programming instruction, academic integrity.

1 Introduction

In 2009 Jeremy Williams and Amy Wong presented a comparative study of closed-book, invigilated exams vs. open-book, open-web (OBOW) exams. They noted that there is not much literature on the efficacy of the traditional closed-book exam, but that these type of exam seems to be the norm. Why? Perhaps because instructors perceive of them as being the only way to determine if students have actually learned the instruction material.

During the Corona pandemic from 2020 onwards, however, it has been almost impossible to conduct in-person exams due to health concerns. Technology-based proctoring systems have turned out to not only be privacy-invading, but also discriminating and even not functioning as advertised [PB20].

Instructors have been asked to come up with alternative assessments. An OBOW exam fits nicely into this need. OBOW exams give students the possibility to demonstrate what they have learned in a real-life situation, instead of regurgitating inert knowledge. They are asked to apply their learned skills and knowledge instead of puzzling out which of the multiple-choice alternatives is the correct one [WW09, p. 229]. This reduces student stress and removes the necessity of proctoring the exam.

In traditional exams, questions are usually asked on the lower two levels of Bloom's taxonomy of learning goals: knowledge and understanding. It is possible with enough effort to construct multiple-choice questions that address all levels of the taxonomy except synthesis [CDM16]. But with open-book exams it is easier to "climb the ladder" to see if students are able to apply, analyze, synthesize, or even evaluate a situation.

With the rise of sites that offer students answers to homework or exam questions within minutes for an affordable fee, it must also be assumed that those who feel the need to cheat using such services will easily be able to do so. Lancaster & Cotarlan have recently documented a rise in the use of such web sites by STEM students [LC21].

Tricia Bertram Gallant [Be17] summarizes much of the published literature on how to reduce student cheating: Give them mastery-oriented environments; use assessments that are meaningful to students because they see them as relevant to their lives; use authentic, real-world tasks; give the students a choice and some control as to how they are to be graded; and by scaffolding assessments to demonstrate mastery of the material. Jamie McKenzie gives an excellent overview of scaffolded assessment in [Mc99].

1 HTW Berlin, Internationaler Studiengang Medieninformatik, Treskowallee 8, 10313 Berlin weberwu@htw-berlin.de, https://orcid.org/0000-0002-7335-6548

Thus, the pandemic provides a good opportunity to re-think what is being assessed in the exams as they are traditionally given. Are we really determining if the learning goals have been reached? Or are we testing how well people function under time pressure without the use of tools? As part of the work force, people will have to work under pressure, but they will have all the necessary materials and tools available to them.

2 An open-book, open-web exam

During the winter term of 2020/21 I taught an introduction to programming course with Java at the HTW Berlin, a University of Applied Sciences with small classes of around 40 students each. I closely follow [BK17] during the course and use exercises that are adapted from the book. Before the pandemic I normally assigned a sequence of exercises, followed by an in-person, proctored exam with Moodle. The exam included multiple-choice questions, essay questions, and small programming exercises without the use of a compiler. Since in-person exams were not permitted because of the pandemic and it is not possible to effectively proctor distance exams, it was necessary to re-think how to assess the students. The last three of the old sequence of exercises lent themselves to be used, as they were already set up as scaffolded exercises that built upon each other.

2.1 Old sequence of exercises

For the last three exercises in the semester I have usually assigned the progressive development of a small game based on the *Game of Zuul* example that is introduced in the Barnes & Kölling textbook [BK17, Chapter 8]. The goal is for the students to be learning responsibility-driven design, coupling & cohesion, and refactoring. The game plan is a simple collection of rooms that are connected by doors. A player can only move between the rooms in the basic version. It is exceedingly boring.

The students are given a very badly programmed version of *Zuul* that does, however, work. The students are to come up with their own game plan before class, which they find quite exciting. Before class, these are checked to make sure they are not too complicated and changes are suggested if necessary.

In the first of these exercises, the students are asked to refactor the game as done in class, and then add extra functionality to the game. As in all exercises for my programming classes, students submit a report the night before the next exercise session as a PDF that contains their game plan, a description in complete sentences of what they did to make the necessary changes requested, and all of their code in an appendix.

For the next exercise, I make them swap reports with another person, chosen at random. They are to take the code from someone else and extend it with additional functionality, adding items to a room. This causes extreme anguish, as the students quite identify with their own game. That is one of the points of this exercise, to encourage ego-less programming. There is usually someone who has already programmed their game through to completion who then becomes angry. There have also been tears on numerous occasions as they hand over their game-child to a fellow student and get a mess back in return. Again, a report is due describing the condition in which they found the code and what they had to do with it to extend the functionality.

In the last exercise their code is returned to them with many changes made by someone else. They first have to assess the damage and then make a last set of changes that actually makes the game more or less playable. This includes the player being able to pick up items from one room and moving them to another room.

