

GESELLSCHAFT
FÜR INFORMATIK



Veronika Thurner, Barne Kleinen,
Juliane Siegeris, Debora Weber-Wulff
(Hrsg.)

Software Engineering im Unterricht der Hochschulen

**24.–25. Februar 2022
Berlin und virtuell**

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings

Series of the Gesellschaft für Informatik (GI)

Volume P-321

ISBN 978-3-88579-715-9

ISSN 1617-5468

Volume Editors

Veronika Thurner, veronika.thurner@hm.edu

Hochschule München Lothstraße 64, 80335 München, Deutschland

Barne Kleinen, Juliane Siegeris, Debora Weber-Wulff

Hochschule für Technik und Wirtschaft Berlin, Wilhelminenhofstraße 75a, 12459 Berlin

Series Editorial Board

Andreas Oberweis, KIT Karlsruhe,

(Chairman, andreas.oberweis@kit.edu)

Torsten Brinda, Universität Duisburg-Essen, Germany

Dieter Fellner, Technische Universität Darmstadt, Germany

Ulrich Flegel, Infineon, Germany

Ulrich Frank, Universität Duisburg-Essen, Germany

Michael Goedicke, Universität Duisburg-Essen, Germany

Ralf Hofestädt, Universität Bielefeld, Germany

Wolfgang Karl, KIT Karlsruhe, Germany

Michael Koch, Universität der Bundeswehr München, Germany

Peter Sanders, Karlsruher Institut für Technologie (KIT), Germany

Andreas Thor, HFT Leipzig, Germany

Ingo Timm, Universität Trier, Germany

Karin Vosseberg, Hochschule Bremerhaven, Germany

Maria Wimmer, Universität Koblenz-Landau, Germany

Dissertations

Steffen Hölldobler, Technische Universität Dresden, Germany

Thematics

Agnes Koschmider, Universität Kiel, Germany

Seminars

Judith Michael, RWTH Aachen, Germany

© Gesellschaft für Informatik, Bonn 2022

printed by Köllen Druck+Verlag GmbH, Bonn



This book is licensed under a Creative Commons BY-SA 4.0 licence.

Vorwort

Dieser Tagungsband begleitet den 18. Workshop der Reihe „SEUH – Software Engineering im Unterricht der Hochschulen“, der im Februar 2022 in Berlin sowie virtuell stattfindet, parallel zur virtuell durchgeführten Tagung „Software Engineering“.

Tradition der SEUH

Die SEUH ist seit vielen Jahren das Forum im deutschsprachigen Raum, auf dem Lehrende aus Universitäten, Hochschulen für angewandte Wissenschaften sowie dualen Hochschulen ihre Erfolge, Misserfolge und Erfahrungen in der Software Engineering Ausbildung vorstellen, diskutieren und gemeinsam die Qualität der Lehre verbessern. Neben den inhaltlichen Impulsen rund um Lehren und Lernen bietet die SEUH viel Raum für Diskussion und den gegenseitigen Austausch. Viele Lehrende haben von der SEUH entscheidende Impulse für ihre Arbeit erhalten.

Thematische Schwerpunkte der SEUH 2022

Die SEUH 2022 wird durchgeführt an der HTW in Berlin und als virtuelles Format, zeitlich abgestimmt mit der Tagung Software Engineering (SE) der Gesellschaft für Informatik, die rein virtuell stattfindet.

Im Fokus der SEUH 2022 stehen die beiden Themenbereiche Nachhaltigkeit und Ethik in der Software-Engineering-Ausbildung sowie Lehren und Lernen nach der Pandemie.

Der Tradition folgend, schafft auch die diesjährige SEUH einen Rahmen für die Diskussion zu aktuellen Fragen der Softwaretechnik im Kontext der Hochschullehre. Marina Köhn von der Beratungsstelle Green-IT des Umweltbundesamtes eröffnet die SEUH 2022 mit einer Keynote zum Thema „Green Coding = Intelligent Coding: Softwarecode ohne Verschwendung von Energie- und Ressourcen“ und greift damit den Themenbereich der Nachhaltigkeit auf. Jutta Weimar, Gründerin und Inhaberin der Facilitation Academy, adressiert am zweiten Tag der Konferenz in ihrer Keynote das Thema „Neue Ideen für virtuelle Lehre“ und moderiert anschließend das Bar Camp, als unmittelbares Anwendungsbeispiel virtueller Zusammenarbeit im Bildungskontext.

Des Weiteren umfasst die SEUH 2022 Beiträge im Vortragsformat mit Diskussion. Hierfür hat das Programmkomitee aus 21 Einreichungen 12 Beiträge ausgewählt, die sich folgenden Themenschwerpunkten zuordnen lassen:

- Software Engineering erlernen
- Software Engineering lehren
- Weiterführende Themen

Die SEUH lebt vom intensiven kollegialen Austausch. Um diesen zu verstärken wird dieses Jahr erstmals neben den üblichen Vortragsformaten auch ein Bar Camp integriert, in dessen Rahmen aktuell für die Teilnehmer:innen relevante Themen gemeinschaftlich bearbeitet werden.

Dank

Eine Tagung wie die SEUH ist nicht denkbar ohne die Menschen, die die Wichtigkeit von Diskussion und Dialog zwischen Gleichgesinnten erkannt haben und dies durch ihre Unterstützung zum Ausdruck bringen. Ihnen wollen wir an dieser Stelle danken.

Zu allererst bedanken wir uns bei den Fachkolleginnen und Fachkollegen, die Beiträge eingereicht und damit eine Auswahl ermöglicht haben. Dem Programmkomitee danken wir für die Begutachtung der Beiträge, deren Einordnung in das Programm sowie das hilfreiche Feedback für die Verbesserung der Beiträge.

Barne Kleinen, Juliane Siegeris und Debora Weber-Wulff danken wir fürs Übernehmen der lokalen Organisation, die professionelle Gestaltung der Webseite, das Aufsetzen der Technik für die virtuelle Durchführung, die Unterstützung bei der Planung im Vorfeld und die hervorragende Organisation vor Ort.

Des Weiteren bedanken wir uns bei den studentischen Hilfskräften der HTW Berlin für ihre Unterstützung vor Ort bei der Durchführung der SEUH 2022.

Ebenso danken wir der Hochschulleitung der HTW Berlin für die Möglichkeit, die Räumlichkeiten und technischen Einrichtungen der Hochschule für die SEUH zu nutzen.

Ein herzlicher Dank gebührt ferner den Sponsoren, ohne deren Unterstützung die Durchführung der SEUH in dieser Form nicht möglich wäre:

- adesso SE
- Hyland Software Germany GmbH
- LIVEPERSON
- Novatec Consulting GmbH
- thoughtworks

Den Mitarbeiterinnen und Mitarbeitern der Gesellschaft für Informatik danken wir für die elegante und zeitgemäße Möglichkeit, diesen Tagungsband im Rahmen der LNI-Reihe in elektronischer Form zu veröffentlichen.

Ein weiteres, großes Dankeschön geht an Benedikt Zönnchen von der Hochschule München, der ebenso fingerfertig wie sorgfältig den Tagungsband zusammengestellt hat.

Nicht zuletzt danken wir allen Interessierten, die an der SEUH 2022 teilgenommen und sich in den Diskurs eingebracht haben.

Wir freuen uns, mit diesem Tagungsband einen Einblick in die SEUH 2022 geben zu können, und wünschen anregendes und inspirierendes Lesen.

Veronika Thurner, Hochschule München
Vorsitzende des Programmkomitees

München, im Februar 2022

Sponsoring

Wir danken den folgenden Unternehmen und Institutionen für die Unterstützung der Konferenz.

The logo for adesso, featuring the word "adesso" in a bold, blue, sans-serif font, followed by a thin, brown, stylized vertical line.The logo for Hyland, featuring the word "Hyland" in a white, serif font, set against a green-to-blue gradient background.The logo for LIVEPERSON, featuring an orange gear icon to the left of the word "LIVEPERSON" in a bold, orange, sans-serif font.The logo for NOVATEC, featuring a black, stylized infinity symbol to the left of the word "NOVATEC" in a bold, black, sans-serif font.The logo for thoughtworks, featuring a red, stylized slash icon to the left of the word "thoughtworks" in a bold, dark blue, sans-serif font.

Tagungsleitung

Leitung des Programmkomitees	Veronika Thurner, Hochschule München
Lokale Organisation	Barne Kleinen, HTW Berlin
	Juliane Siegeris, HTW Berlin
	Debora Weber-Wulff, HTW Berlin

Programmkomitee

Steffen Becker	Universität Stuttgart
Ilona Buchem	BEUTH Hochschule für Technik Berlin
Martin Fränzle	Universität Oldenburg
Michael Goedicke	Universität Duisburg
Barne Kleinen	HTW Berlin
Markus Müller-Olm	Universität Münster
Barbara Paech	Universität Heidelberg
Lutz Prechelt	Freie Universität Berlin
Eva-Maria Schön	HAW Hamburg
Juliane Siegeris	HTW Berlin
Veronika Thurner	Hochschule München
Karin Vosseberg	Hochschule Bremerhaven
Debora Weber-Wulff	HTW Berlin
Heike Wiesener	HWR Berlin

Ständiges Organisationskomitee

Steffen Becker	Universität Stuttgart
Axel Schmolitzky	HAW Hamburg
Veronika Thurner	Hochschule München

Inhaltsverzeichnis

Eingeladene Vorträge

Marina Köhn <i>Green Coding = Intelligent Coding: Softwarecode ohne Verschwendung von Energie- und Ressourcen</i>	15
Jutta Weimar <i>Neue Ideen für virtuelle Lehre</i>	17

Wissenschaftliches Programm

Software Engineering erlernen

Jan H. Boockmann, Kerstin Jacob, Gerald Lüttgen <i>Throw Away Student Software At Semester End? Better Not!</i>	23
Stefan Bente, Jann Intveen, Fabian Krampe <i>Divekit — Digitalisierung und Individualisierung als Schlüssel für eine moderne Softwaretechnik-Ausbildung</i>	29
Sandro Speth, Steffen Becker, Uwe Breitenbücher, Philipp Fuchs, Niklas Meißner, Anna Riesch, Daniel Wetzel <i>IT-REX — A Vision for a Gamified e-Learning Platform for the First Semesters of Computer Science Courses</i>	43

Software Engineering lehren (Teil 1)

Axel Böttcher, Daniela Zehetmeier <i>Vorbereitung auf das wahre Leben — Herausforderungen bei Diskurs und Umgang mit Unsicherheit in der Lehre</i>	51
--	----

Jens Liebehenschel, Martin Simon

<i>Online Teamteaching als Inverted Classroom — Erfahrungen aus den Pandemie-Semestern</i>	63
--	----

Karsten Weicker

<i>Individuelle Benotung von Teamprojekten: Lessons Learnt</i>	75
--	----

Software Engineering lehren (Teil 2)

Sandro Speth, Niklas Krieger, Georg Reißner, Steffen Becker

<i>Teaching during the Covid-19 Pandemic — Online Programming Education</i>	89
---	----

Christin Voigt, Nicole Draxler-Weber, Uwe Hoppe

<i>Virtuelle Präsenz? — Fallbeispiel eines Flipped Classrooms während der Covid-19-Pandemie</i>	95
---	----

Veronika, Thurner, Axel Böttcher

<i>Auf dem Weg zum neuen Normal — Herausforderungen für die Lehre im post-pandemischen Zeitalter</i>	107
--	-----

Ergänzende Themen

Steffen Dick, Stefan Schulz, Christoph Bockisch

<i>A study on the quality mindedness of students</i>	119
--	-----

Jonas Kötter, Agnes Mainka, Natallia Kukhareuka

<i>Emotionale Veränderung von Studierenden bedingt durch die digitale Transformation der Lehr- und Lernformate</i>	125
--	-----

David Zellhöfer

<i>Methodenvermittlung des Software Engineerings als Hebel für die Digitale Transformation der Öffentlichen Verwaltung</i>	137
--	-----

Eingeladene Vorträge

Green Coding = Intelligent Coding: Softwarecode ohne Verschwendung von Energie- und Ressourcen

Marina Köhn

Abstract: Bei der Bewertung der Umweltauswirkungen von Informations- und Kommunikationstechnologien liegt der Fokus hauptsächlich auf die Optimierung von Hardware. Obwohl die Software einen großen Einfluss auf die Energie- und Hardwareeffizienz hat. Das Design und die Architektur der Software bestimmen, wie viel an Hardwareressourcen und elektrischer Energie notwendig ist, um die Software auszuführen. Software kann sparsam oder verschwenderisch mit den Hardwareressourcen umgehen. Je nachdem, wie intelligent sie programmiert wird, benötigt sie beispielsweise weniger oder mehr Prozessorleistung und erzeugt weniger oder mehr Daten die übertragen und gespeichert werden muss. Software kann auch bewirken, dass Hardware vorzeitig obsolet wird. Vor diesem Hintergrund ist es notwendig, dass zukünftige Softwareentwickler*innen die Anforderungen und Methoden kennen, die eine umweltverträgliche Software voraussetzt.

Neue Ideen für virtuelle Lehre

Jutta Weimar

Abstract: „Wie kann ich Kontakt, Fokus und Verbindlichkeit in der virtuellen Lehre herstellen und halten?“ ist eine Kernfrage, die viele Lehrende derzeit intensiv beschäftigt. Die Referentin teilt dazu Impulse aus ihrer Praxis in der Facilitation Academy, www.facilitation-academy.de, und lädt die Teilnehmenden ein zur erlebten Selbsterfahrung und anschließenden Reflexion.

Wissenschaftliches Programm

Software Engineering erlernen

Throw Away Student Software At Semester End? Better Not!

Jan H. Boockmann¹, Kerstin Jacob¹, Gerald Lüttgen¹

Abstract: Software engineering is continuously evolving in order to meet the challenges faced by the ever growing complexity and longevity of digital systems. Students should thus acquire additional practical competencies wrt. software evolution and maintenance. This paper describes how student team projects at our Chair have been redesigned to meet this need, and reports on our first experiences. It also aims at initiating a discussion on the challenges we met, namely the contributions we can expect from students, the effort required from staff, and the individualization of grades.

Keywords: long-living software; project-based learning; software evolution and maintenance

1 Introduction

Today's, and even more so, tomorrow's computer scientists need sound skills in designing and implementing systems in such way that these are maintainable and can evolve over a long period of time. Especially important are competencies in reading and understanding software artifacts written by others, e.g., frameworks, source code, and documentation. As pointed out by Spinellis [Sp03], these competencies are increasingly relevant nowadays, because software products grow larger and become more complex, are developed collaboratively in large teams, and combine a variety of different technologies.

The problem of how to achieve long-living, maintainable systems is called a “grand challenge within software engineering” by Bennett; Rajlich [BR00] and remains an ongoing research topic, see, e.g., the DFG priority programme report by Goltz et al. [Go15]. As noted by Tate [Ta05], reconciling agile practices, which focus on feature-based development, and the goal of maintainable software is especially challenging and subject to ongoing research.

In the light of these circumstances, we observed that the topic of designing and implementing long-living software products had played only a subordinate role in the teaching at our Chair. Our student projects on software development spanned a single semester, development started from scratch, and the generated software was thrown away at semester end. Thus, students (and assessment) focussed on having software that works at the time of hand-in, and less so on writing maintainable code. Given the aim of teaching students the abilities to understand complex systems and write long-living software, we revised the teaching concept of our student projects two years ago. In our new project setting, students work in flexible teams to develop new projects and, even more importantly, to maintain and extend existing projects that have been initiated in previous semesters.

¹ University of Bamberg, Software Technologies Research Group, Germany, firstname.lastname@swt-bamberg.de

The remainder of this paper is structured as follows: Sect. 2 briefly introduces our new teaching concept for software development projects, Sect. 3 summarizes the lessons learned by us, and Sect. 4 proposes questions regarding our teaching concept which might deserve a wider discussion within the community of lecturers in software engineering.

2 Revised Teaching Concept

At our Chair, the teaching of software engineering principles at Bachelor level had spanned two compulsory modules. Usually, Bachelor students in their third semester of study will take a 6 ECTS module called “FSE” that introduces the foundations of software engineering, focussing on requirements engineering, software modelling and design, and quality assurance. The lectures are accompanied by traditional practicals as well as tutorials that introduce students to popular development tools such as *JUnit* and *Cucumber* for unit and acceptance testing, resp. In the following semester, students continue with a 6 ECTS “LAB” project module in which they apply the knowledge gained in FSE to develop in teams a small Java application following a variant of *SCRUM*. The module focusses on the application of agile practices and team work. Students also familiarize themselves with the concepts of object-relational mappers and the *JavaFX* GUI framework. The project tasks are provided by industry partners who function as a customer for the student teams.

However, teaching students the need for documentation, including abstractions such as UML diagrams, is a challenging task, because the simple systems typically discussed in lectures largely do not require documentation for understanding; unfortunately, presenting a complex system, where documentation is necessary, exceeds a semester’s time frame. In many classical team project modules, this problem persists to some degree, because students rarely have to use documentation written by others. Another issue is that students are closely guided by lecturers regarding agile processes, which is necessary to successfully conduct a first team project, but actually contradicts agile principles.

New concept. We now continue the LAB with a second, *elective* team project module (again 6 ECTS), called the “HUB”, where students operate in flexible teams to start new projects from scratch and maintain software products developed in previous semesters. Also taught in collaboration with industry partners, HUB puts a stronger focus on programming practices and develops web-based products based on the *Spring Boot* and *Angular* frameworks. Note that this allows students to fully reuse the business logic implemented in the LAB for a continuation in the HUB module. With the addition of HUB, the technology stack of LAB was revised to also include *Spring Boot*, albeit not in a client-server setting. This turned out to be a good decision, because students now have to use the architecture and design patterns stipulated by the framework, and thus avoid typical mistakes and save valuable time.

Projects currently running in the HUB include, for example, a build monitor that calculates and visualizes software metrics obtained during automatic software builds, and a library

system for administrating the Chair's literature collection ("Handapparat"). Students are required to work on two or three projects at a time, which are however all based on the same technology stack sketched above. For each project, one student acts as the product owner, while the other students typically specialize on one or two technologies.

We believe that the HUB is suitable for teaching the competencies required to develop long-living software. The module trains the ability to read and comprehend complex foreign source code, as students need to become familiar with the software products built in previous semesters to be able to fix bugs or design flaws, to replace libraries and frameworks by current versions, and to implement new features. By having to deal with the sometimes messy source code and frequently incomplete and inconsistent documentation of software products of prior student generations, students experience first-hand the value of clean code and the importance of thorough documentation. This motivates them to strive for high-quality artifacts and to apply, e.g., modern code review.

Successes and new challenges. The combination of the FSE module and the LAB project has been running for approximately ten years at our Chair. The additional HUB project has now been offered for two years with positive student feedback. Students enjoy the development of web-based applications (in particular, learning *Angular*), the freedom of self-organization (e.g., specializing on a certain technology), and the interactions with industry partners (giving students insights into professional work practices). However, from a teaching perspective, the addition of the HUB has posed new challenges:

- (1) It is difficult to estimate how much we can expect from students in our project setting. The applied technologies are not trivial to learn, and the quality and complexity of existing projects differ. We cannot easily draw from past experiences, neither ours nor others, because the HUB differs from a classical team project and focuses on different competencies.
- (2) The time required for teaching the HUB is higher than for the LAB. To answer the multitude of questions raised by students, instructors need detailed knowledge about software requirements that differ among the concurrently developed software products, and about many details of the comprehensive technology stack employed. The necessary teaching resources thus depend on the student cohort and the own experience of the lecturing staff with the employed technologies, and are unevenly distributed throughout a semester.
- (3) Individualizing student grades for the HUB module is challenging when compared to the SWL module, where students collaborate in fixed teams and with close supervision by staff. In the HUB, students operate in flexible teams and collaboratively work on tasks using techniques such as pair programming. Additionally, because the existing projects and applied technologies are plentiful, students need to specialize on different topics during the semester. Not all of their contributions to the projects are directly visible in terms of added or changed functionality, but also include refactoring source code, adding automated tests, communicating with stakeholders, or enhancing documentation.