2.2 Scaffolded open-book, open-web exercises

The exercises had to be adapted to the situation of being an open-book, open-web exam. The first problem that had to be solved was the game plan itself. It was clear that the Internet is filled with already solved answers to the additions to the *Zuul* game itself, as the textbook is very popular. Thus I decided that all students would program the same game. The "exam" would consist of three rounds of 90 minutes each, done on three consecutive weeks during the normal exercise session time slot.

A game was designed that fits in with the current Corona crisis. In the final version the player will have to obtain a key from the secretary's office to open the freezer and get the vaccine that must be transported outside. The room plan is given in Figure 1. This ensures that there are no solutions available anywhere on the Internet.

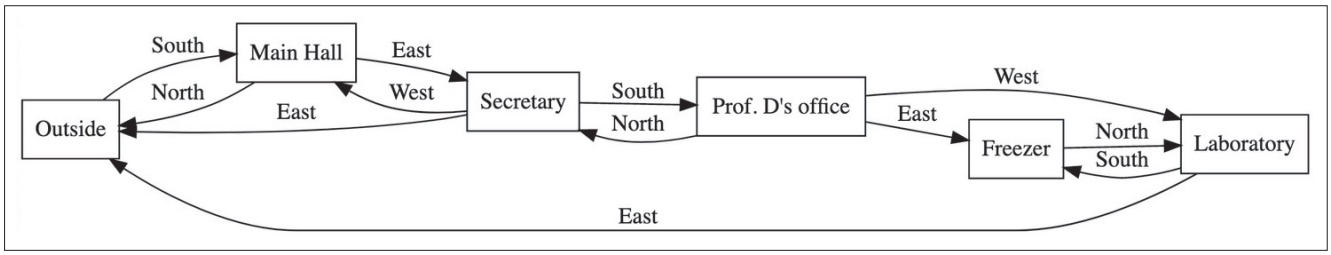

Fig. 1: The 'Stop the Virus' game: Bring the vaccine out of the freezer to the outside

Although switching results with other students is an excellent exercise, it is not useful in an examination situation, as it is possible that a student would have to exchange with a student who did a poor job on the extensions, or that a particular student does not even hand in their exercise. The solution was for me to program a good solution to the first and the second set of additions, so that a student could choose to either continue on their own implementation or to use mine as a starting point.
The first round is a simple repetition of something that had been presented in the lectures. Since this was recorded on video in the university *Mediathek*, it was actually possible to solve Round 1 by just re-watching the video, stopping it after every step, and following the instructions for this new game plan. Many reported doing so, although some reported having trouble finding the right video. Having taken good notes turned out to be extremely useful for this round.

The second round involved applying what had been learned in class to a new situation. In particular, the student had to decide if they were only to add new fields to a room class to keep the item information, or if they should design a new Java class for items. The latter was more work at first, but paid off in later tasks.

The third round demanded much more knowledge about how to work with objects and having items moving through the rooms. A backpack had to be modeled and creativity was encouraged by the introduction of non-player characters. This scaffolding, beginning with a broad and simple basis, continuing though more advanced concepts and ending with creative touches, permitted students to work their way up to the level that was satisfying for them. Ambitious students could spend the time necessary to get everything finished, if they so chose.

3 Detailed mechanics of the exam

There were two exercise groups consisting of 27 persons each who both had exercise sessions on Mondays in parallel with their mathematics exercises. Group one met at 12:15 for programming while group two was at maths, they switched after a short break at 14:00. Each session is 90 minutes. The following setup was used:

- During the normal exercise session for each group everyone first met in Zoom as a group and I explained the tasks. Then each student was put out to their own breakout room (or could leave, if they so wished). I was available in the main room to answer questions during this time.

- The expectation was that the students only need work for 90 minutes, a regular exercise session, on the tasks and then write a report to be handed in by midnight.

- A Moodle quiz was used for setting the tasks and handing in the reports. The first page of the quiz (Figure 2) was an information page stating the values that are expected of the students and what they can expect from the teacher. It was adapted from [Be20, Slide 23], as a clear statement of values and expectations is said to be instrumental in fostering academic integrity. The second page was a description of the game, the game layout, the rules of engagement, a description of what to turn in, and a list of the individual tasks, starting with the simplest ones.

- The students needed to upload a PDF file with their report in order to complete the exam part. On the final page, given in Figure 3, they were requested to "sign" an academic integrity pledge. This also gave the students a simple, no-fault, opt-out of the exam, and served as a safety net in case there were technical difficulties. If there was no name in the box, the exam would not be counted. The government had already declared that not taking part in exams during Corona times would not be counted against the students, which removed quite some pressure.

Open Book Exam Statement of Values

As a student, I expect you to...