3 Lessons Learned

In response to these challenges, we have adjusted the HUB module throughout the past three semesters. The remainder of this section outlines the most important lessons learned by us along the way. However, certain issues remain, which are discussed in Sect. 4.

Online-tutorials are beneficial. Due to the COVID-19 pandemic, we have shifted from face-to-face tutorials to online material, which introduces tools and techniques for practical software development and consists of video recordings, slides, and hands-on tasks. Sample tutorials focus on “*Managing the Development Process using GitLab*”, “*Persisting Data with JPA using Spring Boot*”, and “*Writing Clean Code and Assuring Quality with Code Reviews*”. Providing asynchronous online material gives the students the option to (re-)watch the tutorials at their own pace when facing the challenges addressed in the tutorials, without overwhelming them with too much content at the beginning of the semester.

A good demo project is key for well-structured projects. To ensure that the projects can be maintained by students, all projects should have a similar structure and employ similar technologies. We have learned that providing a demo project is more effective than stipulating a detailed coding convention, and have introduced two demo projects that showcase the technology stacks for the LAB and HUB modules, resp. These projects serve as a skeleton for products developed from scratch, which leads to a similar structure across all projects and minimizes the initial overhead. Previously, students spent the semester start on setting up their own project according to the coding convention, which usually required additional support and clarification by teaching staff. Our demo projects also include a CI/CD pipeline, which checks for proper formatting, builds the projects, runs automated tests, generates metrics such as code coverage, and deploys the application.

Keep it relatively simple wrt. technologies. The provision of tutorial videos and demo projects aids students in acquiring the skills for mastering today’s software development technologies. While we do not forbid students to use additional external libraries, frameworks, and platforms, technology proposals now have to be discussed with the industry partner and the Chair. This discussion primarily focuses on the effect of the inclusion of the requested technology on maintainability. Before, for example, a student already familiar with the *ElasticSearch* engine proposed to use it for one product, which turned out to be too complex for other students and had to be replaced by a simpler search technology in a later semester.

Expect less functionality than in classical student projects. Compared to a project structure in which a new project is developed from scratch and thrown away at semester end, we expect less functionality to be implemented by students in the HUB. When adding

features to a new or existing software project, students need more time for planning and implementation, as they need to consider clean code and provide documentation early on. When extending existing projects, students need time to understand the codebase and product requirements. This is particularly true if the codebase contains source code of poor quality, incomplete documentation, or technologies unfamiliar to the students. For maintaining projects, students need to invest time in tasks that do not result in new functionalities, but still resemble a significant contribution. These activities include, but are not limited to refactoring source code, fixing bugs, adding automated tests, and enhancing documentation.

A journal written by each student helps to individualize grades. Determining the grade of an individual student in our HUB project requires multiple forms of assessment and is based on the student's contribution to the codebase, including documentation. For each software product developed or maintained in the HUB, students collectively submit a report that summarizes the initial state of the product, the contributions made throughout the semester, and directions for future enhancements. To be able to map contributions to individual students, each student has to document each sprint in a journal, including the time spent on each topic they worked on, and a reflection on their learnings. This information is cross-checked in an oral exam, in which individual students answer questions on design and implementation choices, quality assurance, project management, and teamwork. While this grading is time-intensive (about 5–7 person days), it is perceived as fair by students and staff alike. Each product report is also a valuable first source of information for future student generations who need to maintain the product.

Answering questions on low-level implementation details is time consuming. In past semesters, most of the teaching staff's effort in the LAB and HUB modules, and partially also the effort of our industry partners, has been spent on answering student questions concerning implementation-specific details, not at least because of the time-consuming task of inspecting the codebase in question. This winter semester 2021/22, we offer a help desk organized by a student assistant for answering those types of question. In doing so, we seek to enable our industry partners and us lecturers to focus on design-level issues and software quality aspects, which are critical for software longevity.

4 Conclusions

Despite the challenges identified in our teaching concept, we have established our current student project structure because we believe that being able to understand complex systems and writing long-living software is highly relevant when tackling current and future IT projects. After two years of experience with the HUB and based on the positive feedback from students, we have come to the conclusion that our teaching concept is suitable for making students aware of the importance of sustainability in software development and motivating them to write clean code and good documentation.

Remaining challenges. Certain challenges remain, however, which ought to be discussed by the community of lecturers involved in software engineering education. Guiding questions for discussion may be the following:

(1) Should competencies of understanding complex systems and writing long-living software be taught at Bachelor or Master level, or both (see, e.g., Zukunft [Zu16])?

(2) Is giving students practical experience with software maintenance enough, or do we need to accompany project work with additional theoretic input (see, e.g., the module “Software Evolution” taught at Univ. Heidelberg¹)?

(3) How do we grade contributions to the codebase, documentation, and quality assurance in the context of maintaining long-living software? How could a grading scheme be devised that accounts for these different types of contributions? What alternative approaches to our journal can be adopted to simplify the individualization of grades?

(4) What changes are necessary for the teaching concept to scale to large student cohorts? (So far, we have run the HUB module with up to 25 students in a single semester.)

(5) What other approaches exist for teaching the competencies needed for developing long-living software? (One approach is for students to contribute to open source projects, as noted by Spinellis [Sp21].)

Acknowledgements. We thank the reviewers and our colleague Eugene Yip for their suggestions which have helped us to improve the paper.

References

- [BR00] Bennett, K. H.; Rajlich, V.: Software maintenance and evolution: A roadmap. In: ICSE 2000. ACM, pp. 73–87, 2000.
- [Go15] Goltz, U.; Reussner, R. H.; Goedicke, M.; Hasselbring, W.; Mörtin, L.; Vogel-Heuser, B.: Design for future: Managed software evolution. *Comput. Sci. Res. Dev.* 30/3-4, pp. 321–331, 2015.
- [Sp03] Spinellis, D.: Reading, Writing, and Code. *ACM Queue* 1/7, pp. 84–89, 2003.
- [Sp21] Spinellis, D.: Why computing students should contribute to open source software projects. *Commun. ACM* 64/7, pp. 36–38, 2021.
- [Ta05] Tate, K.: *Sustainable Software Development: An Agile Perspective*. Addison-Wesley Professional, 2005, ISBN: 0321286081.
- [Zu16] Zukunft, O.: Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen, 2016, URL: <https://dl.gi.de/handle/20.500.12116/2351>.

¹ https://se.ifi.uni-heidelberg.de/teaching/previous_semesters/ws_202021/softwareevolution.html (accessed 2021/12/08)

Divekit - Digitalisierung und Individualisierung als Schlüssel für eine moderne Softwaretechnik-Ausbildung

Stefan Bente¹, Jann Intveen², Fabian Krampe³

Abstract: Um Kompetenzen in modernen Softwaretechnik-Ansätzen wie Domain Driven Design oder Clean Code zu erwerben, müssen diese von Lernenden im Rahmen von Praktika auch selbst angewendet werden. Erst dann können sie in ihrer Komplexität und ihren Konsequenzen vollständig verstanden werden. Digitale E-Assessment-Tools eröffnen die Möglichkeit, dies mit vertretbarem Betreuungsaufwand zu tun, indem große Teile der studentischen Lösung automatisiert getestet werden. Somit erhalten die Studierenden direktes formatives Feedback, und die Betreuer:innen können sich auf Beratung in komplexen Fragestellungen konzentrieren. Dieses Paper formuliert 12 Prinzipien für E-Assessment-Tools im Bereich Softwaretechnik und Coding, darunter Individualisierung und Praxisnähe der gestellten Aufgaben. Anschließend wird das Open-Source-Framework Divekit vorgestellt, das diese Prinzipien erfüllt. Möglichkeiten und Grenzen von Divekit werden anhand einer Fallstudie kritisch bewertet.

Keywords: E-Assessment; Digitalisierung; Softwaretechnik; Coding; Individualisierung; DDD

1 Einleitung

Ein modern ausgerichtetes Softwaretechnik-Praktikum hat zwei primäre Aufgaben. Auf der einen Seite soll es den *Lernenden* ermöglichen, vorab vermittelte Inhalte selbst praktisch anzuwenden - und zwar in einer Arbeitsumgebung, die möglichst nahe an der berufspraktischen Realität ist. Dies gilt in besonderem Maß für die Vermittlung von komplexen Konzepten wie Domain Driven Design, Test Driven Development oder Clean Code. Hier ist zum echten Kompetenzerwerb tatsächliche Programmierung nötig - Lückentexte und Multiple-Choice-Fragen genügen nicht.

Auf der anderen Seite benötigen *Lehrende* eine effektive Rückmeldung zum Lernerfolg des gesamten Kurses (zur Überprüfung der Lernziele) und jedes einzelnen Mitglieds der Lerngruppe (zur Erteilung von Praktikumstestaten, oder der Benotung einer Klausur). Dies muss einher gehen mit einem möglichst hohen Grad an Automatisierung, um die Ressource "Betreuungspersonal" an den Stellen einsetzen zu können, wo ein echter Erkenntnis-Mehrwert für Studierende entsteht (beispielsweise in komplexen Fachdiskussionen zwischen Betreuer:in und Studierenden).

Diese Ziele lassen sich nur durch eine verstärkte Digitalisierung von studentischen Softwaretechnik-Praktika erreichen. Dieses Paper stellt zunächst die Anforderungen daran

¹ TH Köln, Cologne Institute for Digital Ecosystems (CIDE), Steinmüllerallee 1, 51643 Gummersbach, stefan.bente@th-koeln.de

² s.o., JannIntveen@gmail.com

³ s.o., fabian.krampe@th-koeln.de

auf, und begründet diese ausgehend vom Wissensstand der Software-Community sowie der hochschuldidaktischen Forschung. Dann wird das Werkzeug **Divekit** (Toolkit for **I**ndividual **E**xercises) beschrieben, das an der TH Köln entwickelt wurde und seit mehreren Semestern erfolgreich in verschiedenen Lehrveranstaltungen eingesetzt wird. Diese Erfahrungen werden abschließend in einer Fallstudie bewertet.

2 Zwölf Prinzipien für eine praxisnahe digitale Softwaretechnik-Lehre

Eine praxisnahe digitale Softwaretechnik-Lehre muss einerseits didaktische Anforderungen erfüllen, andererseits aber auch die zurzeit gebräuchlichen softwaretechnischen Paradigmen unterstützen. Im Rahmen einer Masterarbeit [In21] wurde hierzu neben einer umfassenden Recherche auch eine Anzahl von Experteninterviews durchgeführt, um die Erfahrungen von anderen Lehrenden und Expert:innen auf diesem Gebiet einzubeziehen. Daraus ergaben sich 12 Prinzipien, wie eine erfolgreiche digitale Softwaretechnik-Lehre gestaltet sein sollte. Diese stellen den Orientierungsrahmen für Entwicklung und Einsatz von Divekit dar.

(P1) Realitätsnahe Programmieraufgaben gemäß *Active Learning* Moderne Programmierkonzepte werden am besten anhand von realistischen Problemstellungen und mit echten Tools (IDE, Git, Build-Pipelines) erlernt. Mittels *Active Learning* [FB09] können sich Lernende in Einzel- oder Gruppenarbeit damit beschäftigen. Lehrende tragen in bestimmten Intervallen Erkenntnisse zusammen und geben Inhaltsimpulse.

(P2) Mindestens Stufe 3 (*Applying*) der Revised Bloom's Taxonomy Klassische E-Learning-Werkzeuge bleiben oft auf den Stufen 1 (*Remembering*) oder 2 (*Understanding*) der Revised Bloom's Taxonomy [Ar16] stehen. Ein praxisrelevantes Lernen setzt aber mindestens voraus, die erlernten Konzepte auch selbstständig anwenden zu können (*Applying*, Stufe 3).

(P3) Intrinsische über extrinsische Motivation Intrinsische Lernprozesse gelten als besonders effektiv [RD00, 56f] und sollten daher gefördert werden, ergänzt durch extrinsische Faktoren wie etwa eine Erfolgskontrolle bei Praktikumsaufgaben.

(P4) Stetiges formatives Feedback für Lernende durch E-Assessment Ein E-Assessment-System muss effektives, kontinuierliches formatives Feedback [RMP04, 17ff] während des gesamten Lehr- und Lernprozesses bieten, auch wenn ein Teil des Feedbacks automatisch erfolgt.

(P5) Stetiger Zugriff auf individuelles formatives und summatives Assessment für Lehrende Lehrende benötigen Einsicht in das formative Assessment der Studierenden, um Stoff und Lehrmethoden dynamisch an den Lernfortschritt anpassen zu können. Iteratives summatives Assessment (z.B. in Form von Praktikums-Meilensteinen) ermöglicht es zusätzlich, Themen und Konzepte erneut aufzugreifen, die noch nicht gänzlich verstanden wurden [HRL20, S. 281].

(P6) Förderung von Zusammenarbeit im Team, aber individuelle Abgabe von Lösungen

In informellen Studiengruppen spezialisieren sich oft einzelne Studierende auf eine bestimmte Veranstaltung und bearbeiten diese stellvertretend für die ganze Gruppe. In diesem Fall beschäftigen sich nicht alle Gruppenmitglieder in derselben wünschenswerten Tiefe mit den Studieninhalten. Um dem vorzubeugen, sollten Aufgaben in einem Maß individualisierbar sein, das zwar Diskussionen und Zusammenarbeit zwischen Lernenden ermöglicht, aber das einfache Kopieren von Lösungen anderer erschwert.

(P7) Keine Software-Architektur ohne Coding Wer Architektur nicht in Code umsetzen kann, wird im berufspraktischen Kontext selten ernst genommen⁴. Eine praxisnahe digitale Softwaretechnik-Lehre muss daher komplexe Architekturstile und Patterns auch in Code abbilden und die Studierenden praktisch umsetzen lassen.

(P8) Pragmatische Vermittlung von Modellierungs-Fähigkeiten Eine hauptsächliche Fokussierung auf UML-Kenntnisse ist in der Softwaretechnik-Lehre nicht mehr zeitgemäß (siehe P7). Wie in [AB19] ausgeführt, gibt es aber einige UML-Diagramme, die auch bei einer agilen, wenig dokumentenzentrierten Vorgehensweise sinnvoll nutzbar sind. Dazu gehören insbesondere Klassen- und Use-Case-Diagramme, und für besondere Aspekte auch Komponenten-, Deployment-, Status-, Aktivitäts- und Sequenzdiagramme, die bei einer Digitalisierung von Softwaretechnik-Lehre berücksichtigt werden sollten.

(P9) Realistische Größe von Aufgabenstellungen Wenn Studierende immer nur kleine “Aufgabenschnipsel” bearbeiten, dann wird sich kein Gefühl für den Umgang mit Komplexität einstellen. Daher muss es digitale Lehre ermöglichen, komplexe Aufgabenstellungen zu formulieren, die sich über mehrere Iterationen (wie etwa verschiedene Praktikums-Meilensteine) weiterentwickeln lassen.

(P10) Stetige Entwicklung gegen Unit Tests Der Ansatz des *Test Driven Development* (TDD) sieht vor, dass Tests geschrieben werden, bevor die dazugehörige Funktionalität implementiert wird [Be02]. Dies hilft Studierenden, Komplexität durch Aufteilung in kleine Code-Fragmente zu beherrschen, die mithilfe vorab geschriebener Tests überprüft werden.

(P11) Fokus auf Prinzipien des Domain Driven Designs (DDD) Domain Driven Design (DDD) [Ev04] ist in der IT-Community als Basis für moderne, lose gekoppelte Architekturen weit verbreitet⁵. Eine praxisnahe digital unterstützte Programmierausbildung muss es daher ermöglichen, den Studierenden Aufgaben zum Tactical Design (Entities, Repositories, Value Objects, Aggregates, Services [Ev04, S. 89–158]) zu stellen⁶.

⁴ Wie Martin Fowler schreibt: “Like many in the software world, I’ve long been wary of the term ‘architecture’ as it often suggests a separation from programming and an unhealthy dose of pomposity.” [Fo19]

⁵ Als - zugegebenermaßen recht anekdotisches - Indiz mag gelten, dass auf einer größten IT-Community-Konferenzen in Deutschland, der OOP in München, im Jahr 2020 [DA21] in einem Drittel der Vorträge mit Architekturbezug Teilaspekte von DDD thematisiert wurden (16 von 48 Vorträgen, gezählt nach Tags).

⁶ Auch eine Hinführung zum Strategic Design, z.B. dem Begriff des “Bounded Context” [Ev04, S. 335], ist wünschenswert, kann aber eher im Kontext eines Praxis- oder Lehrforschungsprojekts geleistet werden.

(P12) Fokus auf Clean-Code-Regeln und SOLID-Prinzipien *Clean Code* betont die Relevanz und die Bedeutsamkeit von sauber geschriebenem, selbsterklärendem Programmcode. Eng verbunden damit sind die SOLID-Prinzipien [Ma00]. Beide Ansätze sind in der IT-Community durchgehend akzeptiert und sollten in einer digital unterstützten Softwaretechnik-Lehre thematisierbar sein.

2.1 Abdeckung der genannten Prinzipien durch existierende E-Assessment- und E-Learning-Systeme

Keunig et al. [KJH18] haben 2018 systematisch über 100 digitale Programmier-Lernsysteme bezüglich ihres Feedbacks für Lernende untersucht. Eine verwandte Untersuchung haben Wasik et al. [Wa18] durchgeführt, die sogar noch mehr Softwaresysteme zur Onlinebewertung vergleichen. Dabei finden sich einige Ansätze, die eine ähnliche Richtung wie die genannten 12 Prinzipien verfolgen. Hier wäre etwa der *Praktomat* [BHS17] zu nennen, ein E-Assessment-Werkzeug, mittels dessen den Studierenden Programmieraufgaben gestellt und deren Lösungen überprüft werden können. Zusätzlich werden Funktionen zur automatischen Erstellung von Aufgabenvarianten bereitgestellt. Einen stark auf automatisiertes Feedback ausgerichteten Ansatz verfolgt *Code FREAK* [WK21] der FH Kiel. Auch im Bereich der Mathematik existieren viele E-Assessment-Systeme, die Aufgaben individualisieren, beispielsweise *ActiveMath* [Go10, S. 77–83] oder *MATEX* [He18].

Tools, die sämtliche oben geforderten Prinzipien umsetzen, finden sich allerdings nach unseren Recherchen nicht. Die Realität der digitalen Lehre an deutschen Hochschulen scheint eher aus in Moodle oder ILIAS umgesetzten Single-/Multiple-Choice-Fragen und Lückentexten zu bestehen, die aus einem Fragenpool zufällig ausgewählt werden. Ein Praxisbezug (P1-P3, P7-P12) kann damit nicht zufriedenstellend abgebildet werden. Daher fiel hier die Entscheidung, aufbauend auf Gitlab die eigene Lösung **Divekit** als Open-Source-System zu konzipieren und entwickeln.

3 Konzept und didaktischer Rahmen von Divekit

Eine Übungsaufgabe mit Divekit, etwa im Softwaretechnik-Praktikum, durchläuft die fünf Phasen (1) *Konzipieren*, (2) *Individualisieren und Verteilen*, (3) *Bearbeiten*, (4) *Feedback* und (5) *Korrigieren und Auswerten*.

3.1 Phase 1: Konzipieren

Der oder die Lehrende erstellt eine Aufgabe mitsamt einer Musterlösung für die Betreuer:innen, die diese in Beratung und Korrektur (falls manuelle Korrektur nötig ist) einsetzen können. Die Aufgabenstellung besteht aus Code, Diagrammen im UMLet-Format⁷, Konfigurationsdateien sowie Markdown-Dateien zur Beschreibung. All das wird in einem Gitlab-Repository (dem “Origin-Repo”) zusammengefasst.

⁷ UMLet ist ein freier UML-Editor (<https://www.umlet.com/>), der einen großen Teil der UML-Spezifikation in einem einfachen Scripting-Ansatz umsetzt. Dadurch ist er sehr intuitiv zu bedienen, und kann mittels Batch-Processing leicht in ein Framework wie Divekit eingegliedert werden.

In allen genannten Artefakten (Text, Code, Diagramme) können Platzhalter-Begriffe verwendet werden, erkennbar an der Schreibweise Auf diese Weise sind auch komplexe natürlichsprachliche Aufgabenstellungen umsetzbar, die dann von der fachlichen Modellierung bis hin zu Umsetzung in automatisiert getesteten Code von den Studierenden durchgängig bearbeitet werden können.

List. 1: Ausschnitt aus einem Aufgabentext mit Platzhaltern

```
$Robots$ $robot_purpose$. They can be moved across multiple
$robot_grids$ with $walls$, which cannot be passed. A $robot_grid$ is
rectangular and consists of $rasters$. One $raster$ can contain only
one $robot$. Therefore, $robots$ are like $walls$ for each other.
```

Für diese Platzhalter werden in einer Konfigurationsdatei im JSON-Format Variationen spezifiziert. Für einen Platzhalter wie `$robot$` und seine Attribute können dabei Variationen wie “Maintenance Droid” oder “Mining Machine” angegeben werden. Die Kombination der Variationen kann flexibel festgelegt werden. Spezialformen wie etwa getter/setter (`getMaintenanceDroid(...)`) werden von DiveKit automatisch bereitgestellt. Ebenso erkennt Divekit Groß- und Kleinschreibung am Satzanfang (`$Robot$` - `$robot$`) und reguläre Pluralformen (`$robot$` - `$robots$`)⁸.

3.2 Phase 2: Individualisieren und Verteilen

Die generisch verfasste Aufgabenstellung wird so individualisiert, so dass jede:r Studierende:r ein individuelles Gitlab-Repository erhält. Die obige Aufgabenstellung aus dem Origin-Repo (siehe List. 1) kann sich für zwei unterschiedliche Studierende dann so darstellen:

List. 2: Zwei individualisierte Instanzen der Aufgabenstellung aus List. 1

Maintenance droids fix minor problems, like loose plugs, etc. They can be moved across multiple spaceship decks with obstacles, which cannot be passed. A spaceship deck is rectangular and consists of cells. One cell can contain only one maintenance droid. Therefore, maintenance droids are like obstacles for each other.

Mining machines collect minerals like cobalt, lithium, etc. They can be moved across multiple mining fields with barriers, which cannot be passed. A mining field is rectangular and consists of squares. One square can contain only one mining machine. Therefore, mining machines are like barriers for each other.

Diese Individualisierung geschieht in einem automatisierten Batchlauf, in dem die Platzhalter zu zufälligen Varianten expandiert werden. Durch die Kombination unabhängiger Platzhalter entstehen dann schnell so viele Varianten, dass es selbst in einer großen Kohorte keine zwei gleichen Aufgabenstellungen gibt.

Diese Expansion funktioniert nicht nur in Aufgabentexten, wie schon gesehen. Auch der Code (Tests, gegen die Studierende implementieren müssen, aber auch als Hilfestellung

⁸ Unregelmäßige Pluralformen (women, children, mice etc.) können explizit in der Konfigurationsdatei überschrieben werden.

vorgegebene Stub-Klassen) kann entsprechend individualisiert werden. Das gleiche gilt für Markdown-Tabellen, in denen Studierende schon Teillösungen angeben müssen (beispielsweise für ein fachliches Glossar aus Basis des Domain Models), und auch für UML-Diagramme. Dadurch kann die Aufgabenstellung beispielsweise durch ein individualisiertes Klassen-, Zustands-, Sequenz- oder Aktivitätsdiagramm besser erklärt werden (siehe Abb. 1).

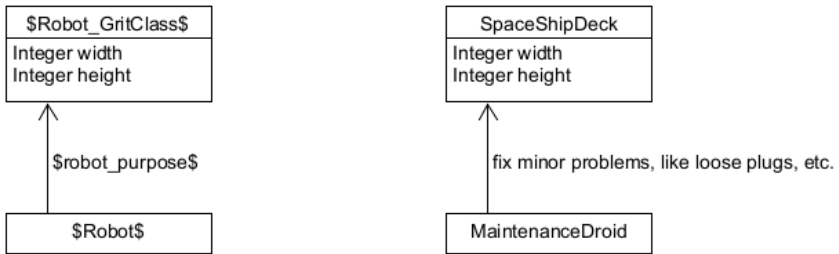


Abb. 1: UML-Diagramm in “Origin-Repo” (links), individualisierte Variante (rechts)

Divekit bringt eine vollständige Test-Bibliothek mit, die speziell auf mittels *Spring Data JPA*[VM] umgesetzte DDD-Konzepte zugeschnitten ist. Andere Teile sind auf REST-APIs ausgelegt. Diese Bibliothek kann leicht für weitere Anwendungsbereiche erweitert werden.

Für alle Studierenden speichert Divekit auf Wunsch die jeweilige Individualisierungs-Konfiguration und wendet sie bei nachfolgenden Aufgabenstellungen erneut an. Dadurch sind auf einander aufbauende Aufgabenstellungen möglich, die die originale Aufgabe erweitern. Jede:r Studierende findet dann wieder die vertraute Konfiguration vor.

Tests können in Divekit vor den Studierenden versteckt werden, da intern immer zwei Repositories pro Studierendem erzeugt werden, eins davon für den Studierenden unsichtbar. Dort können “Hidden Tests” abgelegt werden, deren Code die Lösung direkt verraten würde.

3.3 Phase 3: Bearbeiten

Die Studierenden bearbeiten die gestellte Aufgabe. Jede:r hat ein individuelles Repo mit eigener Aufgabenstellung. Die Studierenden reichen ihre Lösungen durch ein `git commit / git push` ein. In den Lehrveranstaltungen, in denen Divekit bislang eingesetzt wird, werden die Studierenden dennoch ausdrücklich zur Teamarbeit ermuntert. Flankierende ganztägige Übungen in Kleingruppen fördern dies. Das Feedback und die Rückfragen der Studierenden legen nahe, dass die gemeinsame Basisstruktur der Aufgaben erkennbar bleibt, was eine Diskussion ermöglicht. Ein simples Kopieren fremder Lösungen führt aber im Allgemeinen nicht zum Ziel.

3.4 Phase 4: Feedback

Über eine personalisierte Webseite erhalten die Studierenden Feedback. Dort sind die Rückmeldungen aller Tests zu sehen (rot/grün und Fehlermeldung, falls rot). Im Fall von

komplett automatisierten Tests erfolgt das Feedback innerhalb von ca. zwei Minuten nach dem `git push`.

Zusätzlich hat es sich in der Praxis bewährt, ein Diskussionsforum für Fragen zu eröffnen. *Discord* scheint aufgrund seiner große Verbreitung in der Gaming-Community eine gute Wahl zu sein, weil es dadurch Studierenden, Tutoren und Betreuer:innen gleichermaßen vertraut ist und niederschwellig genutzt wird. Es ist sinnvoll, die Studierenden zu Fragen zu ermuntern, wenn das Divekit-Feedback für sie unverständlich ist. Andernfalls droht ein “Brute-Force”-Vorgehen seitens einiger Studierenden, bei dem ohne tieferes Verständnis der Aufgabenstellung zahlreiche Varianten nacheinander durchprobiert werden.

3.5 Phase 5: Korrigieren und Auswerten

Nicht alle Teile einer Lösung lassen sich sinnvoll automatisiert testen. Divekit erlaubt auch manuelles Feedback, das die Betreuer:innen in einer dedizierten Markdown-Datei hinterlegen. Hier bietet Divekit Tools, um die Warteschlange der eingereichten Lösungen zu sehen und die jeweils am längsten wartende Lösung korrigieren zu können. Dieses manuelle Feedback wird dann auf der Testseite neben dem automatisierten Feedback angezeigt.

Als Strategie hat es sich bewährt, basale Aspekte der Lösung automatisiert zu prüfen und das menschliche Feedback erst dann zu geben, wenn die automatisierten Tests “grün” sind. Auf diese Weise wird die knappe Resource “Kontaktzeit” für semantisch höherwertige, komplexe Fragestellung reserviert. Als Beispiel für so ein Vorgehen kann in einem Domain Model (als UML-Klassendiagramm) das Vorhandensein der wesentlichen Entities und deren Haupt-Beziehungen automatisiert geprüft werden (Divekit unterstützt das mit einer eigenen Testbibliothek). Ist dieser Test bestanden, können Betreuer:innen Assoziationstypen und Multiplizitäten manuell prüfen - was für das breite Spektrum richtiger Modellierungen angemessener ist als eine automatisierte Prüfung, die dann unrealistischerweise von “der einen richtigen” Lösung ausgehen würde.

Ein weiteres praktisches Beispiel bei der Umsetzung in Code wäre es, die Clean-Code-Regel “Small!” [Ma09, S. 34] automatisiert prüfen zu lassen (für Divekit wurde dafür im Rahmen einer Bachelorarbeit eine auf PMD [Pr21] basierende Clean-Code-Testbibliothek entwickelt). Die semantisch weitaus komplexere “Stepdown Rule” (“We want the code to read like a top-down narrative” [Ma09, S. 37]) wird dann hingegen manuell durch Betreuer:innen geprüft.

3.6 Umsetzung der 12 Prinzipien durch Divekit

In der nachfolgenden Tabelle ist zusammengefasst, wie Divekit die oben eingeführten 12 Prinzipien für eine praxisnahe digitale Softwaretechnik-Lehre unterstützt.

Tab. 1: Unterstützung der 12 Prinzipien durch Divekit

Prinzip	Unterstützende Features / Good Practices in Divekit
(P1) Realitätsnahe Programmieraufgaben gemäß <i>Active Learning</i>	Große, zusammenhängende Aufgabenstellung; Individualisierung erzwingt Beschäftigung mit dem Stoff; flankierender Flipped-Classroom-Ansatz für Inhalte
(P2) Mindestens Stufe 3 (<i>Applying</i>) der Revised Bloom's Taxonomy	Studierenden müssen die Aufgabenstellung selbstständig in Modelle und Code umsetzen
(P3) Intrinsische über extrinsische Motivation	Praxisnähe, Komplexität der Aufgabenstellung und realistische Tools (Git, IDE) wirken intrinsisch motivierend; zusätzliche extrinsische Leistungskontrolle
(P4) Stetiges formatives Feedback für Lernende durch E-Assessment	Testseite von Divekit gibt formatives Feedback; Unverständliches wird im Dialog mit den Lehrenden geklärt, die über Discord erreichbar sind
(P5) Stetiger Zugriff auf individuelles formatives und summatives Assessment für Lehrende	Divekit-Werkzeuge erlauben das kontinuierliche Monitoring der Gesamtkohorte; für einen individuellen Eindruck kann das Studierenden-Repo geclont werden
(P6) Förderung von Zusammenarbeit im Team, aber <i>individuelle</i> Abgabe von Lösungen	Jede:r gibt eine individuelle Lösung ab (auch für große Kohorten aufwandsneutral skalierbar); flankierende Übungen in Kleingruppen zur Förderung der Teamarbeit
(P7) Keine Software-Architektur ohne Coding	Komplexe Aufgabenstellungen mit Modellierung und Coding; Speicherung der Individualisierungs-Konstellationen für aufeinander aufbauende, immer komplexer werdende Praktikums-Meilensteine
(P8) Pragmatische Vermittlung von Modellierungs-Fähigkeiten	Individualisierung von UML-Diagrammen in der Aufgabenstellung; Test-Library zur automatischen Prüfung von UML-Klassendiagrammen
(P9) Realistische Größe von Aufgabenstellungen	Siehe Kommentar zu P1-P3.
(P10) Stetige Entwicklung gegen Unit Tests	“Test müssen grün werden”, um das Praktikum zu bestehen; Divekit kann auch prüfen, ob selbst Tests geschrieben wurden
(P11) Fokus auf Prinzipien des Domain Driven Designs (DDD)	Native Test-Library für Entities, Value Objects und Aggregates; Strategic Design (z.B. “Bounded Contexts”) allerdings nur indirekt über Tabellentests prüfbar

Prinzip	Unterstützende Features / Good Practices in Divekit
(P12) Fokus auf Clean-Code-Regeln und SOLID-Prinzipien	Dedizierte Test-Library für ausgewählte Clean-Code-Regeln und SOLID-Prinzipien

4 Anwendung von Divekit am Beispiel der Veranstaltung Softwaretechnik 2 (ST2)

Die Veranstaltung *Softwaretechnik 2 (ST2)* im Studiengang Informatik der TH Köln beschäftigt sich mit Architektur und Umsetzung von komplexen Softwaresystemen. Durchschnittlich besuchen den Kurs etwa 120 Studierende. Im Sommersemester 2020 wurde erstmals Divekit eingesetzt, um DDD-Konzepte als Programmieraufgaben, die automatisch getestet wurden, in das Praktikum zu integrieren. Aufgaben wurden auf Gitlab hochgeladen, mithilfe von JUnit-Tests überprüft und die Ergebnisse auf einer Übersichtsseite dargestellt. Alle Tests auf der Übersichtsseite mussten bis zum Ende einer Deadline erfolgreich durchlaufen sein, damit das Praktikum als bestanden galt.

Aufgrund der beginnenden Corona-Pandemie waren Präsenztermine nicht mehr möglich. Vorlesungen fanden online über Zoom statt. Da sämtliche Praktikumsaufgaben von zu Hause bearbeitet werden mussten, wurden diese individualisiert und automatisch auf Gitlab verteilt. Auch wenn die JUnit-Tests am Ende bestimmt haben, ob jemand bestanden hat, durften Betreuer über Discord nach Belieben angeschrieben werden, um Ratschläge oder Hilfestellungen zu erhalten. Wenn nötig, fanden sich Lernende und Betreuer in einem Discord-Channel ein, um das Problem zu diskutieren oder gegebenenfalls auch eine Bildschirmübertragung zu starten, damit Betreuer einen tieferen Einblick in die Aufgabebearbeitung hatten.

Nicht nur die Praktika, sondern auch die Klausur musste digital von Zuhause stattfinden. Folglich wurde auch die Klausur ebenfalls mit Hilfe von Divekit individualisiert und verteilt. Lernende hatten jeweils ein eigenes Repo mit Klausuraufgaben. Für Fragen während der Klausur wurde ein eigener Fragen-Channel in Discord eingerichtet, passende Antworten wurden in einen anderen Antworten-Channel eingestellt. Natürlich waren auch Fragen außerhalb von Discord über E-Mails oder Telefonate mit einem Betreuer erlaubt.

4.1 Feedback der Studierenden

Nach dem Praktikum und der Klausur wurde auf Discord Feedback dazu gesammelt. Außerdem wurde eine anonyme Feedback-Umfrage gestartet, an der sich 17 Studierende beteiligten. Das Gesamtkonzept des Praktikums kam bei den Lernenden gut an. Die Kommunikation über Discord wurde positiv wahrgenommen. Hier fiel insbesondere auf, dass sich Lernende auch gegenseitig über Discord halfen. Auch die automatischen Tests wurden geschätzt, weil sie ein schnelles Feedback boten.

Bezüglich der Klausur wurde das Konzept mit viel Programmierung während der Klausur gut angenommen. Schwierigkeitsgrad und Komplexitätsgrad der Klausuraufgaben wurden

als angemessen empfunden. Allerdings berichteten mehrere Studierende von einem Zeitproblem. Für die Zukunft wünschten sich die Studierenden klarere Aufgabenstellungen mit weniger Kontextwechseln. Diese scheinen ein schnelles Verstehen der Aufgaben zu behindern. Leider sind sie ein Nebenprodukt der Individualisierung - eine durchgehende Aufgabenstellung für die ganze Klausur wäre sehr komplex für die Ersteller.

Ein Großteil der Studierenden gab an, dass die Praktikumsabgabe über automatisierte Tests besser sei als herkömmliche mündliche Abgaben. Meistens waren Fehlermeldungen von Unit-Tests aussagekräftig und nachvollziehbar und erlaubten somit ein schnelles Debugging. Dies wurde allerdings erheblich erschwert, wenn Tests versteckt waren, der Programmcode des Tests also nicht einsehbar war.

Wenn Lernende bei der Bearbeitung der Aufgaben keinen Fortschritt machen konnten, weil das Feedback der automatischen Tests nicht ausreichte, wurde Feedback über Discord ermöglicht. Fast alle Studierenden gaben an, dass sie mit dem Kommunikationskanal Discord zur Betreuung viel zufriedener sind als mit herkömmlichen Kanälen wie beispielsweise Sprechstunden, E-Mails oder Foren. Dies könnte unter anderem durch das schnelle Feedback begründet sein, das durch Discord ermöglicht wird. Zwei Drittel der Studierenden erklärten, dass sie im Schnitt innerhalb von einer Stunde eine Hilfestellung erhielten.

Auch aus Sicht der Lehrenden war die Veranstaltung ein Erfolg. In Bezug auf DDD konnte der Bereich des *Tactical Designs* gut in Praktikumsaufgaben thematisiert werden, auch wenn diese von der Komplexität nicht an zu lösende Probleme aus der Praxis heranreichten. Mithilfe der Technologien Java und Spring konnte aufgezeigt werden, inwiefern sich bestimmte Konzepte aus dem DDD in Programmcode überführen lassen.

Das obige Feedback floss in die Weiterentwicklung von Divekit sowie der Aufgabenstellungen für ST2 im Sommersemester 2021 ein. Statt der Vorlesungen wurden zusätzlich der gesamte Stoff als Videos bereitgestellt, um die Veranstaltung im Flipped-Classroom-Format stattfinden zu lassen. Das Feedback wurde in ähnlicher Weise wie im SS2020 erhoben und war durchgehend ähnlich positiv. Das Zitat eines Studierenden fasst (in für die Lehrenden ermutigender Weise ...) viel positives Feedback der Studierenden zusammen: *“Meiner Meinung nach eines der, wenn nicht sogar das beste Modul mit dem meisten Realitätsbezug im gesamten Studium. Die Vorlesungsthemen sind endlich mal in moderner Form aufbereitet und auch die verwendeten Tools sind aktuell. Warum können sich andere Dozenten nicht ein Vorbild daran nehmen?”*

4.2 Vergleich der Studierenden-Ergebnisse zu früheren Klausuren

Vergleicht man die Klausurergebnisse aus den Jahren 2020 und 2021, wo Divekit eingesetzt wurden, mit früheren Jahren (2017 - 2019), so liegt der Notenschnitt mit Divekit bei 2,5 (drei Klausuren), in der Zeit vor 2020 ohne Divekit bei 3,0 (sechs Klausuren). Die Klausuren sind vor und nach Einführung von Divekit inhaltlich vergleichbar⁹.

⁹ Die Klausuren sind nach der Umstellung auf Divekit eher schwerer geworden, da Code jetzt tatsächlich während der Klausur geschrieben und in einem Repo gepusht werden muss, anstatt wie vorher eher kurze “Pseudocode”-Fragmente auf Papier zu schreiben.

Die Durchfallquote bleibt gleich (jeweils ca. 14,5%), wobei zu beachten ist, dass es in den betrachteten Klausuren ab 2020 aufgrund der Pandemie eine unbegrenzte Freiversuchsregelung gibt; man kann also spekulieren, dass Studierende seit dieser eventuell eher auch ohne Vorbereitung einen Versuch wagen, so dass es als Erfolg zu werten ist, dass sich die Durchfallquote nicht verschlechtert hat.

4.3 Weitere Veranstaltungen

Über Softwaretechnik 2 hinaus wird Divekit an der TH Köln noch in mehreren anderen Veranstaltungen eingesetzt (Softwaretechnik 1 und Algorithmik in Informatik (BA), Coding Essentials 1, Application Design und Microservice Architectures in Code & Context (BA)). Aus Platzgründen kann hier nicht im Detail darauf eingegangen werden, das Feedback der Studierenden ist aber ähnlich wie bei Softwaretechnik 2.