- honestly demonstrate your knowledge and abilities,
- not communicate with other persons taking the exam,
- not to offer assistance to other exam takers,
- not to use assistance from any other person during the exam,
- only submit own work at the end of the day.

As a teacher, I promise to...

- grade the exam fairly,
- give you feedback,
- be available for questions during the lab time,
- respect your privacy and not store or otherwise use the video material that could be collected during proctoring.

Fig. 2: Open-book exam statement of values

Honor statement

I verify that I have taken this exam all by myself with no outside help.
I did not have technical problems during the exam.
Please type your name in the box below if you want the exam to count.

If you want to cancel the exam with no penalty, please leave it blank or type "Cancel this exam" in the box.

Fig. 3: Honor statement at the end of the exam

4 Grading

The grading for the class was a combination of points earned during the semester and the final exam rounds. During the semester students could earn up to 30 points for completing exercises done in pair programming and submitting written reports detailing the process followed and including the program code in an appendix. Each of the exam rounds was worth 25 points. Having 105 points possible, but only 100 points be equal to 100% for the course, has cut down the squabbling about points deducted considerably over the past years.

The exam reports for each round was graded according to the following rubric:

- **2** points for formal aspects of the report, such as page numbers and spelling.
- **8** points for the quality of the process description, one point for each task.
- **14** points for the quality of the code. This includes naming conventions, comments, structure, extent of refactoring, and correctness of the code. Between one and four points were given for each task depending on the difficulty.
- **1** point for the reflection on what issues were encountered during this exam.

The two points for formal aspects were well-invested, as after many points had been deducted in the first round for violations, all students had well-formatted reports that had been spell-checked and included metadata and page numbers in the subsequent rounds.

During the semester students were required to reflect in a few sentences on what they learned each week. This helps keep the students focused on their learning. In the exam this was good feedback for me on how they were coping with the situation.

Grading was, unfortunately, much work for me. I normally only had half of the reports to grade, as we do pair programming in the labs. I had to finish grading and give feedback to everyone during the week so that they all had the same basis for starting the next round. I needed much more time for grading each submission than I normally do for grading an exam. But the feedback was found by the students to be extremely helpful, so I felt it was worth the investment of time.

Students who did not hand in Round 1 failed the class. Two persons only did exercises for part of the semester but did not take part in the exam. 39 out of 54 registered students attempted the first and the second round. Two students who had enough points to pass after these rounds did not submit the third round. An alternative, in-person exam was also offered for the second grading period for students who did not wish to take part in the experiment. Four students chose this option, two showed up and passed that exam. The others repeated the course in the next semester.

The median number of points awarded per round dropped from a 22 in Round 1 to 19 in Round 2 and 17 in Round 3. The final grades for the course showed a typical distribution: 17 with an A, 14 with a B, 7 with a C, 3 with a D, and 2 Fs.

5 Feedback and discussion

After every round the students were encouraged to give me anonymous feedback. The consensus was that the questions were fair and doable in the time given. Students were very happy that they were able to use all the materials available—their notes, videos of the lectures, the textbook, the Internet—to solve the problems. One remarked that it was so informative at the beginning of Rounds 2 & 3 to compare their solutions to mine. That gave them the self-confidence that they were on the right track.

They liked being able to go for a walk to clear their heads after a concentrated working session, and one noted that it was a relief to be able to deal with an emergency from work without panic during the exam time. One wrote that they are a slow writer and really appreciated being able to take time to make the exam good. Many felt that the stress of an exam had been lifted from them, and some even remarked that this was the most fun they have ever had taking an exam!

They noted that they really missed working in pairs. This did, however, give them the perspective that most of them do actually prefer to work in a team and not as lone programmers. Especially having to write the report alone and not split the work with a buddy was found to be problematic.

I specifically asked if they were worried about others cheating. One felt that the exam was too close to the book, so that if you had read the book and been to class you knew what you had to do. Two were worried that others might be violating the rules and getting help from others. But most were not, one even remarking that the important thing for them was to "master the challenge, not get the best grade in class".

The feedback given the students helped them to learn, even as they were doing an assessment. Of course, giving good feedback is time-consuming, so it would not be easy to do for a larger group of students unless there were many teachers available to give feedback. Instead of finding ways to assess large groups of students, it would be better for learning to lobby for smaller classes and more teaching staff.

I personally feel that this experiment has given me an authentic assessment of the learning that was done during the second Corona pandemic semester. The students actually learned a lot about programming during the exam, so I will definitely repeat the exam in non-pandemic times—but, of course, with a different room plan and different tasks.

Bibliography

[BK17] Barnes, D. J.; Kölling, M.: Objects First with JavaTM. A Practical Introduction Using BlueJ, 6th edition, Pearson, Boston, 2017.