5 Bewertung und Ausblick

Die mit dem Einsatz von Divekit einhergehende Digitalisierung der Lehre hat in mehreren Softwaretechnik-nahen Modulen entscheidend dazu beigetragen, die Herausforderungen der Corona-Pandemie gut zu bewältigen. Nach übereinstimmendem Feedback von Studierenden und Lehrenden führt Divekit zu mehr Praxisnähe, Motivation und Effizienz in der Lehre. Daher wird das Framework in jedem Fall weiter entwickelt und eingesetzt werden, auch wenn irgendwann wieder "normale" Präsenzlehre durchgehend möglich sein wird.

Die Erfahrungen zeigen aber auch, dass eine technische Lösung allein nicht zu besserer Lehre führt. Ebenso wichtig sind ein ganzheitliches Konzept (wie etwa das konsequente Umsetzen von *Active Learning* mit Elementen des Flipped Classroom und Übungs-Workshops), kontinuierliche Ansprechbarkeit der Lehrenden über eine Messaging-Plattform wie etwa Discord, und Good Practices im Einsatz von Divekit (wie etwa ein geschicktes Zusammenspiel von automatisiertem und manuellem Feedback).

Viele Herausforderungen bleiben noch offen und bieten Raum für weitere Forschung und Entwicklung. Dazu zählt in erster Linie die Usability des Frameworks, sowohl für Studierende wie auch für Lehrende. Für Studierende ist das Feedback häufig noch zu kurz und kryptisch, da es ursprünglich aus den Messages fehlgeschlagener Unit-Tests stammt. Lehrende wiederum brauchen bessere Unterstützung beim Verfassen komplexer Aufgabenstellungen - die sprachliche und semantische Variationsvielfalt bei vielen Platzhaltern ist schwer zu beherrschen. Große zusammenhängende Aufgabenstellungen sind aber insbesondere bei Klausuren für Studierende hilfreich, weil sie den Stressfaktor von fachlichen Kontextwechseln minimieren.

Weitere Entwicklungsziele sind der Einsatz von Gamification zur weiteren Steigerung der intrinsischen Motivation, Nutzung von KI für komplexe semantische Fragen, und vieles mehr. Ein kontinuierliches Monitoring des Lernerfolgs bleibt ein zentrales Anliegen, um - gerade in hybriden Lehrformaten - wirklich zu wissen, ob die Studierenden erreicht werden und wo deren Schwierigkeiten liegen.

Divekit soll weiter in Richtung eines Ökosystems mit zahlreichen Nutzer:innen entwickelt werden. Das Framework ist als Open Source unter <https://github.com/divekit> veröffentlicht. Interessierte sind herzlich eingeladen, es zu nutzen und Verbesserungen, neue Features und Good Practices für den Einsatz beizusteuern.

Literatur

- [AB19] Anke, J.; Bente, S.: UML in der Hochschullehre: Eine kritische Reflexion. In: SEUH 2019. Feb. 2019.
- [Ar16] Armstrong, P.: Bloom's taxonomy, Vanderbilt University Center for Teaching, 2016.
- [Be02] Beck, K.: Test-driven development: by example. Addison Wesley, 2002.
- [BHS17] Breitner, J.; Hecker, M.; Snelting, G.: Der Grader Praktomat. In (Bott, O. J.; Fricke, P.; Priss, U.; Striwe, M., Hrsg.): Automatisierte Bewertung in der Programmierausbildung. Digitale Medien in der Hochschullehre 6, Waxmann Verlag GmbH, S. 159–172, 2017, URL: <https://www.waxmann.com/automatisiertebewertung/>.
- [DA21] DATACOM, S.: Konferenzprogramm OOP 2020, 2021, URL: <https://www.oop-konferenz.de/oop-2021/programm/konferenzprogramm>, Stand: 23. 12. 2021.
- [Ev04] Evans, E.: Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.
- [FB09] Felder, R. M.; Brent, R.: Active learning: An introduction. ASQ higher education brief 2/4, S. 1–5, 2009.
- [Fo19] Fowler, M.: Software Architecture Guide, Aug. 2019, URL: <https://martinfowler.com/architecture/>, Stand: 15. 10. 2021.
- [Go10] Gogvadze, G.: ActiveMath-generation and reuse of interactive exercises using domain reasoners and automated tutorial strategies, 2010.
- [He18] Helfrich-Schkarbanenko, A.; Rapedius, K.; Rutka, V.; Sommer, A.: Mathematische Aufgaben und Lösungen Automatisch Generieren. Springer, 2018.
- [HRL20] Hazzan, O.; Ragonis, N.; Lapidot, T.: Guide to Teaching Computer Science: An Activity-Based Approach. Springer Nature, 2020.
- [In21] Intveen, J.: Digitalisierung und Individualisierung der praxisorientierten Lehre von modernen Coding-Ansätzen, Magisterarb., Technische Hochschule Köln, Cologne Institute for Digital Ecosystems (CIDE), 2021.
- [KJH18] Keuning, H.; Jeuring, J.; Heeren, B.: A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education 19/1, 3:1–3:43, Sep. 2018, URL: <https://doi.org/10.1145/3231711>, Stand: 25. 10. 2021.
- [Ma00] Martin, R. C.: Design Principles and Patterns, 2000, URL: https://web.archive.org/web/20150906155800/http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf, Stand: 15. 10. 2021.

-
- [Ma09] Martin, R. C.: Clean code: a handbook of agile software craftsmanship. Pearson Education, 2009.
- [Pr21] Project, P. O. S.: PMD Source Code Analyzer, 2021, URL: <https://pmd.github.io/pmd-6.37.0/>, Stand: 25. 12. 2021.
- [RD00] Ryan, R. M.; Deci, E. L.: Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25/1, S. 54–67, 2000.
- [RMP04] Ridgway, J.; McCusker, S.; Pead, D.: Literature review of e-assessment. Futurelab, 2004.
- [VM] VMware, I.: Spring Data JPA, en, URL: <https://spring.io/projects/spring-data-jpa>, Stand: 15. 10. 2021.
- [Wa18] Wasik, S.; Antczak, M.; Badura, J.; Laskowski, A.; Sternal, T.: A Survey on Online Judge Systems and Their Applications. *ACM Computing Surveys (CSUR)* 51/1, S. 1–34, 2018, ISSN: 0360-0300; 1557-7341.
- [WK21] Woelk, F.; Kasch, H.: Code FREAK: Automatisches Feedback für die Programmierausbildung. *Die neue Hochschule/2021-5*, S. 28–31, Okt. 2021.

IT-REX — A Vision for a Gamified e-Learning Platform for the First Semesters of Computer Science Courses

Sandro Speth,¹ Steffen Becker,¹ Uwe Breitenbücher,¹ Philipp Fuchs,¹ Niklas Meißner,¹
Anna Riesch,¹ Daniel Wetzel¹

Abstract: Digital learning is becoming increasingly important, especially in situations when students cannot attend presence lectures as we experienced during the Covid-19 pandemic. However, while there are platforms that support generic learning concepts such as multiple-choice questions, we believe that the first semesters of computer science courses can benefit from tailored learning platforms that support IT-specific learning. For example, programming tasks that are automatically checked for correctness. Moreover, in times when students cannot meet each other, self-organization and motivation quickly become severe problems. Therefore, this paper presents a vision for a gamified e-Learning platform specialized for the first semesters of computer science courses.

Keywords: e-Learning; Gamification in education; Study Beginners; Computer Science

1 Introduction, Motivation, and Research Questions

Due to the Covid-19 pandemic and the resulting need for staying at home, digital teaching and asynchronous learning are becoming increasingly important since students have to learn from anywhere. Especially, in 2020 around 1.39 billion students stayed at home [UN]. While there are many e-Learning platforms available, there is currently no platform specialized in higher-level education for computer science courses of the first semesters. However, the first semester is crucial for study beginners because students often are not used to the vast amount of content to learn in only a short period. Moreover, especially this content creates the foundation for further semesters. While many e-learning platforms such as ILIAS or Moodle offer plugins for structuring content or writing and checking source code, lecturers often do not instrument them in their full capacity as their usability often lacks an easy-to-use interface. Additionally, many student beginners have challenges to be *intrinsically motivated* for learning new content regularly, especially if the tasks do not arouse interest which plays a vital role in the motivation of a learner [De97]. In particular, when lecture recordings are only uploaded without further structure, traditional platforms fail to keep students engaged in learning new content regularly instead of forcing themselves through the content just before the exam. Since intrinsic motivation is crucial to students who are responsible for their own learning [FKY18], an e-learning platform needs to *extrinsically motivate* students in this asynchronous setting which leads to our first research question:

¹ University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany {sandro.speth, steffen.becker, uwe.breitenbuecher, philipp.fuchs, niklas.meissner, anna.riesch, daniel.wetzel}@ms.informatik.uni-stuttgart.de

RQ1: *"How can an e-learning platform for asynchronous learning extrinsically motivate students to learn regularly new computer science content in the first semesters?"*

Furthermore, if students fail to intrinsically motivate themselves to repeat previously learned topics, they often do not practice content they do not master well. Instead, they often tend to concentrate only on learning new content without practising old content in full depth. As a result, such students often tend to forget the content they learned before. Therefore, we need to extrinsically motivate students to repeat old learning content regularly, concentrating on concepts they do not master well. This leads us to our second research question:

RQ2: *"How can an e-learning platform extrinsically motivate students to regularly repeat previously learned computer science content in the first semesters?"*

In this paper, we present a vision for an interactive e-learning platform tailored for the first semesters of computer science courses in order to keep them motivated and to support social interactions. The platform is organized around a gamification approach that aims to motivate students in learning, repeating, and practising learned knowledge while keeping them engaged in their studies and offering social interaction.

2 Vision for a Digital Learning Platform Tailored for the First Semesters of Computer Science Courses

In this section, we introduce our vision for the gamified e-learning platform *IT-REX*. Our vision is to motivate students by challenging them with tasks specialized for the individual content of computer science courses in the first semesters. Therefore, in *IT-REX*, students have to solve computer science-specific tasks to deepen their knowledge and experience in the respective content, e.g., programming tasks, diagram drawing tasks, etc. While identifying and defining appropriate tasks for realizing learning concepts and resolving open learning fields is a mandatory building block for our e-learning platform *IT-REX*, this does not necessarily motivate students to regularly learn and repeat new and old content. Therefore, we use gamification as second approach in combination with computer science-specific tasks, which is an emerging trend in education [Sa15] to motivate and engage students [KAY14; Mu11]. Since there has been a lot of research done on how *gamification in education* improves the quality of learning of students [Su14], in our opinion, combining course-specific learning with gamification approaches is an important aspect for extrinsic motivation. Furthermore, gamification supports students in their goal setting [BHH20], emphasizes learning, and improves knowledge understanding [Co13]. As a single type of task does not apply well for various different learning goals and concepts, *IT-REX* must be tailored towards different types of computer science learning concepts and goals. Thus, we must first derive the concepts and goals relevant to first-semester computer science courses.

Learning Concepts That Need to be Supported by the IT-REX Platform We analyzed several first semester lectures of different computer science study programs and found that the following four types of courses are essential in most programs: (i) programming, (ii) software engineering, (iii) theoretical computer science, and (iv) mathematics. Each of these types of courses consists of several learning concepts, e.g., inheritance in object orientation. Furthermore, each learning concept can have multiple learning goals, for example, “The student knows the concept and can explain it”, or “The student can apply the concept”. Depending on the concept and goal, different types of gamification have to be supported by the platform. Since there are already well-established platforms for mathematics, we will not look further into mathematics and concentrate on teaching programming, data structures, algorithms, software engineering, and theoretical computer science.

On an abstract level, all learning concepts of a regular programming introductory lecture relate to learning how to read, write, and explain source code. Moreover, typically basic programming paradigms necessary for this are introduced, source code quality is discussed, and modelling software is practised. To test the basic understanding of programming and program paradigms, it makes sense to create single-choice and multiple-choice quizzes, assignment questions, cloze texts, etc. They can be automatically checked and, therefore, offer fast feedback to the student. However, learning to write source code is more difficult as programming is only taught well through practice. Thus, we need a means to enable students to program and to check their code for functional correctness and code quality automatically. To automatically check the source code, we plan to integrate software such as ArTEMIS² by Krusche et al., which enables to check in source code via git and run unit tests on it for individual feedback to the student. In case the lecture focuses on data structures and algorithms, already many game-like challenges are available online, e.g., Hackerrank³, which can be used to improve the training in those concepts. We will integrate such tools in IT-REX. Besides understanding and writing source code, modelling software is also a crucial part of every introductory programming course. Thus, drawing diagrams and testing them automatically has to be supported by IT-REX, too. Moreover, theoretical computer science must be included, for example, by a task in which the students have to evaluate first-order logic formulas that are then automatically checked for correctness.

Gamification of Higher-level Computer Science Education in the First Semesters

Based on the identified types of tasks, learning concepts, and learning goals relevant to study beginners in computer science, we envision the following concept for gamification in our e-learning platform IT-REX. Gamification is the central aspect of the learning platform as it is intended to create an incentive for students to learn, thus, extrinsically motivating them. In our platform, comparable to a Tamagotchi in the late 90s, the students need to take care of an IT-REX, a dinosaur-like character. The IT-REX grows with increasing progress within a course and gives an impression of the overall learning progress by solving quizzes and other types of tests. Furthermore, students can compare their IT-REX with others resulting

² <https://github.com/lslintum/Artemis>

³ <https://www.hackerrank.com/>

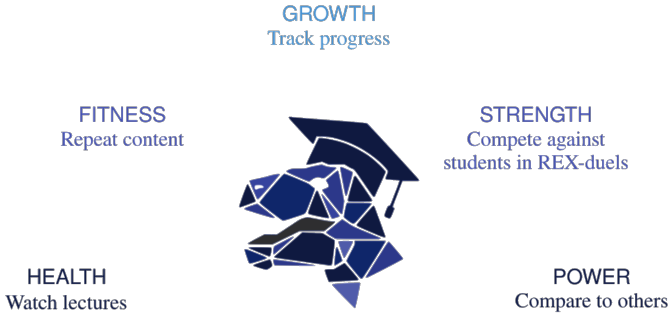


Fig. 1: IT-REX features and their objectives.

in a competition incentive. As tests, there are single-choice, multiple-choice, cloze text, arrangement questions, and more—as described in the previous Sect. 2. The lecturer must create the test questions and answers and link the test to its reflected learning goal. Students have to take tests to unlock the following content of the course. Furthermore, there are tests to repeat previous contents. By solving the tests, the students can get points for a scoreboard to compare with others. Similar to the language app Memrise, we will add a turbo mode in which the students have to solve as many tests as possible in a short period with a limited number of mistakes, thus, resulting in pressure and improving their skills under time which is also a good practice for exams. As some students do not feel well under pressure, the turbo mode is optional and, therefore, not needed to take care of an IT-REX. However, the points and benefits for the IT-REX are increased significantly in turbo mode to motivate even outstanding students to practice already learned course contents again. These concepts seem to be appropriate to motivate students for individual learning regularly.

Besides the comparison with other students on a scoreboard, a major gamification concept of the platform are *REX-Duels*. A REX-Duel is a game in which two or more students can compete against each other alone or as teams in terms of solving tasks. This duel is used for a more direct competition incentive and social interaction. It also includes the types of tasks explained in Sect. 2, which must be solved against each other in time. Furthermore, REX-Duels can contain programming tasks which are checked automatically through ArTEMIS. Through CodeTogether, the students can voice chat and collaborate on the same code base before pushing it to ArTEMIS. Accordingly, the IT-REX of the winner or winning teams receive a reward for the victory, but all students improve their knowledge.

Based on a students' progress, different features of their IT-REX representing the learning status of each student are improved. These features are shown conceptually in Fig. 1. In total, five features will be supported: (i) *Health* shows the current learning progress of a student. If the IT-REX is doing poorly, this is the indicator that the student is lagging behind. (ii) *Fitness* shows how well a student repeats previous chapters. For example, if the IT-REX is limp, the student has to repeat old chapters and “train” so that the fitness increases. Furthermore, tests that the student did not solve correctly are repeated more frequently. (iii) *Growth* serves as

a progress bar, so the students know how much of the course is still ahead of them. The IT-REX receives food with each new concept learned. Therefore, it grows and moves up in levels. An upper limit for the levels indicates the end of the course that the student is aiming for. (iv) *Strength* is used to enforce a certain level of interaction between the students and aims at motivating them to participate regularly in REX-Duels: By participating in REX-Duels, the IT-REX gets strength points that depict how a student's knowledge level is compared to others. While both participants increase their IT-REX' strength points in a REX-Duel, the winner will gain significantly more strength points which aims at motivating students to prepare for REX-Duels by learning regularly. (v) *Power* shows the composite value of the other properties so that a ranking of students can be created.

By solving tests correctly and winning REX-duels, the students additionally gain *coins* that can be used to buy things in a virtual shop, e.g., to change the appearance of their IT-REX by purchasing visual aspects such as costumes, accessories, and other elements.

3 Related Work

IT-REX combines various features of well-established e-learning platforms. For example, Udemy or Microsoft Learn courses offer their content structured in chapters or modules. The actual content can be videos, texts, or tasks. Microsoft Learn concludes each module with a learning level quiz that must be passed to complete the module — thus, these are concepts that of course make also sense to be supported by IT-REX. In particular, the gamification aspects in quiz applications are significant for the extrinsic motivation of students [Ra20]. Especially apps for learning foreign languages, such as Memrise or Lingodeer, often rely on gamification to motivate their learners. However, such language apps concentrate primarily on enforcing regular repetition through their gamification aspects instead of motivating also to learn new chapters. For example, in an earlier version of Memrise, learning and repeating vocabulary let an alien grow, similar to our IT-REX concept. Moreover, in Memrise each vocabulary has a plant which dries without regular repetition which motivates learner to do so. Incorrectly answered tasks must be repeated more quickly, which is of course also an appropriate concept to be integrated in IT-REX as this directly supports learners in precisely practicing content. As a result, many of these apps focus mainly on single dimensions, while IT-REX supports multiple dimensions of learning simultaneously. Furthermore, points to be achieved, increasing difficulty levels, and avatar constructions can also contribute to extrinsic motivation through gamification [Ra20], which has been shown by various apps. Therefore, in IT-REX, students can change the visual aspects of their IT-REX via coins.

4 Conclusions & Future Work

The presented ideas tailored for practising computer science content such as programming and data structure design could significantly help students to deepen their understanding.

Moreover, the presented ideas of gamification should support (i) regularly learning as the health status of the IT-REX, which represents the own learning progress, is directly shown on the platform. In addition, (ii) REX-Duels should also additionally improve interaction with other students. Both parts might help motivate students, which could result in better education. We plan to realize these concepts based on our already implemented basic platform for IT-REX⁴ in the future. Moreover, we plan a detailed evaluation on our first-semester programming course to design and improve the platform optimally.

References

- [BHH20] Bai, S.; Hew, K. F.; Huang, B.: Does gamification improve student learning outcome? Evidence from a meta-analysis and synthesis of qualitative data in educational contexts. In. Vol. 30, Elsevier, p. 100322, 2020.
- [Co13] Cook, W.: Five Reasons You Can't Ignore Gamification. Chief Learning Officer 12/5, pp. 46–55, 2013.
- [De97] Dev, P. C.: Intrinsic Motivation and Academic Achievement: What Does Their Relationship Imply for the Classroom Teacher? Remedial and special education 18/1, pp. 12–19, 1997.
- [FKY18] Firat, M.; Kılınc, H.; Yüzer, T. V.: Level of intrinsic motivation of distance education students in e-learning environments. Journal of Computer Assisted Learning 34/1, pp. 63–70, 2018.
- [KAY14] Kiryakova, G.; Angelova, N.; Yordanova, L.: Gamification in education. In. Proceedings of 9th International Balkan Education and Science Conference, 2014.
- [Mu11] Muntean, C. I.: Raising engagement in e-learning through gamification. In: Proc. 6th international conference on virtual learning ICVL. Vol. 1, pp. 323–329, 2011.
- [Ra20] Razali, N.; Nasir, N.; Ismail, M.; Sari, N.; Salleh, K.: Gamification Elements in Quizizz Applications: Evaluating the Impact on Intrinsic and Extrinsic Student's Motivation. In: IOP Conference Series: Materials Science and Engineering. Vol. 917. 1, IOP Publishing, p. 012024, 2020.
- [Sa15] Sandusky, S.: Gamification in Education. In. The University of Arizona, 2015.
- [Su14] Surendele, G.; Murwa, V.; Yun, H.-K.; Kim, Y. S.: The Role of Gamification in Education – A Literature Review. Contemporary Engineering Sciences 7/29, pp. 1609–1616, 2014.
- [UN] UNESCO: Education: From disruption to recovery, URL: <https://en.unesco.org/covid19/educationresponse>.