[Be17] Bertram Gallant, T.: Academic Integrity as a Teaching & Learning Issue: From Theory to Practice. Theory Into Practice, 56(2), pp. 88–94, 2017. www.tandfonline.com/doi/full/10.1080/00405841.2017.1308173

[Be20] Bertram Gallant, T.: Going Remote *with* Integrity. International Center for Academic Integrity, 2020. https://academicintegrity.ucsd.edu/events/Going-Remote-with-Integrity1.pdf, accessed 10.04.2021.

[CDM16] Carneson, J.; Delpierre, G.; Masters, K.: Designing and Managing Multiple Choice Questions. 2nd edition, DOI 10.13140/RG.2.2.22028.31369, 2016.www.researchgate.net/publication/309263856_Designing_and_Managing_Multiple_Choice_Questions_2nd_Ed, accessed 10.04.2021.

[LC21] Lancaster, T.; Cotarlan, C.: Contract cheating by STEM students through ta file sharing website: A Covid-19 pandemic perspective. International Journal for Educational Integrity, 17(1), pp. 1–16, 2021. https://doi.org/10.1007/s40979-021-00070-0

[Mc99] McKenzie, J.: Scaffolding for success. [Electronic Version]. The Educational Technology Journal, 9(4), December 1999. http://fno.org/dec99/scaffold.html, accessed 19.08.2021.

[PB20] Patil, A.; Bromwich, J. E.: How It Feels When Software Watches You Take Tests. [Electronic Version]. New York Times, Sept. 29, 2020. www.nytimes.com/2020/09/29/style/testing-schools-proctorio.html, accessed 19.08.21

[WW09] Williams, J. B.; Wong, A.: The efficacy of final examinations: A comparative study of closed-book, invigilated exams and open-book, open-web exams. British Journal of Educational Technology, 40(2), pp. 227–236, 2009. https://doi.org/10.1111/j.1467-8535.2008.00929.x

Towards Criteria for Valuable Automatic Feedback in Large Programming Classes

Dominic Lohr[1], Marc Berges[2]

Abstract:
Automatically generated feedback systems and assessments are popular tools to deal with the rapidly increasing number of students, especially in STEM-related subjects such as Computer Science. However, in the development of such systems, the focus is often on compensating resource issues only, and the quality of the systems' feedback is not considered. To determine what kind of feedback participants of large introductory programming courses (CS1) use and what expectations they have regarding effective feedback on programming difficulties, a survey was conducted in a large introductory programming class (300 students). The results show that during online semester, students made less use of possibility to ask tutors questions in lab sessions. This is in line with statements of experienced tutors who were previously interviewed. Moreover, the results show that it is not sufficient to merely point to errors in program code but that students need more detailed information concerning the underlying cause of the error, especially in automated systems.

Keywords:
Automatic feedback; CS1; programming education

1 Introduction

Although the number of computer science students increases, there is still a dramatic need for well-qualified computer scientists. For example, in Germany, the number of CS graduates has doubled in the last decade [St20]. Moreover, there is also unprecedented growth in computer science students at universities worldwide [Na18]. In all STEM-related subjects and other 'non-technical' fields, acquiring essential programming competencies has become mandatory. Thus, it is not surprising that at many educational institutions resources for providing students with the best possible support during the introduction to programming skills are scarce.

To make individual support possible despite the rapidly increasing number of students, specific resources are needed, such as peer tutors, who positively impact students' academic performance [CKK82]. However, it is often difficult to find enough tutors for the large number of students and it has not become easier with special situations like distance learning during the pandemic last year. Even worse, to keep the learning effects high, groups in student practice courses have to be kept as small and homogeneous as possible to provide students with individual support [Pi12].

These lab sessions, supervised by peer tutors, are an important opportunity for students to get good feedback on their programming effort. A lecturer cannot address all individual misconceptions during their lecture or provide individual feedback to each student. Each student has an individual regarding educational biography [Kn08] and level of competency, especially at the beginning of a computer science program. Moreover, it is evident that personalized feedback is essential when learning to program, which is an ability that is hard-to-learn [QL17]. To address the difficulties and cope with misconceptions, it is desirable to provide students with an additional opportunity to receive feedback tailored to their needs by using education technology (e.g., intelligent tutoring system).

According to Jacobs, the learning effectiveness of exercise tasks is mainly influenced by two factors: active task processing on the one hand and feedback following their processing on the other [Ja02]. But what kind of feedback do CS1 students use, and what

1 Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik dominic.lohr@fau.de

2 Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik marc.berges@fau.de

expectations towards good feedback on their programming difficulties do they have? To answer these questions, the following paragraphs present a short survey of students in a large introductory programming class.