⁴ <https://github.com/IT-REX-Platform>

Software Engineering lehren (Teil 1)

Vorbereitung auf das wahre Leben — Herausforderungen bei Diskurs und Umgang mit Unsicherheit in der Lehre

Axel Böttcher,¹ Daniela Zehetmeier²

Abstract: Es ist keine Seltenheit, dass unsere Studierenden in ihrem späteren Berufsleben mit Software konfrontiert werden, die über lange Zeit gewachsen ist. Um später die damit verbundenen Herausforderungen meistern zu können, benötigen die Studierenden einerseits die Fähigkeit, mit Unsicherheit umzugehen. Denn Anforderungen sind oft unklar, oder es nicht immer möglich, unzureichend dokumentierte Altsysteme in kurzer Zeit soweit zu verstehen, dass Wartungsarbeiten oder Erweiterungen daran vorgenommen werden können. Andererseits bedarf es der Beurteilungsfähigkeit, um technische Realisierungen zu bewerten und kritische Diskurse zu Software mit Anderen führen zu können.

In diesem Beitrag beschreiben wir am Beispiel des Moduls Software-Archäologie unsere Erfahrungen, diese wichtigen Kompetenzen in der Lehre zu adressieren. Die Beobachtungen zeigen, dass für viele Studierenden das Führen von Diskursen ungewohnt ist und dass sie mit Unsicherheiten schlecht umgehen können. Wir schlussfolgern, dass weitere Methoden entwickelt werden müssen, um diese Kompetenzen zu lehren und zu messen.

Keywords: Höhere kognitive Fähigkeiten; Beurteilungsfähigkeit; Diskurs in der Lehre; Software Maintenance and Evolution; Nachhaltige Software

1 Motivation

Die Ausbildung unserer Studierenden in Softwareentwicklung und Software Engineering erfordert die Auseinandersetzung mit sehr vielen abstrakten Konzepten und formalen Sprachkonstrukten, die nahezu beliebig miteinander kombinierbar sind. Dies bringt uns in das klassische Dilemma von zu viel Stoff und zu wenig Zeit [Le11]. Wir haben beobachtet, dass wir deshalb nicht nur im Grundstudium dazu neigen, den Fokus der Lehre stark auf die Kompetenzebenen des Verstehens und Anwendens zu setzen.

Wir formulieren für die Lehrveranstaltungen immer wieder Lernziele auf höheren Kompetenzebenen gemäß der überarbeiteten Lernzieltaxonomie von Bloom [An01, Th15] und adressieren diese in Lehre und Prüfung [TBS16]. Dennoch bleiben aus unserer Sicht wichtige Entwicklungsschritte, die mit höheren Kompetenzebenen in Verbindung stehen, außen vor.

¹ Hochschule München, Fakultät für Informatik und Mathematik, Lothstraße 64, D-80335 München, axel.boettcher@hm.edu

² Lufthansa Aviation Training GmbH, Südallee 15, D-85356 München-Flughafen, daniela.zehetmeier@lat.dlh.de

Ein Beispiel einer solchen vernachlässigten Kompetenz ist die Beurteilungsfähigkeit. In der Praxis ist es essentiell, beurteilen zu können, welche der vielen Möglichkeiten, ein programmiertechnisches Problem zu lösen, sinnvoll sind und welche weniger sinnvoll. Wir unternahmen erste Versuche, diese Kompetenz bereits im Grundlagenmodul Softwareentwicklung zu fördern und in der Prüfung zu messen. Die zur Messung der Kompetenz entwickelte Prüfungsaufgabe erzielte mit einem Mittelwert von 23% der bei dieser Aufgabe erreichbaren Punkte den traurigen Rekord innerhalb dieser Prüfung. Diese Erfahrungen führten uns zu der Hypothese, dass wir in unseren Lehrveranstaltungen dem gemeinsamen Reflektieren der Entwicklungsprozesse und der Evaluation der daraus resultierenden Artefakte zu wenig Raum geben.

Des Weiteren ist die Fähigkeit, mit fremden („Legacy“) Projekten umzugehen, eine wichtige Kompetenz, die wir kaum adressieren. Viele unserer Lehrbeispiele und Praktikumsaufgaben in den ersten Semestern basieren auf glasklaren Anforderungen, um gerade keine Unklarheiten und Unsicherheiten aufkommen zu lassen. Das resultiert aus der Notwendigkeit, bei den großen Anfängerkohorten die Korrekturen und Bewertung effizient durchführen zu können. Außerdem versuchen wir, uns bei der Bewertung nicht angreifbar zu machen, indem wir den Raum für Diskussionen möglichst klein halten. Es ist aber keine Seltenheit, dass Software-Systeme lang leben und über Jahrzehnte hinweg genutzt werden [Du12]. Damit ist es also nicht unwahrscheinlich, dass unsere Studierenden in ihrem späteren Berufsleben mit der Aufgabe konfrontiert werden, Wartungsarbeiten an existierender Software vorzunehmen. Die Arbeit in einem derartigen, nicht mehr engmaschig qualitätsgesicherten Umfeld, bringt für Entwicklerinnen und Entwickler Unsicherheiten mit sich, mit denen sie umzugehen lernen müssen. Neben Erfahrung im Umgang mit Legacy-Projekten und im Lesen fremden Codes hilft dabei zielgerichtete Analyse und Diskussion.

Es mangelt also letztlich am Führen von kritischen Diskursen mit fundierten Argumenten im Hörsaal [HP18]. Wichtig hierbei ist es, dass die Studierenden den Diskurs als gewinnbringend für sich selbst empfinden, und einen Erkenntnisgewinn davon tragen können.

2 Zielsetzung

Aufgrund dieser Erkenntnisse wollen wir die Lehre anpassen und die genannten Defizite gezielt in vertiefenden Kursen adressieren. Die oben genannten Punkte lassen sich beispielsweise dadurch in die Lehre integrieren, dass wir die Studierenden mit einer umfangreichen existierenden Codebasis arbeiten lassen. Je weniger ein solches Projekt unter Einhaltung der klassischen Qualitätsstandards entwickelt wurde, desto mehr müssen wir dabei, wie die Archäologen es formulieren würden, „Erkenntnisse aus dem ziehen, was im Altertum von Menschenhand geschaffen worden ist“ [Si00]. Ausgehend von einer solchen metaphorischen Anspielung an die Altertumswissenschaften wurde bereits 2002 von Hunt und Thomas der Begriff *Software-Archäologie* vorgeschlagen [HT02].

In diesem Beitrag beschreiben wir am Beispiel eines Moduls Software-Archäologie unsere Erfahrungen damit, diese wichtigen Themen im Bereich des Software Engineering bzw. der Software Maintenance bzw. Evolution in der Lehre zu adressieren, die aus unserer Sicht oft außen vor bleiben. Es soll ein Rahmen geschaffen werden, für gemeinsame Diskurse auf Augenhöhe mit den Studierenden, zum Inhalt und zur Qualität fremder Software.

Übergeordnetes Ziel des Moduls ist es auch, die Studierenden für die Wertschätzung von bestehender Software zu sensibilisieren. Beispielsweise ist die Auswahl einer Programmiersprache und damit verbundener Frameworks und Teststrategien immer im Kontext des Entwicklungszeitpunkts zu sehen. Berücksichtigt werden müssen immer auch Tatsachen wie etwa, dass viele Programmierer:innen Quereinsteiger aus unterschiedlichen Bereichen waren, wie der Elektrotechnik oder Physik. Und dass damit Anwendung von Prinzipien wie Modularisierung, Separation of Concerns etc. oft vernachlässigt wurden.

3 Literatur

Software Maintenance wird durchaus in Standard-Curricula erwähnt. Die GI-Empfehlungen für Bachelor- und Masterprogramme der Informatik [Zu16] erwähnen Folgendes als Teil der Realisierungskompetenz: „Für die Wartung und Erweiterung von Software ist die Fähigkeit notwendig, sich in vorhandenen Quelltext einzuarbeiten und diesen sinnvoll weiter zu entwickeln“.

Software-Wartung als Teildisziplin des Software Engineering wird selten als eigenes Modul, wenn überhaupt, dann als ein Bestandteil im Rahmen einer Lehrveranstaltung zu Software Engineering unterrichtet.

Zur Vernachlässigung des Lesens fremden Quellcodes stellten bereits 2003 van Deursen et al. in einem Workshop-Beitrag für *Program Comprehension* fest: „Students learn how to write new programs but they are not taught how to read and change existing and large ones“ [De03].

Auf der didaktischen Seite analysiert Bower Hörsaal-Diskurse mit Methoden der strukturierten Inhaltsanalyse [Bo09]. Bower fasst in seiner Arbeit jedes Zwiegespräch in der Lehre zur Softwareentwicklung als Diskurs auf. Wir verstehen hier unter Diskurs nur die kritische Auseinandersetzung mit den Artefakten und ihren Entstehungsprozessen.

Smith wie auch Pinto verfolgen den Ansatz, Maintenance und Evolution am existierenden Open-Source-Projekt zu unterrichten [Sm14, Pi17]. Die Bedeutung von kritischem Diskurs wird dabei nicht erwähnt. Wir erweitern dieses Feld zur Software-Archäologie um eine retrospektive Architektur- und Anforderungsanalyse. Dabei fokussieren wir in der Lehre auch die kritische Diskussion der Artefakte und die Beurteilung der Analysemethoden.

4 Kursdesign

Das Modul Software-Archäologie ist als Wahlfach mit fünf ECTS für die Bachelor-Studiengänge Informatik und Wirtschaftsinformatik konzipiert. Es ist für je zwei Semesterwochenstunden seminaristischem Unterricht sowie Praktikum konzipiert. Als Prüfungsform ist eine zweigeteilte Prüfung, bestehend aus einer benoteten Studienarbeit sowie einer benoteten mündlichen Prüfung vorgesehen.

Corona-bedingt musste der Kurs im Sommersemester 2021 komplett online stattfinden. Wir hatten die Möglichkeit das Modul im Pair-Teaching zu unterrichten [ZBBK18]. Dieses Setting ermöglichte das Vorleben von Diskursen durch Diskussionen zwischen den Dozierenden.

4.1 Lernziele des Moduls

Folgende Lernziele formulierten wir ausgehend von den Kompetenzen, die aus unserer Sicht für die gegebenen Aufgaben erforderlich sind.

- Sie analysieren existierenden Quelltext, um ihn zu verstehen, und um
 - Rückschlüsse auf die Intention der ursprünglichen Entwickler:innen zu ziehen
 - Requirements zu identifizieren, sodass diese als Grundlage für Refactorings oder eine Re-Implementierung dienen können
- Sie dokumentieren die gewonnen Erkenntnisse mit geeigneten Mitteln.
- Sie wenden Techniken des Reverse Engineering systematisch und gezielt an.
- Sie wenden Techniken des Refactoring systematisch und gezielt an.
- Sie analysieren Kontrollfluss theoretisch oder anhand existierender Ausgangsdaten.
- Sie entwerfen und implementieren Testinfrastruktur für Legacy Code und führen diese aus.
- Sie diskutieren Vorgehensweisen und Arbeitsergebnisse in Ihrer Praktikumsgruppe und im Plenum.

4.2 Projekt im Modul

Im Zentrum der Lehrveranstaltung sollte ein existierendes Projekt stehen, an dem entlang sich eine inhaltliche Auseinandersetzung innerhalb eines Semesters orientieren kann. Ähnlich wie Smith et al. [Sm14] formulierten wir vorab Kriterien an ein solches Projekt:

- Das Projekt muss hinreichend groß und komplex sein, sodass keine Gruppe von Studierenden versucht, eine komplette Reimplementierung „am Wochenende“ vorzunehmen.
- Das Projekt muss technische Schulden aufweisen.
- Das Projekt muss verschiedene Schnittstellen bedienen.

- Das Projekt sollte nicht aus einem akademischen Kontext stammen.

Insofern kämen prinzipiell Projekte aus dem Open-Source-Umfeld in Betracht, ebenso wie Projekte aus der Praxis [Pi17]. Anders als z. B. in [Pi17] wollten wir gerade keine Open Source Community, die als Fallback bei Schwierigkeiten an uns vorbei kontaktiert werden hätte können.

Ähnlich wie von Smith et al. [Sm14] erwähnt, hatte auch unsere Suche nach einem geeigneten Projekt viel Zeit verschlungen und war zunächst erfolglos geblieben. Mit dem Wechsel der Autorin in die Industrie eröffnete sich die Möglichkeit, auf mehrere Projekte aus dem beruflichen Kontext zuzugreifen, welche die Anforderungen erfüllen. Die Auswahlentscheidung fiel auf ein historisch gewachsenes, ca. zwölf Jahre altes, in ColdFusion [Na10] realisiertes Projekt. Die Software ist nach wie vor im produktiven Einsatz. Der Umfang beträgt etwa 30.000 Codezeilen. Das System bedient mehrere Schnittstellen, ist weitgehend undokumentiert, verfügt über keinerlei Tests und bringt diverse weitere technische Schulden mit sich.

4.3 Themen und Aufgabenstellungen

In den ersten beiden Dritteln des Semesters sollte die bestehende Software analysiert werden mit dem Ziel, eine am arc42-Template [SH16] orientierte Architekturdokumentation zu erstellen. Der letzte Schritt besteht dabei in einer Beschreibung von Anforderungen, die sich ex-post aus der Software extrahieren lassen. Neben einer Unterstützung von Wartungsarbeiten kann die Dokumentation so eine Grundlage für die anstehende Reimplementierung des Systems bieten. Deshalb war dann im letzten Teil ein modernes User Interface auf Basis der Anforderungen zu entwerfen, sodass der Nutzen der archäologischen Arbeit sichtbar wird.

Die thematische Struktur umfasste folgende Themenblöcke:

Glossar Die erste Aufgabenstellung umfasste neben der Einrichtung des Projekts das Anlegen eines Glossars. Die Vorgabe war, das Glossar fortlaufend zu erweitern, um die im Laufe der Zeit jeweils hinzugewonnenen Erkenntnisse.

Extraktion einer API-Dokumentation Diese Aufgabe verlangte die Erstellung einer Dokumentation der (pre-REST) HTTP-API des Projektes einschließlich einer Beschreibung der gewählten Vorgehensweise in einem Wiki. Die konkrete Gestaltung der Dokumentation war den Studierenden überlassen.

Schnittstellenbeschreibung und Querschnittsthemen Das arc42-Template anzulegen und mit Glossar sowie API-Dokumentation zu füllen war der dritte Arbeitsauftrag. Darüber hinaus mussten Stakeholder, Randbedingungen und Kontext identifiziert, sowie querschnittliche Konzepte und externe Schnittstellen des Systems „soweit möglich“ beschrieben werden.

Dokumentation der Datenbank In diesem Schritt sollten die Beziehungen zwischen den existierenden Datenbanktabellen durch Reverse Engineering analysiert, beschrieben und evaluiert werden.

Laufzeitsicht Nachdem die Studierenden sich ausführlich mit der statischen Sicht auf die Applikation befasst hatten, sollten sie im nächsten Schritt dynamische Sichten erfassen. Einzelne API-Backend-Funktion waren in Aktivitätsdiagramme mit wesentlicher Abstraktion zu überführen.

Dokumentation der Anforderungen Eine ex-post-Extraktion der Anforderungen war als Grundlage für eine Reimplementierung der Anwendung gedacht.

GUI Am Ende des Semesters sollte die Studierenden das Wissen das sie über die Applikation und den fachlichen Kontext erworben haben in ein Re-Design des User Interface fließen lassen.

Zu jedem Thema wurde ein Aufgabenblatt ausgegeben. Die Aufgaben waren in Vierer-Teams zu bearbeiten. Ergebnisse wurden im Plenum vorgestellt und gemeinsam diskutiert.

5 Beobachtungen und Reflexion

Uns war bei Konzeption der Lehrveranstaltung wichtig, dass das Projekt viele Unsicherheiten beinhaltet. Die historisch gewachsene Software konfrontierte auch uns Dozierende mit einer gewissen Unsicherheit, denn der komplette Funktionsumfang der Applikation hatte sich auch uns bis zum Ende der Veranstaltung nicht vollständig erschlossen.

Während der Vorbereitung der einzelnen Lehrveranstaltungsstunden, aber auch in der Retrospektive, diskutierten wir immer wieder verschiedene Aussagen, Arbeitsergebnisse, aber auch Verhaltensweisen von Studierenden. Drei wesentliche Erkenntnisse scheinen uns an dieser Stelle einer besonderen Erwähnung wert: inhaltliche Diskussionen, die sich in der Lehrsituation ergeben, Aufgabenstellungen, die allein durch den nicht vollständig erschlossenen Kontext vage sind, sowie der Umgang mit der daraus resultierenden Unsicherheit.

5.1 Diskussionen und Raum für Fragen

Wir Lehrenden sahen uns in dieser Veranstaltung als Coaches, welche die Studierenden in der Entwicklung mit theoretischem Input und Diskussionen zu auftretenden Fragestellungen auf Augenhöhe unterstützen. Zusätzlich präsentierten wir Prozesse der Erkenntnisgewinnung für ausgewählte Aspekte im Unterricht. Wir planten explizit Raum für Diskussionen ein und standen in wöchentlich zwei Praktikumsstunden für Fragen zur Verfügung.

Dennoch war dieser Ansatz nicht ausreichend, denn die Arbeitsergebnisse legen nahe, dass die Studierenden aus der Diskussion keine Erkenntnisse gewinnen oder diese nicht transferieren konnten. Gründe hierfür könnten sein, dass die Studierenden nicht die Notwendigkeit von Diskussionen sehen und auch nicht die notwendigen Voraussetzungen mitbringen, um Diskussionen auf geeignetem Abstraktionsniveau zu führen oder sich daran zu beteiligen. Dadurch können sie dann auch keinen Erkenntnisgewinn daraus erlangen. Manche Studierende beurteilen in der Evaluierung vorgelebte Rückfragen im Diskurs als Inkompetenz:

*Es kommt sehr schlecht rüber, wenn es so aussieht wie wenn
Dozent X Hilfestunde für Dozent Y hält.*

Für uns bleibt hier die Frage, wie wir mit dieser Erkenntnis in Zukunft umgehen. Wie schaffen wir die nötigen Voraussetzungen zum Führen von Diskursen?

5.2 Umgang mit quantitativ vagen Aufgabenstellungen

Die Aufgabenstellungen waren insofern quantitativ vage formuliert als wir Dozierende für viele Fragen selbst auf Hypothesenbildung und nachträgliche Überprüfung in einer kritischen Diskussion angewiesen waren.

Beispiel: „Beschreiben Sie externe Schnittstellen soweit möglich.“ (als Aufgabenteil im Rahmen der Architekturdokumentation mit arc42). Wir konnten hier selbst keine genaue Aussage hinsichtlich der Anzahl an zu identifizierenden Schnittstellen machen. Allein das Fehlen einer quantitativen Aussage, sorgt unter den Studierenden für ein Gefühl von Unsicherheit. Daher forderten die Studierenden immer wieder quantitative Aussagen, wie viel sie zum Bestehen des Moduls machen müssen.

Eine Bewertung muss am Ende auch die Vorgehensweise berücksichtigen. Diese haben wir uns immer wieder beschreiben lassen. Wir haben uns an dem gängigen Bewertungssystem orientiert, nachdem eine Leistung eine Note 2 verdient, wenn sie voll den Anforderungen entspricht – also gemessen an dem, was in der Aufgabe von den Dozierenden gefordert war. Bei einer 1 muss die Leistung den Anforderungen in besonderem Maße und bei einer 3 im allgemeinen entsprechen. Wir mussten dabei keiner der Gruppen Mängel (Note 4) bescheinigen.