2 Theoretical Background on Feedback

There is a large number of definitions of the term feedback in an educational setting. According to Boud and Molloy, feedback is a "process whereby learners obtain information about theirwork in order to appreciate the similarities and differences between the appropriate standards for any given work, and the qualities of the work itself, in order to generate improved work" [BM13]. Whereas, John Hattie defines "the main purpose of feedback as to reduce discrepancies between current understandings and performance and a goal" [HT07] and even postulates that feedback is the most substantial influencing factor to improve learning performance [Ha15]. To reduce the before-mentioned discrepancies, Hattie et al. stress, among others, that teachers should "create a learning environment in which students develop self-regulation and error detection skills" [HBP96].

Draper [Dr05] identifies three general sources of feedback: Feedback from the learner (**internal feedback**), feedback from the learner's environment, and feedback given from the teacher (**external feedback**). Only the last of the mentioned sources represents a possibility to provide automated feedback.

Furthermore, a distinction is made between **simple feedback**, in which only the solution for a task is presented, and **elaborated feedback**, which includes, for example, differentiated instructions for self-correction. Thus, the latter goes far beyond providing a sample solution and may require even more resources than the development of the actual task [Ba17]. A detailed description of different feedback types can be found in [KJH19]. In 1996, Kluger and DeNisi's meta-analysis presented different types of feedback and examined their effectiveness [KD98]. Three years later, John Hattie demonstrated in his detailed synthesis of over 70 meta-analyses that computer-assisted instructional feedback is one of the most effective forms of feedback [Ha99].

3 Feedback in CS

In 2007, Hattie and Timberley published a feedback model that establishes a framework for the different levels and stages of the feedback process [HT07]. Ott et al. [ORS16] investigated the extent to which this feedback model can be applied to the CS1 context and concluded that automated feedback plays a role at the first three levels: Task level (1), Process level (2) and Self-regulation level (3). The Self level (4) of Hatties' and Timberleys' levels of feedback was not considered, because according to the authors, it does not have a significant effect to improve the learning performance. They developed a roadmap of effective feedback practices for the different levels and stages based on Hatties' and Kimberleys' feedback model. "In their roadmap, automated assessment of exams is placed at the task level, student support through adaptive feedback from automated tools at the process level, and tutoring systems and automated assessment as options for self-assessment of students at the self-regulation level." [KJH19]

Automated feedback and assessment systems, as many other authors also call them, have been around since the 1960s [Ho60] and the number has increased in recent years. Keuning et al. conducted an extensive systematic literature review on automatic feedback generation for programming tasks, examining 101 tools they classified into feedback content categories based on Narciss [Na01]. They examined the tools in terms of type, the technique used for the generation, the possibility of adaption by the teacher, and the quality and effectiveness. The researchers concluded that most of the systems are limited to finding errors but do not offer any help for eliminating the problems found or point to the next steps necessary to do so [KJH19].

4 Methodology

To evaluate what students' expectations regarding feedback on programming difficulties look like, we designed a short survey. The survey was conducted online in a large first-year introductory programming course (CS1) in which the object-oriented programming language Java is used. The course is usually held face-to-face but was online due to the pandemic situation. About 500 students are enrolled in the course, most of them first-year students of Computer Science, Business Informatics or International Information Systems. The students have different programming skills, as they come from different educational backgrounds.

As part of the lecture, students have to complete programming assignments each week, which are automatically corrected. To complete the assignments, students have access to public unit tests to test the functionality of their code. For correction, the submitted codes are additionally tested with secret tests, and finally, points are assigned based on the combination of public tests and secret tests.

To conduct the survey, all students enrolled in the course received an invitation link to the questionnaire by email, which was sent about two weeks after the lecture period. The survey was offered in German and English and was available for one week. Participation was voluntary and completely anonymous. At the time of the survey, approximately 300 students were still enrolled in the course. 93 students answered the survey, which is a reasonably good response rate for online surveys [Nu08].

The first question in the survey focused on the sources for feedback students used during the pandemic situation.

Q1: How often have you used the following platforms/resources to get feedback on your programming tasks?

The provided platforms and resources were collected beforehand in interviews with experienced peer tutors of the course. In addition to the suggested options, students could name other feedback options in a text field. Items had to be answered on a 5-point Likert scale regarding the extent of use (never (1), almost never (2), sometimes (3), almost every time (4), every time(5)).

The second question yielded the expectations towards valuable feedback. Here, open-ended questions were provided.

Q2: What are your expectations towards valuable feedback on programming difficulties?

The answers of Q2 were analyzed using qualitative text analysis based on the methodology of Mayring [Ma14].

Limitations

The survey was conducted at the end of the semester. Therefore many students who dropped out may not have participated in the survey, so the number of unreported cases could be significantly higher. Furthermore, it is questionable how much the survey results were influenced by the current unique situation of distance learning.

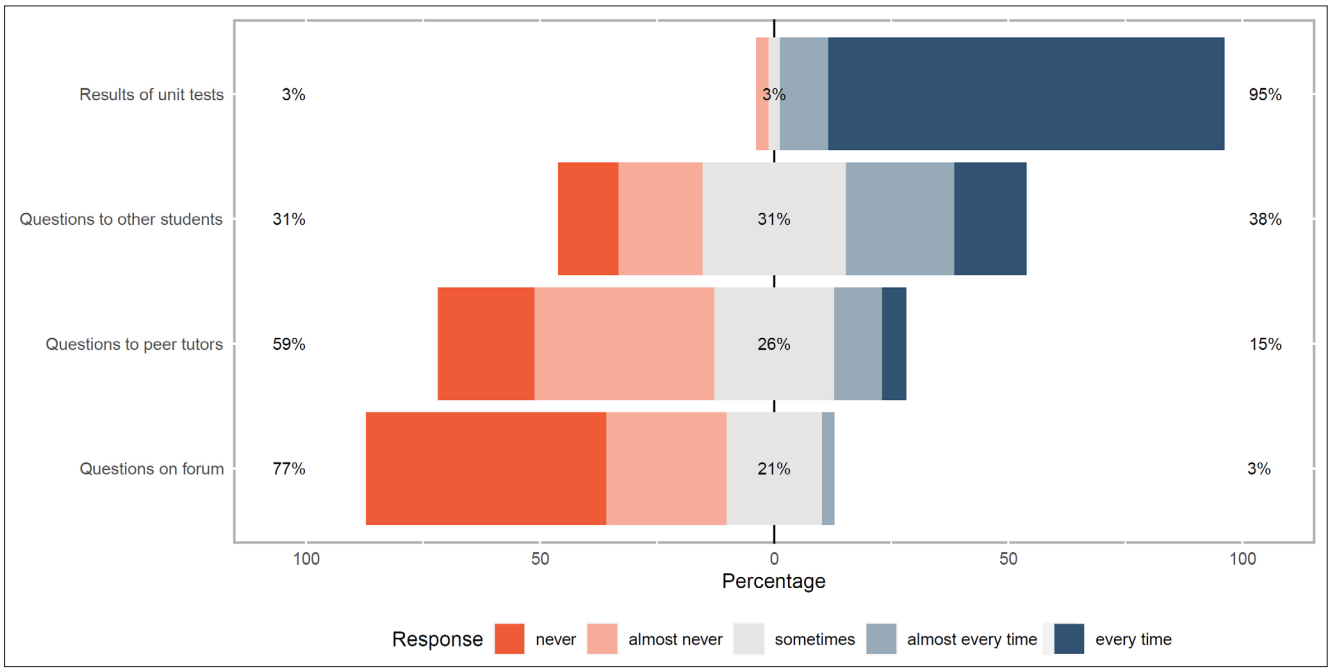

Fig. 1: Frequency of used feedback resources

5 Results

5.1 How do students get feedback on programming tasks?

The most frequently used form of feedback is the results of the given Unit-Tests. Almost 95% of students indicated that they always (81%) or almost always (14%) used the outcomes of the unit tests, which were provided to them along with the programming tasks (see Figure 1).

20% of the students indicated an additional form of feedback in the 'Others' field. After analyzing the answers, we divided them into four categories, shown in Table 1.

Tab. 1: Additional resources for feedback

categories	
internet research	websites like StackOverflow, internet forums, Tutorials
customized Unit-Tests	outcomes of Unit-Tests written by students or fellow students
expert interviews	expert-feedback from the students' environment
Discord-Server	created by students to share e.g. solutions

The most common answers under 'Others' were websites or internet forums. *StackOverflow.com* and the Google search engine were explicitly mentioned. All these answers were grouped under the term **internet research**. Some participants also mentioned that they used feedback from the results of additional JUnit tests written by themselves or fellow students. There also seems to be a so-called 'Discord-Server', which was created and hosted by students for students explicitly for the lecture. Discord is a social platform which includes, among other things, group chats. From the survey results, it appears that they used this platform to discuss course content, share sample solutions, and get feedback from fellow students.

5.2 What expectations do students have concerning valuable feedback?

The qualitative analysis of the responses to Q2 revealed two broad categories, presented below with sample citations from the students.

Feedback does not only indicate where I made an error but supports me to help myself
Many students indicate that it took them a very long time (sometimes more than 20 minutes) to understand the code or the results of the test cases. They found it essential that tests indicate whether the program code works or not and help them find out where the error is located, what caused it, and give feedback on what can be done to fix it. They emphasize that it is not enough to know what values were expected in the test cases to determine what exactly was wrong with the code.