Im Laufe des Semesters stellten wir uns immer wieder die Frage: ob wir selbst das Projekt besser kennen müssten, um solche quantitativen Aussagen zu treffen. Wir kamen wiederholt zu dem Schluss, dass genau der fehlende Wissensvorsprung die Realitätsnähe in der Lehrveranstaltung herstellt. In der Evaluation am Semesterende zeigte sich ein sehr heterogenes Bild unter den Studierenden in Bezug auf ihre Fähigkeit zum Umgang mit den auftretenden Unsicherheiten:

„Die Lehrveranstaltung ist sehr realitätsnah, wodurch meiner Meinung nach, die Relevanz der Inhalte gezeigt wird. Ich persönlich finde genau deswegen die VL sehr interessant.“

„Die Aufgaben waren nicht gut gestellt und es gab viele Diskussionen, was genau gemacht werden musste.“

Wie schaffen wir trotz naturgemäß vager Arbeitsaufträge genügend Sicherheit für die Studierenden, sodass sie nicht zusätzlich in einer dauernden Unsicherheit hinsichtlich ihrer zu erwartenden Noten sind?

5.3 Einfluss der Unsicherheit auf die Qualität

Die Arbeitsergebnisse des Semesters zeigen, dass mit zunehmender Unsicherheit auch die Qualität in einigen Gruppen sinkt. Eine Gegenüberstellung der Aufgabenstellungen, deren Unsicherheit und die Qualität der Arbeitsergebnisse bekräftigt diese Beobachtung:

Aufgabe Glossar: hoher Grad an Unsicherheit, schlechte Gesamtbewertung. Zu Beginn des Semesters bot der Auftrag, ein Glossar anzulegen noch eine große Sicherheit für die Studierenden: Einige Begriffe aus dem Kontext des Projekts wurden in der Lehrveranstaltung ausführlich besprochen. Aber die Anzahl der expliziten Hinweise, Begriffe in das Glossar aufzunehmen, verringerte sich zügig. Studierende unterschätzten die Bedeutung von Begriffen aus dem fachlichen Kontext der Anwendung und erweiterten das Glossar kaum mehr. Die Begriffe waren natürlich auch in späteren Teilen der Lehrveranstaltung von Bedeutung und somit war die Orientierung in der Fachlichkeit mit zunehmender Unsicherheit behaftet, allein durch die Vernachlässigung des Aufschreibens. Alles in allem waren die entstandenen Glossare von minderer Qualität. Die Studierenden differenzierten unzureichend, welche Begriffe wichtig sind, viele Ergebnisse waren lediglich Abkürzungsverzeichnisse.

API Dokumentation: niedriger Grad an Unsicherheit, gute Gesamtbewertung. Die automatische Erstellung der API-Dokumentation war für die Studierenden mit wenig Unsicherheit verbunden. Sie wussten, welche Quelltextdateien sie analysieren sollten und welche Informationen zu extrahieren waren. Zusätzlich machten wir einen Vorschlag zur Darstellung der Informationen. Wir überließen es der Kreativität der Studierten, in welcher Sprache sie den Vorgang automatisieren und wie sie die gewonnenen Informationen aufbereiten. Die Studierenden mussten also nicht unterscheiden, welche Informationen wichtig und welche unwichtig sind, die Aufgabe bot ein hohes Maß an Sicherheit. Die Ergebnisse der Studierenden waren durchwegs sehr gut zu bewerten.

Schnittstellen: mittlerer Grad an Unsicherheit, mittlere Gesamtbewertung. Eine Teilaufgabe der arc42-Dokumentation war die Identifikation und Beschreibung der externen Schnittstellen. Hier machten wir keine quantitative Aussagen, wie viele

Schnittstellen vorhanden sind und identifiziert werden können. Die Studierenden hatten also ein gewisses Maß an Unsicherheit, da sie selbst entscheiden mussten, wann sie ihre Recherchen beenden. Die Gesamtbewertung der Teilaufgabe fiel im mittleren Bereich aus. Wir vermuten, dass die positive Tendenz sich durch die guten Suchmechanismen und die Verwendung von standardisierten Schnittstellen (z.B. `http-Requests`) erklären lässt.

Datenbank: mittlerer Grad an Unsicherheit, mittlere Gesamtbewertung. Bei der Beschreibung der Datenbankstruktur ist die Teilaufgabe zur Extraktion der Tabellen und Felder mit einem geringen Grad an Unsicherheit zu bewerten. Diese Informationen können mit einem üblichen Werkzeug automatisiert gewonnen werden. Alle Gruppen konnten diese Aufgabe mit einem guten Ergebnis absolvieren. Aufgrund mangelhaften Datenbankdesigns mussten die impliziten Primärschlüssel-Beziehungen aus dem Code retrospektiv herausgearbeitet werden. Dies bedeutete für die Studierenden ein hohes Maß an Unsicherheit, weshalb die Aufgabe im ganzen auch einen mittleren Grad an Unsicherheit aufweist. Nur wenige Gruppen fanden nachvollziehbare und stimmige Beziehungen. Wohl deshalb lagen die Leistungen dieser Aufgabe im mittleren Bereich.

Laufzeitsicht: hoher Grad an Unsicherheit, schlechte Gesamtbewertung. Hier steigt die Unsicherheit auf einen hohen Grad an, da die Studierenden die wichtigen von den unwichtigen Teilen der Funktion unterscheiden müssen und auch Erklärungen für zahlreiche *magic numbers* finden sollten. Diese Abstraktion und der Umgang der damit verbundenen Unsicherheit fiel den Studierenden sichtlich schwer. Großteils ähnelten die Aktivitätsdiagramme eins-zu-eins den Codezeilen im Sinne einer Nacherzählung. Nur einige wenige Gruppen beschrieben die Essenz der Funktion und fanden Erklärungen für die *magic numbers*. Damit ist die Gesamtbewertung der Ergebnisse als schlecht einzuordnen.

Anforderungen: hoher Grad an Unsicherheit, schlechte Gesamtbewertung. Ähnlich wie bei den Aktivitätsdiagrammen hatten die Studierenden große Probleme bei der Beschreibung der Anforderungen an ihre zuvor analysierten Funktionen. Zu 'raten', welche Anforderungen Grundlage für die Funktionen sind und diese im Kontext zu beschreiben fiel allen Gruppen sichtlich schwer. Die Anzahl der Rückfragen waren während der Bearbeitungszeit auch auf dem Höhepunkt. Trotz zahlreicher Diskussionen und Unterstützung durch die Dozierenden konnte die Aufgabe nur mit einem eher schlechten Gesamtergebnis bewertet werden.

UI Design: niedriger Grad an Unsicherheit, gute Gesamtbewertung. Diese Aufgabe weist einen niedrigen Grad an Unsicherheit auf, da der Funktionsumfang erhalten bleiben soll und Hinweise gegeben wurden. Alles in allem modernisierten alle Gruppen das Design und machten die Nutzung der Funktionen komfortabel und benutzerfreundlich. Das Gesamtergebnis kann damit als sehr gut bis gut bewertet werden.

Die Evaluationsergebnisse weisen auf die vorhandene Verunsicherung von Studierenden hin:

„Projekt aus der Realität, auch wenn es nicht schön ist mit CF zu arbeiten“

„[...] das Projekt mit Lufthansa hat sich irgendwie nur bedingt geeignet, da man viele Herausforderungen/Aufgabenstellungen nur mir Raten lösen konnte“

„Der unübersichtliche Code und schlechter Codestil machen das analysieren des Projekts umständlich. Das mag eine akkurate Repräsentation der Wirklichkeit sein, motiviert aber nicht zum Ausarbeiten der Praktikumsaufgaben über das Minimum hinaus.“

Unter den Studierenden, auch der höheren Semester, zeigt sich ein sehr heterogenes Spektrum. Die einen können mit der Unsicherheit der Aufgabenstellung umgehen – die anderen nicht. Folglich müssen wir als Dozierenden den Umgang mit Unsicherheit gezielt trainieren. Pair-Teaching ist aus unserer Sicht ein geeignetes Mittel um Diskurs vorzuleben und zu transportieren [ZBBK18, BUM09]. Doch es bleibt die Frage, wie die Kompetenz noch zusätzlich unterrichtet werden kann, wo es anscheinend nicht ausreicht, Raum für Diskussionen zu bieten und diese zu veranschaulichen.

Trainiert man den Umgang mit Unsicherheiten und den Diskurs gezielt in der Lehre, sollten diese Kompetenzen nach dem Prinzip des Constructive Alignment [Bi96] auch in adäquater Form geprüft werden. Doch wie bewertet man den Umgang mit Unsicherheiten oder das Führen von Diskursen? Welche Kriterien gibt es, um die Kompetenz objektiv zu messen und die Bewertung transparent zu kommunizieren?

6 Diskussion und Ausblick

Wir haben in diesem Beitrag unsere Beobachtungen bei der Adressierung von Beurteilungsfähigkeit und dem Umgang mit Unsicherheiten am Beispiel der Veranstaltung Software-Archäologie vorgestellt.

Das Führen von Diskursen im (virtuellen) Hörsaal war unser Mittel, die Punkte in die Lehre zu integrieren. Diskurs braucht die Möglichkeit, unterschiedliche Standpunkte einnehmen zu können, von denen viele eine Berechtigung haben. Eine wichtige Erfahrung ist, dass ein gewachsenes umfangreiches und ohne klare Qualitätsstandards entwickeltes Fremdprojekt, gute Voraussetzungen dazu bietet. Die Firmenkooperation hat sich dafür als hilfreich erwiesen: das Industrie-Projekt kommt nicht aus des Professors „krauser Gedankenwelt“, wodurch gegenüber den Studierenden eine gefühlte Legitimierung, Glaubhaftigkeit und Authentizität hergestellt wird.

Trotzdem ist es schwierig, die Studierenden für einen Diskurs zu gewinnen. Unsere Studierenden drängen eher auf eindeutige Antworten oder Prozessbeschreibungen, die sie für die Prüfung verinnerlichen oder in der Studienarbeit anwenden können. Bieten

wir ihnen das nicht, fallen sie in eine große Unsicherheit. Hier entsteht ein Dilemma für die Dozierenden: durch zu viel und zu früh gegebenes Feedback passen sich die entstehenden Arbeitsergebnisse der Gedankenwelt der Dozierenden an – ein Resultat, das gerade vermieden werden soll. Das aus Studierendensicht „verweigerter“ Feedback nimmt ihnen gefühlt die Möglichkeit, ihre Noten im Verlauf des Semesters verbessern zu können.

Ermutigung, eingeschlagene Wege weiter zu verfolgen, Wertschätzung für Arbeitsergebnisse und die Diskussion der Vor- und Nachteile von Vorgehensweisen und Artefakten hilft, mit Unsicherheit im Projekt umzugehen. Sicherheit künstlich herzustellen fördert demgegenüber nicht die Fähigkeit, Unsicherheiten zu auszuhalten und damit auch im zukünftigen Berufsleben umgehen zu können.

Die Unterstützung der Studierenden in einem solchen Kurs, in dem höhere kognitive Fähigkeiten adressiert werden, übersteigt das Zeit-Budget, das für eine Lehrveranstaltung vorgesehen ist, erheblich. Hier fehlen effiziente Unterstützungsformen oder die Möglichkeit einer flexiblen Verrechnung von Arbeitsleistung hinsichtlich der Erfüllung der Lehrverpflichtung.

Zusammenfassend ergeben sich aus diesem Beitrag mehrere Fragen: Wie können wir Diskurs fördern, wie schaffen wir die nötigen Voraussetzungen zum Führen von Diskursen zu Software im Hörsaal? Welche weiteren Methoden eignen sich, um den Umgang mit Unsicherheit in die Lehre zu integrieren? Und: wie können wir diese Fähigkeiten objektiv messen, sodass die Messkriterien kommuniziert werden können? Dadurch könnte ein Faktor der Unsicherheit, nämlich die Unsicherheit bzgl. der Prüfungsleistungen verringert werden. Die Studierenden könnten sich dann auf die Unsicherheit fokussieren, die der Projektkontext mit sich bringt.

Dank

Die Autorin/der Autor danken der Lufthansa Aviation Training für das zur Verfügung gestellte Projekt einschließlich aller Quelltexte.

Literaturverzeichnis

- [An01] Anderson, Lorin W.; Krathwohl, David R.; Airasian, Peter W.; Cruikshank, Kathleen A.; Mayer, Richard E.; Pintrich, Paul R.; Raths, James; Wittrock, Merlin C., Hrsg. A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Longman, New York, 1. Auflage, 2001.
- [Bi96] Biggs, John: Enhancing teaching through constructive alignment. Higher education, 32(3):347–364, 1996.
- [Bo09] Bower, Matt: Discourse analysis of teaching computing online. Computer Science Education, 19(2):69–92, 2009.

- [BUM09] Böttcher, Axel; Utesch, Matthias; Moore, Austin: Erfahrungen mit Pair-Teaching für Software Engineering: Kooperation von Hochschule und Industrie. In: Tagungsband Software Engineering im Unterricht der Hochschulen (SEUH), Februar 2009 in Hannover. S. 5–15, 2009.
- [De03] van Deursen, A.; Favre, J.-M.; Koschke, R.; Rilling, J.: Experiences in teaching software evolution and program comprehension. In: 11th IEEE International Workshop on Program Comprehension, 2003. S. 283–284, 2003.
- [Du12] Durdik, Zoya; Klatt, Benjamin; Koziolok, Heiko; Krogmann, Klaus; Stammel, Johannes; Weiss, Roland: Sustainability guidelines for long-living software systems. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM). S. 517–526, 2012.
- [HP18] Hartung, M. J.; Pausch, R.: Soll man Studenten zwingen, im Hörsaal zu sitzen? Die Zeit, 73(2):61, 2018.
- [HT02] Hunt, Andy; Thomas, Dave: Software archaeology. IEEE Software, 19(2):20–22, 2002.
- [Le11] Lehner, Martin: Viel Stoff - wenig Zeit: Wege aus der Vollständigkeitsfalle. Haupt, Bern; Stuttgart; Wien, 2011.
- [Na10] Nadel, Ben: Adobe ColdFusion Anthology: Clear and Concise Concepts from the Fusion Authority. Apress, Berkeley, CA, 2010.
- [Pi17] Pinto, Gustavo Henrique Lima; Filho, Fernando Figueira; Steinmacher, Igor; Gerosa, Marco Aurelio: Training Software Engineers Using Open-Source Software: The Professors' Perspective. In: 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T). S. 117–121, 2017.
- [SH16] Starke, Gernot; Hruschka, Peter: arc42 in Aktion: Praktische Tipps zur Architekturdokumentation. Hanser, München, 2016.
- [Si00] Sinn, Ulrich: Einführung in die klassische Archäologie. C. H. Beck, 2000.
- [Sm14] Smith, Thérèse Mary; McCartney, Robert; Gokhale, Swapna S.; Kaczmarczyk, Lisa C.: Selecting Open Source Software Projects to Teach Software Engineering. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education. SIGCSE '14, Association for Computing Machinery, New York, NY, USA, S. 397–402, 2014.
- [TBS16] Thurner, Veronika; Böttcher, Axel; Schlierkamp, Katrin: Aligning learning objectives and exams: Moving upwards on the expertise level stack. In: IEEE Global Engineering Education Conference (EDUCON). S. 455–462, 2016.
- [Th15] Thurner, Veronika; Böttcher, Axel; Schlierkamp, Katrin; Zehetmeier, Daniela: Lernziele für die Kompetenzentwicklung auf höheren Taxonomiestufen. In: Tagungsband Software Engineering im Unterricht der Hochschulen (SEUH), Dresden. S. 9–20, 2015.
- [ZBBK18] Zehetmeier, Daniela; Böttcher, Axel; Brüggemann-Klein, Anne: Designing Lectures as a Team and Teaching in Pairs. In: Proc. 4th International Conference on Higher Education Advances (HEAd'18), Valencia. S. 873–880, 2018.
- [Zu16] Zukunft, Olaf: , Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (Juli 2016), 2016.

Online Teamteaching als Inverted Classroom – Erfahrungen aus den Pandemie-Semestern

Jens Liebehenschel,¹ Martin Simon¹

Abstract: Bereits fünf Mal wurde das Modul „Algorithmen und Datenstrukturen“ der Informatik Bachelor-Studiengänge an der Frankfurt University of Applied Sciences als Großveranstaltung im Teamteaching angeboten. Es wird kontinuierlich weiterentwickelt, sowohl inhaltlich als auch didaktisch. In den beiden Sommersemestern 2020 und 2021 wurde, bedingt durch die COVID-19-Pandemie, die Veranstaltung im bewährten Teamteaching-Format, jedoch online und mit Elementen eines Inverted Classroom angeboten. Die Erfahrungen waren teilweise für die Lehrenden überraschend. In der vorliegenden Arbeit werden insbesondere diese beiden Pandemie-Semester im Detail analysiert und diskutiert. Ausblicken werden wir auf eine für den nächsten Zyklus geplante wesentliche Veränderung des Lehrformats, mit dem Ziel, der lernrelevanten Diversität der Studierenden noch besser gerecht zu werden. Das geplante Format ist sowohl in der Präsenz- als auch in der Online-Lehre einsetzbar.

Keywords: Teamteaching; Inverted Classroom; Online-Lehre unter Pandemiebedingungen; Differenzierung durch Zusatzmaterialien; Differenzierung durch parallele Veranstaltungen

1 Rahmenbedingungen und Überblick

Auch wenn Teamteaching kein neues Konzept ist, wird es an Hochschulen relativ selten eingesetzt. Dennoch gibt es Erfahrungen, zum Beispiel in [KR16; Le06; WMW06]. Einer der Autoren des vorliegenden Artikels verfügt über mehrjährige Erfahrung im Teamteaching, von der Einführung im hier betrachteten Modul [LS17] bis hin zum Einsatz unter Pandemiebedingungen im Online-Format.

Das Modul „Algorithmen und Datenstrukturen“ ist für das zweite Semester der Informatik Bachelor-Studiengänge an der Frankfurt University of Applied Sciences vorgesehen. Es wurde im Sommersemester 2016 erstmalig als Großveranstaltung im Teamteaching angeboten. Darüber wird in [LS17] berichtet. Das ursprüngliche Ziel war, durch ein innovatives Lehrformat den Studierenden in einem geeigneten Modul das Lernen zu erleichtern. Dieses Modul wurde bewusst ausgewählt, weil es von den Studierenden als „schwierig“ eingestuft wird. Dies lässt sich auch anhand objektiver Kriterien wie beispielsweise höherer Durchfallquoten und einem relativ hohen Prozentsatz Studierender, die diese Klausur „vor sich herschieben“, feststellen.

¹ Frankfurt University of Applied Sciences, FB2–Informatik, Nibelungenplatz 1, 60318 Frankfurt, Deutschland, [liebehenschel,martin.simon]@fb2.fra-uas.de

Nachdem das Modul „Algorithmen und Datenstrukturen“ im Sommersemester 2016 erstmalig als Großveranstaltung im Teamteaching angeboten wurde, hat sich dieses Format bei Studierenden und Lehrenden etabliert. Die Veranstaltung wurde seitdem, mit einer Ausnahme, immer so durchgeführt. Mittlerweile besteht das Team aus vier Lehrenden, von denen jeweils zwei die Veranstaltung mit gleichbleibenden Lehrinhalten als Team anbieten.