The following quotes from students exemplify the results:

> *"The feedback should give me a hint in the right direction to come up with the problem [in the code] myself without spending hours stuck on it."*

> *"More feedback, or any feedback at all, on the errors committed (their cause and correction) because the description of the tests was often not sufficient to find the error in a reasonable time."*

> *"It shows if the code is correct and helps you a bit to find the next steps or the problem."*

Good feedback is not only providing solutions
The students emphasize that it can sometimes be helpful to get a sample solution for the given tasks. However, they say a solution alone is not enough to precisely understand how the program code works or find an error.

Again, the results are exemplified by the following quotes from students:

> *"A complete solution to the programming tasks sometimes would have been very helpful for me to be able to understand how one had to proceed at all."*

"[Valuable Feedback] doesn't give me a solution but puts me on the right track. It should also clear up any syntactical ambiguities."

6 Discussion and Conclusion

First, results indicate a low level of usage of peer tutors as a resource for feedback. Here, a possible reason could be the partially changed format due to the pandemic situation. In the past, the students were supervised by a tutor in a computer lab and were able to ask questions face-to-face if needed. Recently, there was one practice class for the entire group, so students may not have felt comfortable asking questions, which is indicated by a sample quote from one of the participants: *"I tried to find tutorials on similar tasks [...] I felt uncomfortable asking questions in the group."* To develop automatic feedback systems for programming education, results implicate a need for an engaging environment to encourage students to ask questions. Additionally, in future research, we would like to determine whether the phenomenon described is solely due to distance learning or what different reasons exist. It would be desirable to know how pracitcal courses can be designed in the digital format so that students feel more comfortable asking questions.

Some of the answers to feedback sources listed under 'Others' are ambiguous regarding their use. For example, in the case of 'Internet research', it must be differentiated whether the students merely acquired knowledge from websites or specifically searched for compilation errors to obtain better feedback on the tasks and find out about the causes of the error.

Similarly, the predominant feedback resource was the results of the unit tests. However, considering the answers to Q2, the reason was not the quality of the feedback but the fact that these tests are the basis for scoring the tasks. A large proportion of students indicated in their Q2 responses that this form of feedback was not sufficient to identify the causes of their errors. Again, results show that understanding the cause of an error is an essential issue for valuable feedback.

The failure of the unit tests as a valuable feedback resource is, on the one hand, that CS1-students do not yet have the necessary experience in dealing with unit tests or do not understand the source code of the given tests. Here, further competencies in testing could improve the situation. The implication for automatic feedback is the demand to include the students' previous knowledge in the system. Feedback can only be valuable if it is understood.

However, even if students understand how to use the tests and their results, other problems remain. The survey results show that even if the test results point out where the error occurred, students still have difficulty deducing the cause of the error. Moreover, they often do not have the necessary skills/competencies to infer from the symptom to the cause of the error, a process that is often challenging even for experienced tutors. Even Keuning et al. conclude that "only giving test-based feedback will not in all cases help a student to fix an incorrect program" and "to really help a student, just pointing at an error may not help" [KJH19] which is in line with our results. So, automatic feedback must focus on the roots of the errors rather than on the apparent symptoms.

Existing feedback systems are mainly limited to identifying errors. Thus, they represent a form of simple feedback only. However, the results of this work are in line with results from other research, regarding the fact that it is not sufficient for novice programmers to know at which point in the program code an error occurred. They need detailed information on the underlying causes of the error. So, future efforts will therefore focus, among other things, what possibilities exist to infer possible causes of the errors from given symptoms (e.g., results of unit-tests or static code analysis) of a program code. Ultimately, the main goal is to provide students with automated, personalized, elaborated feedback tailored to the individual student that provides novice programmers clues about the possible causes of incorrect program code and what steps can be taken to fix it. With all this, students' knowledge and abilities are elaborated.

Bibliography

[Ba17] Bayerlein, Oliver: Lernerbeobachtungen zur Nutzung von Feedback bei einem videogestützten Online-Sprachkurs für Deutsch als Fremdsprache. Informationen Deutsch als Fremdsprache, 37(6):570–576, 2017.

[BM13] Boud, David; Moloy, Eizabeth, eds. Feedback in higher and professional education: Understanding it and doing it well. Routledge, London, 2013.

[CKK82] Cohen, Peter A.; Kulik, James A.; Kulik, Chen-Lin C.: Educational Outcomes of Tutoring: A Meta-analysis of Findings. American Educational Research Journal, 19(2):237–248, 1982.

[Dr05] Draper, Steve: Feedback: A Technical Memo. www.psy.gla.ac.uk/~steve/feedback.html, 2005. Visited 2021-07-28.

[Ha99] Hattie, John: Influences on Student Learning. Inaugural lecture, University of Auckland, Auckland, New Zealand, 1999.