	Sommersemester							
	2015	2016	2017	2018	2019	2020	2021	2022
	Präsenz					Online		?
Teamteaching in Großveranstaltung		✓	✓		✓	✓	✓	✓
Inverted Classroom						✓	✓	✓
Differenzierung durch Zusatzmaterial							✓	✓
Studentisches Tutoring							✓	✓
Vorlesung parallel zu Inverted Classroom								✓
Notendurchschnitt beider Klausuren	3.96	3.76	3.55	3.61	3.34	3.71	3.06	

Tab. 1: Eingesetzte Methoden und Notendurchschnitte in den verschiedenen Jahren

Überblicksartig ist schon einmal in Tabelle 1 die Entwicklung des Konzepts der Lehrveranstaltung sowie der Notendurchschnitte für die Jahre 2015-2021 dargestellt. Folgende Anmerkungen zu Tabelle 1 sind dabei wichtig für deren Interpretation:

- 2015 wurde die Veranstaltung von einem der Lehrenden aus dem oben angesprochenen Team als Großveranstaltung durchgeführt.
- 2018 wurde die Veranstaltung von drei Lehrenden in kleineren Gruppen durchgeführt, wobei nur einer der drei Lehrenden aus dem oben angesprochenen Team stammt und nach dem vom Team verwendeten Skript liest sowie vergleichbare Klausuren konzipiert. Der Notendurchschnitt seiner Klausur ist hier aufgeführt.
- Der Notendurchschnitt von 2021 entspricht nur der ersten Klausur, da die Ergebnisse der Nachklausur zum Zeitpunkt der Drucklegung noch nicht vorlagen. Sie verschlechtert erfahrungsgemäß den Notendurchschnitt manchmal um bis zu 0.2.
- Die Notendurchschnitte wurden immer etwas besser, außer 2018, als kein Teamteaching durchgeführt wurde, und in dem ersten Pandemie-Semester 2020. In den beiden Fällen setzte sich der positive Trend aber danach wieder fort.
- Die Klausuren haben vergleichbare Schwierigkeitsgrade, soweit dies überhaupt möglich ist.

Im Folgenden werden zunächst die grundsätzlichen Herausforderungen der Lehrveranstaltung „Algorithmen und Datenstrukturen“ an der Frankfurt University of Applied Sciences aus Sicht der Autoren dargestellt. Anschließend wird auf die spezifischen mit der COVID-19-Pandemie einhergehenden zusätzlichen Herausforderungen in den Sommersemestern 2020 und 2021 eingegangen.

Wie in [LS17] berichtet, beruht der Inhalt der Veranstaltung auf einem sehr guten und ausführlichen Skript. Die Lehrinhalte blieben im Zeitverlauf weitgehend unverändert. Auch der fürs Teamteaching benötigte Lehrplan ist inhaltlich relativ unverändert gegenüber der initialen Erstellung geblieben.

In die jeweils im Sommersemester angebotene Lehrveranstaltung schreiben sich jährlich um 400 bis 500 Studierende mit vielfältigen Bildungshintergründen und Lebenssituationen ein – im Folgenden die wesentlichen Diversitätsdimensionen: Mathematische Grundbildung der Studierenden, Kenntnisse und Erfahrung in der Programmierung, soziale und fachliche Interaktionsfähigkeiten und zeitliche Restriktionen.

Die organisatorischen Rahmenbedingungen der Pandemie-Semester haben die daraus resultierenden Herausforderungen noch verstärkt. Während im Sommersemester 2020 die Studierenden im zweiten Semester immerhin einen Großteils ihres ersten Semesters in Präsenz lernen konnten, kennen die Studierenden des zweiten Semesters im Sommersemester 2021 lediglich das Format der Onlinelehre. Auch die persönlichen und sozialen Kontakte zu den Mitstudierenden sind, nach unserer Wahrnehmung, entsprechend weniger stark ausgeprägt. Dies stellt für die Studierenden eine enorme Herausforderung dar, welche mit Motivationsverlusten und Leistungseinbußen einhergehen und zur psychischen Belastung beziehungsweise im schlimmsten Fall ernsthaften Erkrankung werden kann, vgl. [SK21].

Die Onlinelehre birgt auch aus Lehrendenperspektive Herausforderungen. So lässt sich etwa die Dynamik eines Tafelvortrags zur Erarbeitung mathematischer Sachverhalte nur schwer in ein digitales Format übersetzen. Auch der Einsatz physischer Exponate, wie etwa ein großes Schiebepuzzle aus Holz [LS17], entfaltet in Präsenz eine wesentlich stärkere Wirkung. Die wohl größte Herausforderung liegt aber im digital massiv limitierten persönlichen Kontakt zu den Studierenden, woraus insbesondere eine eingeschränkte Wahrnehmung von nonverbalen Feedback der Studierenden resultiert. Da in digitalen Großveranstaltungen das Gros der Teilnehmenden auf die visuelle Teilnahme via Webcam verzichtet, agieren die Lehrenden vielfach im Ungewissen darüber, ob und womit sie ihre Studierenden erreicht haben. Dies kann durch aktivierende Methoden wie durchgeführte Umfragen etwas abgemildert werden, diese ersetzen jedoch nicht den „Blick in die Runde“.

2 Detaillierte Diskussion der Konzepte

Für die Lehrenden stellte sich im Sommersemester 2020 also erstmals die Frage, wie unter den pandemiebedingten Rahmenbedingungen der digitalen Lehre konstruktiv mit den zunehmend heterogenen Bildungsverläufen und diversen Lebensumständen der Teilnehmerinnen und Teilnehmer umgegangen werden kann, um bestmögliche Unterstützung anzubieten. Da die Lehrenden in Präsenz seit 2016 Teamteaching erfolgreich umgesetzt hatten, wurde an dieser Methode festgehalten. Dies bedeutet, die beiden Lehrenden planen die Lehrveranstaltungen gemeinsam und führen diese in gleichberechtigter Zusammenarbeit und in rotierenden Rollen durch. Die Lehrenden können sowohl gemeinsam als auch differenzierend auf den

Lernprozess der Studierenden einwirken und diesen insbesondere vielfältige Perspektiven auf dieselbe Fragestellung anbieten. Wöchentlich werden die Lehrveranstaltungen gemeinsam kritisch analysiert, evaluiert und bei Bedarf den Bedürfnissen der Studierenden entsprechend angepasst.

Obgleich die Vorlesung im Teamteaching in Präsenz von Studierenden wie auch Lehrenden positiv bewertet wurde, vgl. [LS17], erschien den damals verantwortlichen Lehrenden das Konzept einer digitalen Frontalvorlesung ungeeignet, da sie vor dem Hintergrund der oben genannten Herausforderungen ein direktes Feedback an die Studierenden aber auch einen synchronen Rückkanal für Feedback an die Lehrenden als essentiell erachteten. In der Onlinelehre müssen, noch stärker als in Präsenz, motivierende Reize gesetzt werden, vgl. etwa [Pr21], und um der Gewöhnung an diese Reize vorzubeugen sind die Lehrenden gefordert, kontinuierlich für Abwechslung zu sorgen, vgl. [He17]. Bei der Rhythmisierung der Lehrveranstaltung bietet Teamteaching zahlreiche Möglichkeiten und Vorteile gegenüber der Lehre durch eine einzelne Lehrperson. Darüberhinaus entstand die Idee, die Studierenden aktiv in die Gestaltung der Lehrveranstaltungen einzubeziehen und so deren Interaktion untereinander und im Bezug auf die Lehrenden zu fördern. Daher entschieden wir uns für ein hybrides Konzept, welches das Inverted Classroom Konzept mit regelmäßigen vorlesungsartigen Schlaglichtern verbindet. An Materialien wurden ein Vorlesungsskript, Screencasts zu ausgewählten Themen sowie C- und Python- Implementierungen der behandelten Algorithmen zur Verfügung gestellt. Das verbindende zentrale Element ist ein detaillierter Lehrplan, welcher die zum jeweiligen Termin zu behandelnden Abschnitte des Skripts beziehungsweise Screencasts, Programmcodes sowie im Vorfeld zu beantwortende Fragen enthält. Inverted Classroom bedeutet in diesem Zusammenhang, dass die Studierenden vor der Präsenzphase das bereitgestellte Material bearbeiten und Fragen zu diesem Material in einem Diskussionsforum auf dem E-Learning-System stellen und idealerweise auch untereinander diskutieren. Die Lehrenden moderieren dieses Forum und geben den Studierenden bei Bedarf Hilfestellung. In der Präsenzphase werden dann die Fragen der Studierenden sowie ggf. weitere Fragen der Lehrenden gemeinsam diskutiert. Auch hier planen die Lehrenden die Präsenzphasen gemeinsam, führen sie in rotierenden Rollen durch und analysieren diese wöchentlich. Punktuell werden sogenannte Breakout-Sessions zur Aktivierung der Studierenden eingesetzt. Flankierend wurden sieben Online-Übungsgruppen angeboten. Im Sinne des Constructive Alignment Ansatzes [Bi96] wurden Lehr-/Lernmethoden, Lernziele und Prüfungsform bereits bei der Planung der Lehrveranstaltung aufeinander abgestimmt und transparent kommuniziert.

Aufgrund des Feedbacks aus der Evaluation im Sommersemester 2020 sind wir zu der Auffassung gelangt, dass das hybride Konzept durchaus geeignet ist für die Durchführung der Lehrveranstaltung unter Pandemiebedingungen. Das Format im Sommer 2021 war grundsätzlich identisch zum oben beschriebenen, diesmal jedoch mit zehn Online-Übungsgruppen. Im Folgenden stellen wir gezielt eingeführte punktuelle Verbesserungen in Form von Impulsen vor, die die Studierenden dabei unterstützen sollen, von ihrem individuellen Vorwissen zu den angestrebten Lernergebnissen zu gelangen.

2.1 Jupyter-Notebooks

Die Studierenden erhielten schon zuvor über das E-Learning-System Zugriff auf alle in der Veranstaltung behandelten Algorithmen und Datenstrukturen in C und Python. Die Sprache C ist den Studierenden aus dem ersten Semester bekannt. Python nicht, jedoch werden viele Algorithmen im Skript auch als Python-Code mit einfach verständlichen Sprachkonstrukten eingeführt. In manchen Übungsaufgaben ist die Ausführung von Programmcode ein Teil der Aufgabe oder zumindest hilfreich, um die Aufgabe erfolgreich zu bearbeiten. Erfahrungsgemäß arbeiten dennoch die wenigsten Studierenden mit dem Programmcode. Die Gründe dafür sind vielfältig, lauten beispielsweise fehlende Zeit, Lerneffekt nicht erkannt oder entsprechende Software nicht installiert, vgl. hierzu auch [Ra19; Sc17].

Aus diesem Grund entwickelte einer der Autoren Jupyter Notebooks, welche zum „Herumspielen“ mit grundlegenden Algorithmen und Datenstrukturen einladen sollen [Li21]. Diese Implementierungen sind nicht nur den Studierenden der Veranstaltung zugänglich, sondern im Internet frei verfügbar. Es existieren derzeit zwölf Algorithmen und Datenstrukturen, die meisten in jeweils drei Ausprägungen.

Kompakter Code mit wenigen Tests. Auf diesem Weg können die Algorithmen und Datenstrukturen „ausprobiert“ werden. Änderungen sind einfach möglich. Beispielsweise können die zu sortierenden Schlüssel für einen Sortieralgorithmus geändert werden.

Code mit ausgewählten Statistiken für den Algorithmus oder die Datenstruktur in grafischer Form. Hier ist zum Beispiel das Laufzeitverhalten für unterschiedlich große Eingaben erkennbar. Die Änderung des Codes ist hier auch möglich. Es können sowohl Daten als auch Statistiken und deren Visualisierung abgeändert werden, jedoch ist dies wegen der benötigten Code-Instrumentierung etwas aufwendiger als im vorhergehenden Punkt.

Code mit einer statischen Visualisierung des Verhaltens zur Laufzeit. In dieser Variante werden die einzelnen Schritte zur Laufzeit des Algorithmus oder der verwendeten Datenstruktur nebeneinander angezeigt. Dies hilft, das Verhalten der Algorithmen zu studieren. Im Vergleich zur dynamischen Visualisierung kann man hier auf einen Blick gut erkennen, was während des Ablaufs des Algorithmus oder der Arbeit mit einer Datenstruktur passiert. Ein Beispiel ist in Abbildung 1 zu finden. Dort wird ein Feld mittels Bubblesort sortiert. Die Visualisierung zeigt die einzelnen Schritte im Sortiervorgang, von links nach rechts eine „Spalte“ pro anzuzeigendem Sortierschritt. Eine Auswahl der angezeigten Schritte (wie Vergleiche und Vertauschungen) ist im Code ebenfalls durch Setzen von Parametern sehr einfach. Änderungen an der Visualisierung sind hier auch realisierbar, erfordern jedoch deutlich mehr Aufwand wegen der Einarbeitung in die komplexere Code-Instrumentierung und Visualisierungs-Funktionalität.

Jupyter Notebooks als interaktive Ausführungsumgebung wurde gewählt, um den Studierenden den Einstieg in Python zu erleichtern. Diese werden auf [Li21] in einer Form zur Verfügung gestellt, die ohne Installation auf jedem Computer und mobilen Endgerät genutzt

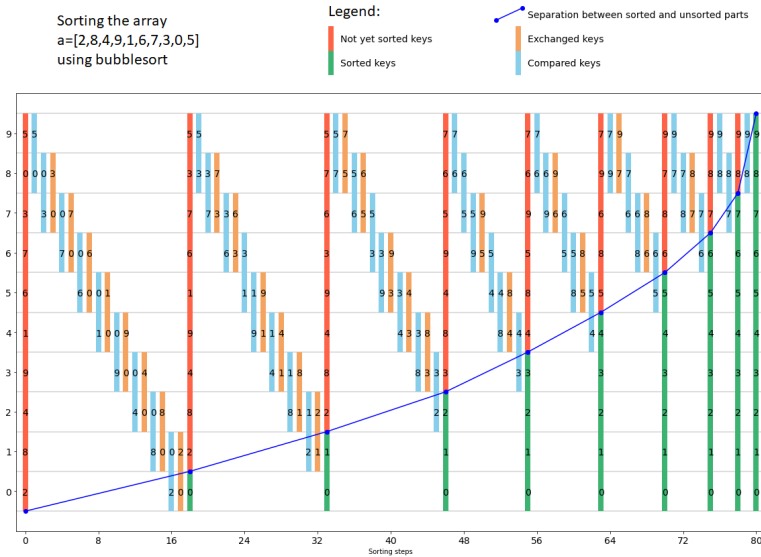


Abb. 1: Visualisierung von Bubblesort eines Feldes [2,8,4,9,1,6,7,3,0,5] [Li21]

werden kann. Es ist so möglich, sehr schnell eigene Experimente mit den Algorithmen und Datenstrukturen durchzuführen. Deutlich mehr Hintergrundinformation ist in den FAQs auf [Li21] zu finden.

Bei der Einführung der drei nachfolgend aufgeführten Neuerungen war es uns wichtig, im Sommersemester 2021 ein noch stärkeres Augenmerk auf den Aspekt der Feedbackkultur zu legen, um die Studierenden noch besser im Lernprozess während des Pandemie-Semesters zu unterstützen. Feedback wirkt laut [HT07] auf drei Ebenen:

- Feed-Up: Was ist das Ziel?
- Feedback: Wie weit ist die einzelne Person?
- Feed-Forward: Was sind die nächsten Schritte?

Die Autoren erachten diese Perspektive als geeignet, ein agiles System innerhalb der Feedbackkultur zu etablieren. Während dem Punkt Feed-Up nach Meinung der Autoren durch das Modell des Constructive Alignment Rechnung getragen wird, erfordern die Punkte Feedback und Feed-Forward in Anbetracht der großen Lerngruppe besondere Aufmerksamkeit. Dazu wurden im Sommersemester 2021 zusätzliche optionale Self-Assessment Fragen auf dem Niveau der Level 1 bis 3 der SOLO Taxonomie [BT11] sowie Peer-Review-Übungen auf dem Niveau der beiden höchsten Level implementiert. Zur stärkeren Akzentuierung sowohl der Feedback als auch der Feed-Forward Komponente wurde zudem ein studentisches Tutoring initiiert.

2.2 Fragen in Moodle

Einer der Autoren hat bereits vor der COVID-19-Pandemie eine Lehrveranstaltung im ersten Semester eines Informatik Bachelor-Studienganges an der Frankfurt University of Applied Sciences zum Teil digitalisiert [Li20]. Einer der Aspekte war die Erstellung von Fragen zu ausgewählten Themengebieten im E-Learning-System. Diesen Ansatz haben die Autoren hier übernommen. Zunächst wurden (aus Zeitgründen semesterbegleitend) zu zwölf Themengebieten insgesamt 73 Fragen vorbereitet und ins E-Learning-System eingestellt. Es wurde darauf geachtet, durch die Fragen die Level 1 bis 3 der SOLO Taxonomie abzudecken. Eine Erweiterung des Fragenkatalogs – auch auf weitere Themengebiete – ist möglich und in der Diskussion für den nächsten Zyklus. Drei unterschiedliche Fragetypen wurden eingesetzt: Freitext und Single- und Multiple-Choice. Die Studierenden erhalten auf diesem Weg eine Möglichkeit, den individuellen Lernstand jederzeit zu überprüfen. Somit ist dies eine gute Möglichkeit des direkten und schnellen Feedbacks. Die Studierenden können nach Beantwortung der Fragen eines Themengebiets nicht nur Ihren Erfolg betrachten, sondern sehen auch die korrekten Antworten. Die Tests sind durch die Studierenden beliebig oft wiederholbar.

2.3 Peer-Review-Übungen

Ebenso wie die Fragen in Moodle stellt diese Neuerung ein begleitendes formatives Assessment für die Studierenden dar. Es wurden im Laufe des Semesters vier optionale Übungsblätter angeboten, wobei im Vorfeld transparent kommuniziert wurde, dass dieses Angebot sich an jene Studierende richtet, die sämtliche Fragen im Lehrplan, Übungsaufgaben und Fragen in Moodle ohne größere Schwierigkeiten bearbeiten können oder diese gar als leicht empfinden. Nach Ablauf der Bearbeitungszeit haben die Studierenden – nach Abgabe eines eigenen Lösungsvorschlags – eine detaillierte Musterlösung erhalten, auf deren Basis dann die eigene Abgabe sowie die Abgabe einer Kommilitonin oder eines Kommilitonen bewertet wurde. Neben dem Aspekt der Binnendifferenzierung erachten wir das Peer-Feedback als gewinnbringend, sowohl was den Aspekt des kooperativen Lernens betrifft, als auch im Hinblick auf den gegenseitigen Austausch unter den Studierenden, nicht nur in den Pandemie-Semestern.

2.4 Studentisches Tutoring

Zur Umsetzung dieser Neuerung wurde eine studentische Hilfskraft im Umfang von vier Semesterwochenstunden eingestellt. Dieser Tutor hatte die Veranstaltung im vorigen Sommersemester mit weit überdurchschnittlichen Leistungen abgeschlossen. Er hat in seiner Rolle als Tutor ein Forum auf der Moodle Plattform moderiert, welches ausschließlich den Studierenden zugänglich war, und in diesem Rahmen mit seinem Wissen sowie durch

individuelles Feedback und Feed-Forward die Teilnehmerinnen und Teilnehmer unterstützt. Ziel dieser Neuerung war es, den Studierenden einen virtuellen Raum für Diskussionen – ohne Beobachtung durch die Lehrenden – bereitzustellen als Substitut für physische studentische Lern- und Diskussionsräume, welche unter Pandemiebedingungen an der Hochschule nur sehr eingeschränkt zur Verfügung stehen.

3 Feedback durch Studierende

3.1 EvaluierungsService EvaS

Der EvaluierungsService EvaS [Ev21] unterstützt die Lehrenden an der Frankfurt University of Sciences bei der Durchführung von Befragungen. Er ist Bestandteil der allgemeinen Qualitätssicherung und ermöglicht sowohl die standardisierte, anonyme, wiederholbare Abfrage und Auswertung von Daten in Bezug auf Bewertung der Veranstaltungen als auch die individuelle Erstellung von Fragebögen zu spezialisierten Themen. Im Rahmen der Teamteaching-Veranstaltung wurde mit EvaS ein Fragebogen entwickelt, der ausschließlich die Aspekte des Teamteaching erfasst, vgl. [LS17], dieser wurde in den Pandemie-Semestern jeweils auf die gegebenen Rahmenbedingungen und Lernangebote angepasst.