[Ha15] Hattie, John: Lernen sichtbar machen. Schneider Verlag Hohengehren, Baltmannsweiler, 2015.

[HBP96] Hattie, John; Biggs, John; Purdie, Nola: Effects of Learning Skills Interventions on Student Learning: A Meta-Analysis. Review of Educational Research, 66(2):99–136, 1996.

[Ho60] Hollingsworth, Jack: Automatic graders for programming classes. Communications of the ACM, 3(10):528–529, 1960.

[HT07] Hattie, John; Timperley, Helen: The Power of Feedback. Review of Educational Research, 77(1):81–112, 2007.

[Ja02] Jacobs, Bernhard: Aufgaben stellen und Feedback geben. http://psydok.psycharchives.de/jspui/handle/20.500.11780/1024, 2002. Visited 2021-07-24.

[KD98] Kluger, Avraham N.; DeNisi, Angelo: Feedback Interventions: Toward the Understanding of a Double-Edged Sword. Current Directions in Psychological Science, 7(3):67–72, 1998.

[KJH19] Keuning, Hieke; Jeuring, Johan; Heeren, Bastiaan: A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education, 19(1):1–43, 2019.

[Kn08] Knobelsdorf, Maria: A typology of CS students' preconditions for learning. In: Proceedings of the 8th International Conference on Computing Education Research. ACM, New York, NY and USA, pp. 62–71, 2008.

[Ma14] Mayring, Philipp: Qualitative content analysis: theoretical foundation, basic procedures and software solution. http://nbn-resolving.de/urn:nbn:de:0168-ssoar-395173, 2014. Visited 2021-07-28.

[Na01] Narciss, Susanne: Feedback Strategies for Interactive Learning Tasks. In (Driscoll, Marcy P.; Jonassen, David H.; Harris, Phillip, eds): Handbook of Research for Educational Communications and Technology, AECT Series, 2, pp. 125–143. Taylor & Francis, Hoboken, 2001.

[Na18] National Academies of Sciences, Engineering, and Medicine: Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments. National Academies Press, Washington, D.C., 2018.

[Nu08] Nulty, Duncan D.: The adequacy of response rates to online and paper surveys: What can be done? Assessment & Evaluation in Higher Education, 33(3):301–314, 2008.

[ORS16] Ott, Claudia; Robins, Anthony; Shephard, Kerry: Translating Principles of Effective Feedback for Students into the CS1 Context. ACM Transactions on Computing Education, 16(1):1–27, 2016.

[Pi12] Pinto, Yma: The efficacy of homogeneous groups in enhancing individual learning. Journal of Education and Practice, 3(3):25–38, 2012.

[QL17] Qian, Yizhou; Lehman, James: Students' Misconceptions and Other Difficulties in Introductory Programming. ACM Transactions on Computing Education, 18(1):1–24, 2017.

[St20] Statista: Anzahl der Studierenden im Fach Informatik in Deutschland nach Geschlecht in den Wintersemestern von 1989/1999 bis 2019/2020, 2020.

9. Fachtagung Hochschuldidaktik Informatik (HDI) 2021

15. – 16. September 2021 in Dortmund

Vorabdruck der Konferenzbeiträge

Impressum

1. Auflage 2021, 60 Exemplare

Satz:
Jan Hillers, FernUniversität in Hagen,
Dez. 5.2.3 – Grafik

Herstellung:
FernUniversität in Hagen, Dez. 5.2.2 – Druckerei

Herausgeber:
- Jörg Desel, Simone Opel
  Fak. MI - LG Softwaretech. u Theo. Progr,
  FernUniversität in Hagen;
- Juliane Siegeris
  HTW Berlin

ISBN:
978-3-00-070267-9

9. Fachtagung Hochschuldidaktik Informatik (HDI) 2021

Auf der Fachtagung HDI werden alle Aspekte der informatischen Bildung im Hochschulbereich behandelt. Dazu gehören Themenbereiche wie Lehrinnovationen, Curriculumsentwicklung, fachdidaktische Konzepte oder der Umgang mit Diversität, die jeweils bezogen auf Informatikstudiengänge an Universitäten und Hochschulen, verwandten Studiengängen oder der Informatiklehre in anderen Studiengängen erforscht und diskutiert werden.

Herausgeber:

- Jörg Desel, Simone Opel
  Fak. MI - LG Softwaretech. u Theo. Progr,
  FernUniversität in Hagen;
- Juliane Siegeris
  HTW Berlin

Tagungsband

FernUniversität in Hagen

Universitätsstraße 47
58097 Hagen

www.fernuni-hagen.de

Titelfoto:
Tempura/E+/GettyImages

ISBN: 978-3-00-070267-9

FernUniversität in Hagen

STIFTERVERBAND
Bildung. Wissenschaft. Innovation.