3.2 Qualitative Ergebnisse

In den Evaluationen der beiden Pandemie-Semester ist zunächst auffällig, dass das Verhältnis zwischen eingereichten Evaluationsbögen und Teilnehmenden an der Klausur in beiden Pandemie-Semestern jeweils weit hinter dem in Präsenz beobachteten Verhältnis zurückbleibt. Ferner sticht beim Vergleich die deutlich schlechtere Bewertung durch die Studierenden für das Sommersemester 2021 ins Auge; die Gründe hierfür sind für uns nur schwer zu eruieren. Ein möglicher Erklärungsansatz könnte wie folgt lauten: Aus den Freikommentaren geht hervor, dass ein Teil der Studierenden die Vorbereitung anhand des Skripts und der bereitgestellten Hilfsmittel als zu anspruchsvoll empfunden hat, woraus teilweise ein frustrierendes Gefühl des „Abgehängtseins“ resultierte. Auch der subjektiv höher empfundene Zeitaufwand wird von Studierenden kritisiert. Andere Studierende weisen auf die aus der Diversität der Teilnehmenden resultierende Herausforderung hin, das Niveau der Präsenzphasen für möglichst viele angemessen zu gestalten – ein berechtigter Kritikpunkt, dem in der kommenden Veranstaltung Rechnung getragen werden wird, vgl. Abschnitt 4. Demgegenüber stehen positive Freikommentare von Studierenden, welche die erhofften Effekte wie Eigenverantwortlichkeit, nachhaltigeres Verständnis, Aktivierung und gesteigerte Interaktion in der Gruppe positiv hervorheben.

Nachstehend sind einige Zitate der Studierenden aus dem Feedback durch [Ev21] zusammengefasst².

² Dabei wurden nur diejenigen Zitate ausgewählt, die einen Bezug zum Inverted Classroom aufweisen – allgemeine Bemerkungen fachspezifischer Art sind hier nicht dargestellt.

Positiv wurde bewertet:

- Angeeignetes Wissen kann direkt vertieft werden.
- Eigenverantwortung; Man kann sich selbst alles beibringen und bei diesem Konzept lernt man wie.
- Man wird mehr dazu angeregt, die Lehrmaterialien rechtzeitig zu bearbeiten, sodass man eher auf dem gleichen Wissenstand ist wie die anderen Studenten, was den Austausch erleichtert.
- Bildung von Lerngruppen, mehr gemeinsame Arbeit, bessere Gruppendynamik, tolle Diskussionen.
- Durch das Durcharbeiten der Lernmaterialien vor der Veranstaltung kann man sich mehr Gedanken über den Stoff machen und dadurch gezielter Fragen stellen.
- Kein Frontalunterricht, nicht eintönig, abwechslungsreicher.
- Man wird aktiv eingebunden.
- Es gibt zusätzliche Übungsaufgaben.

Negativ wurde bewertet:

- Es ist schwierig, präzise Fragen zu formulieren. Setzt ein Verständnis voraus. Wer das nicht hat, fällt durchs Raster.
- Die Präsenzphase hat keine Auswirkung auf meinen Lernerfolg. Ich muss mir alles alleine beibringen, von daher komme ich in die Präsenzphase und weiß schon alles. Die Aufgaben, die wir in der Präsenzphase durchführen, sind zu dem Zeitpunkt so einfach, dass ich sie ignorieren kann. Wenn ich jedoch nichts vorher vorbereite, sind die Aufgaben so kompliziert, dass ich nichts verstehe.
- Die Unflexibilität in Sachen Zeit, wann und wie man den Stoff lernt.
- Man lernt besser aber dafür braucht man meistens doppelt so viel Zeit, um ein Thema zu verstehen als bei einer normalen Vorlesung. Das führt dazu, dass man sehr leicht zurückfällt und vielleicht auch nicht mit den Veranstaltungen mithalten kann.

3.3 Quantitative Ergebnisse

Tabelle 2 stellt die Klausurergebnisse in Form des Notenspiegels dar, und zwar für die Veranstaltung aus dem Sommersemester 2019, welche in Präsenz im Teamteaching als klassische Vorlesung stattfand, und die Veranstaltungen aus den Pandemie-Semestern 2020

und 2021, welche online und im Inverted Classroom Konzept durchgeführt wurden³. Die

Note	1	1.3	1.7	2	2.3	2.7	3	3.3	3.7	4	5	∅
2019	11	10	5	4	7	6	6	12	10	9	27	3.16
2020	10	0	2	2	6	4	7	8	7	15	40	3.72
2021	12	4	5	6	11	11	11	10	8	12	20	3.06

Tab. 2: Notenspiegel

Vorjahresklausuren wurden durch einen der Autoren gestellt und sind in Form und Inhalt objektiv vergleichbar und vom Schwierigkeitsgrad auf demselben Niveau anzusiedeln. Wie man sieht, ist die Klausur im Sommersemester 2021 nicht nur deutlich besser ausgefallen als die Klausur aus dem Sommersemester 2020; auch im Vergleich zur Klausur aus dem in Präsenz durchgeführten Sommersemester 2019 ist eine leichte Verbesserung festzustellen. Dies war, insbesondere in Anbetracht der Diskrepanz zwischen der summativen Bewertung und dem im Rahmen der Evaluation erhobenen Feedback der Studierenden, für uns eine kleine Überraschung.

Obgleich es sich hier nur um eine Momentaufnahme und eine verhältnismäßig kleine Stichprobe handelt, kann man bei aller Vorsicht vermuten, dass das bereits vor den Pandemie-Semestern initiierte und kontinuierlich weiterentwickelte und an aktuelle Rahmenbedingungen angepasste Teamteaching-Konzept der Mehrheit der Studierenden in dieser herausfordernden Situation geholfen zu haben scheint.

4 Ausblick auf Änderungen des Lehrformats

Das vorgestellte Konzept wurde gewählt, um der lernrelevanten Diversität der Gruppe der Studierenden besser gerecht zu werden. Schon in der Diskussion mit den Studierenden in der letzten Veranstaltung wurde den beiden Lehrenden klar, dass dies nicht bei allen Studierenden zum Lernerfolg beigetragen hat. Für einen Teil der Studierenden war das eigenständige Vorbereiten der Lerninhalte sehr herausfordernd. Um auch diesem Aspekt gerecht zu werden, wird das Lehrformat im Sommersemester 2022 dahingehend geändert, dass die Differenzierung der Lerngelegenheiten schon in der Vorlesung beziehungsweise der Präsenzphase des Inverted Classrooms ansetzt: Nach derzeitigem Stand der Planung wird die Veranstaltung wie bisher im Teamteaching als Großveranstaltung begonnen. Es werden zunächst ein paar Wochen lang Vorlesungsanteile mit Inverted Classroom kombiniert. Dadurch lernen die Studierenden beide Konzepte und deren Konsequenzen für den individuellen Lernprozess kennen. Danach werden zum gleichen Termin zwei parallele Veranstaltungen angeboten werden, eine als Vorlesung, die andere als Inverted Classroom.

³ Zur besseren Vergleichbarkeit wurden die Ergebnisse der Nachklausuren nicht berücksichtigt, da die Ergebnisse der Nachklausur aus dem Sommersemester 2021 zum Zeitpunkt der Drucklegung noch nicht vorlagen.

Somit hat diese Änderung keinen Einfluss auf den Stundenplan, jedoch auf die Raumplanung. Eine wichtige Voraussetzung ist die unbedingte Einhaltung des vorhandenen Lehrplans. In jeder der beiden Veranstaltungen müssen die gleichen Themen behandelt werden. Somit ist es sowohl für die Studierenden als auch für die Lehrenden möglich, zwischen den beiden Formaten zu wechseln. Außerdem passen die Vorlesung beziehungsweise Präsenzphase zeitlich wie bisher zu den Übungen. Natürlich wird es wie bisher für die Veranstaltung mit beiden Formaten nur eine Klausur geben.

Besonders wichtig in diesem Format wird die Kommunikation mit den Studierenden sein. Die Studierenden müssen sich der Vielzahl der Lernangebote bewusst sein, um individuell eine gute Wahl treffen zu können. Die Lehrenden müssen dazu eine Orientierung geben.

Jedoch hat dieser Ansatz auch einen Nachteil. Denn nach dem Aufspalten der Veranstaltungen werden beide Veranstaltungen nicht mehr als Teamteaching angeboten. Eine Möglichkeit der Kompensation wäre der Einsatz weiterer Lehrender, sofern die Rahmenbedingungen dies ermöglichen.

Auch wenn das Zusatzangebot zur Unterstützung durch den studentischen Tutor von weniger Studierenden angenommen wurde als erwartet, erscheint den Lehrenden die Beibehaltung sinnvoll. Während die mäßige Resonanz im Online-Format einer zunehmenden „Online-Müdigkeit“ der Studierenden geschuldet sein könnte, erscheint eine Übertragung des Ansatzes in die Präsenzlehre vielversprechend. So könnten für die Studierenden zu festen Zeiten in festen Räumen studentische Mitarbeitende zur Diskussion über den Lernstoff zur Verfügung stehen.

5 Fazit

Im vorliegenden Artikel wurde die Entwicklung des Lehrkonzepts für die als „schwierig“ geltende Veranstaltung „Algorithmen und Datenstrukturen“ an der Frankfurt University of Sciences vorgestellt. Im Hinblick auf die Klausurergebnisse kann man von einer erfolgreichen Entwicklung sprechen. Mit Ausnahme des ersten Pandemie-Semesters waren die Notendurchschnitte immer etwas besser als im vorangegangenen Zyklus. Für das Sommersemester 2022 ist geplant, die Veranstaltung nach einer gemeinsamen Anlaufphase in der Großgruppe aufzuteilen. Bei identischen Lerninhalten werden zwei parallele Veranstaltungen durchgeführt, eine als Vorlesung, die andere als Inverted Classroom. Dieses Format kann sowohl in Präsenz als auch online angeboten werden. Auf diesem Weg hoffen die Lehrenden, zukünftig den Studierenden ein noch besseres – weil stärker individualisierbares – Angebot bereitstellen zu können.

Literatur

- [Bi96] Biggs, J.: Enhancing teaching through constructive alignment. Higher education/32(3), S. 347–364, 1996.

- [BT11] Biggs, J.; Tang, C.: Teaching for Quality Learning at University. Open University Press, McGraw-Hill Education, 2011.
- [Ev21] EvaS: EvaluationsService, 2021, URL: <https://www.frankfurt-university.de/index.php?id=4627>.
- [He17] Helmke, A.: Unterrichtsqualität und Lehrerprofessionalität, 7. Auflage. Klett-Kallmeyer, 2017.
- [HT07] Hattie, J.; Timperley, H.: The Power of Feedback. Review of educational research/77(1), S. 81–112, 2007.
- [KR16] Kricke, M.; Reich, K.: Teamteaching. Beltz Verlagsgruppe (Julius Beltz GmbH & Co. KG) Weinheim und Campus Verlag GmbH Frankfurt, 2016.
- [Le06] Leavitt, M. C.: Team Teaching: Benefits and Challenges. Speaking of Teaching, Stanford University Newsletter on Teaching 16/1, 2006, URL: <http://web.stanford.edu/dept/CTL/Newsletter/teamteaching.pdf>.
- [Li20] Liebehenschel, J.: Digitization of a Lecture - An Experience Report. In: IEEE Global Engineering Education Conference, EDUCON. S. 1–6, 2020.
- [Li21] Liebehenschel, J.: Grundlegende Algorithmen und Datenstrukturen in Jupyter Notebooks, 2021, URL: <https://github.com/JensLiebehenschel/AlgDat>.
- [LS17] Liebehenschel, J.; Schäfer, J.: Teamteaching - ein Fallbeispiel. CEUR Workshop Proceedings/1790, S. 91–99, 2017.
- [Pr21] Prietl, B.: Ein maximal selbstreflexives Lehrforschungsprojekt – oder: Was sich aus Studierendenperspektive aus dem Corona-Semester für digitale Fernlehre im Hochschulkontext lernen lässt. In (Prietl, B., Hrsg.): Online-Lehre auf dem Corona-Prüfstand. Wie JKU-Studierende den Umstieg auf Distance Learning im Sommersemester 2020 erlebt haben: Endbericht zum Lehrforschungspraktikum “Digitalen Wandel erforschen”. Johannes Kepler Universität, Linz, S. 115–129, 2021.
- [Ra19] Radfelder, O.; Vosseberg, K.; Erb, U.; Lipskoch, H.: Informatik ist nicht nur Programmieren–aber ohne Programmieren ist nichts Informatik. CEUR Workshop Proceedings/2358, S. 65–74, 2019.
- [Sc17] Schmolitzky, A. W.: Zahlen, Beobachtungen und Fragen zur Programmierlehre. CEUR Workshop Proceedings/1790, S. 83–90, 2017.
- [SK21] Schmermund, K.; Kaiser, R.: Wenn das digitale Studium zur Belastung wird, 2021, URL: <https://www.forschung-und-lehre.de/lehre/wenn-das-digitale-studium-zur-belastung-wird-3413/>.
- [WMW06] Wadkins, T.; Miller, R. L.; Wozniak, W.: Team Teaching: Student Satisfaction and Performance. Teaching of Psychology 22/2, S. 118–120, 2006.

Individuelle Benotung von Teamprojekten: Lessons Learnt

Karsten Weicker¹

Abstract: Teamprojekte werden individuell über ein Bewertungsschema benotet, in dem aus unterschiedlichen Quellen Punkte in den Bereichen Programmierung, Dokumente, Engagement und Präsentationen gesammelt werden. Zu den Quellen zählen neben der Dozentenbeobachtung auch die Analyse der Git-Repositories und Peer-Bewertungen. Auswertungen von 50 Projekten mit 405 Studierenden zeigen, dass sich die einzelnen Metriken gut ergänzen und so den gesamten Entwicklungsprozess berücksichtigen können.

Keywords: Software-Entwicklungsprojekte; Benotung; VCS-Analyse; Peer-Bewertung

1 Einleitung

Software-Entwicklungsprojekte mit häufig auch großen Teams, d.h. > 6 Teilnehmer:innen, gehören heute zum Standardrepertoire des Informatik-Studiums, da so Anforderungen und Komplexität von Projekten aus dem Berufsalltag der Informatiker:in abbildbar sind. Dies geht zumeist mit organisatorischen und prüfungsrechtlichen Schwierigkeiten bei der Benotung der Projekte einher: die Teams sind zu groß bzw. die individuelle Notenfindung gestaltet sich schwierig (vgl. den beispielhaften Auszug aus einer Prüfungsordnung: “Dabei muss der als Prüfungsleistung zu bewertende Beitrag des Einzelnen als individuelle Prüfungsleistung deutlich abgrenzbar und zu bewerten sein. Die Gruppe sollte in der Regel nicht mehr als drei Studierende umfassen.”²).

Da die praktizierten (und fachdidaktisch auch sinnvollen) Teamgrößen meist über der Empfehlung der Gruppengröße liegen, muss die Bewertung der individuellen Anteile mit besonderer Sorgfalt durchgeführt werden. Typische Strategien hierfür sind:

1. Verzicht auf eine Note zugunsten eines Bestanden/Nicht-Bestanden auf Basis von Minimalkriterien,
2. Vergabe einer Gruppennote mit Abweichungen nach oben und unten auf Basis einer Peer-Bewertung oder Verhandlung der Peers,
3. Vergabe einer Gruppennote mit Abweichungen nach oben und unten aufgrund von Einschätzungen durch die Dozent:in und

¹ HTWK Leipzig, Fakultät Informatik und Medien, Gustav-Freytag-Str. 42A, 04277 Leipzig, karsten.weicker@htwk-leipzig.de

² Prüfungsordnung der Fakultät für Mathematik und Informatik für den Studiengang Informatik, Friedrich-Schiller-Universität Jena, <https://www4.uni-jena.de/unijenamedia/p-43470.pdf>

4. Ermittlung der Note als Akkumulation mehrerer (evtl. gewichteter) Noten für einzelne Artefakte.

Jede Strategie birgt Herausforderungen und Schwierigkeiten: (1) würdigt nicht den hohen Aufwand der Studierenden und wirkt tendenziell demotivierend, (2) kann prüfungsrechtlich kritisch sein, da die Prüferrolle eindeutig zugeordnet ist, (3) erschwert ggf. der Prüfer:in die Argumentation, dass die Leistung einzelner trotz eines guten Projekts nicht bestanden ist, und (4) kann leicht als eine Sequenz von Hausaufgaben verstanden werden, was den eigentlichen zu benotenden Projekt- und Teamkompetenzen nicht gerecht wird.

In den an der HTWK Leipzig in den Informatik- und Medieninformatikstudiengängen durchgeführten Software-Entwicklungsprojekten, die zwei Semester (3. und 4. Fachsemester) dauern, wird seit 2014 ein Bewertungsschema angewandt, das über ein Punktesystem die Beobachtungen der Dozent:innen und die Peer-Bewertung mit Metriken aus dem Software-Entwicklungsprozess verknüpft [We16]. Dabei werden die Noten ausschließlich individuell für jede Student:in ermittelt – es geht im Gegensatz zu den oben angeführten Strategien (2) und (3) keine Note für das gesamte Projekt in die individuelle Benotung ein.

Das Konzept folgt dabei der zentralen Idee, über verschiedene Metriken/Teilaspekte den unterschiedlichen Facetten einer Gruppenarbeit (individuelle Programmierleistung vs. sozialen Kompetenzen) gerecht zu werden und es insbesondere den “Trittbrettfahren” schwer zu machen, mit wenigen optimierten Aktionen oder gutem Selbstdarstellungsvermögen die Note wesentlich zu beeinflussen.

Ähnliche Ansätze bezüglich der (halb-)automatischen Analyse einzelner Metriken bei der Benotung von Projekten können in der Arbeit von Coppit [Co06] bzgl. der Arbeitspakete in einem Ticketsystem und Buffardi [Bu20] bzgl. der Analyse der Versionskontrolle gefunden werden. Eine gewichtete Aggregation mehrerer Aspekte in der Form von individuellen und in der Gruppe erstellten Artefakten wurde von Hummel [Hu13] vorgeschlagen, die allerdings stark an dokumentenzentrierten Vorgehensmodellen ausgerichtet ist.

Nach der Benotung von sechs Jahrgängen an der HTWK liegen Daten aus insgesamt 50 Projekten mit 405 Bachelorstudent:innen vor, die das Projekt bis zum Ende (Abgabe des individuellen Abschlussberichts) durchgeführt haben. Auf Basis dieser Daten wird das Bewertungsschema genauer hinsichtlich seiner Anwendbarkeit bewertet.

2 Konzept

Der Benotung unterliegt ein Punkteschema, in dem 120 Punkte (plus 5 Zusatzpunkte) erreichbar sind. Ab 48 Punkten (40%) ist die Prüfung bestanden und die weiteren Noten werden gemäß einer linearen Skala berechnet.

Die Dozenten nehmen während des Projekts an verschiedenen Meetings der Teams teil (Produktvisions- und Technologieworkshop, Sprintplanning, Sprintreview und Retrospektive) und machen sich Notizen zur Beteiligung der Teammitglieder. Zudem werden pro Team drei Code-Walkthroughs durchgeführt, an denen die Teammitglieder ihre Features in der Software wie auch im Quelltext erläutern. Zu diesen Beobachtungen kommen individuelle Zuarbeiten für die Produktvision und die Technologierecherche, die Dokumentation eines Sprints, ein individueller Abschlussbericht sowie die Daten, die in der Versionsverwaltung (hier: git) sowie dem Issue-Tracking-System (Redmine, Jira bzw. YouTrack) erhoben wurden, und eine abschließende Peer-Bewertung.

Für die verschiedenen Bewertungsaspekte Programmierung, Dokumente, Engagement und Präsentationen werden Punkte aus mehreren Teilaspekten zusammengetragen. Die zuletzt angewandte Version steht in Tabelle 1. Details zu den einzelnen Metriken können [We16] entnommen werden.

Bei der Peer-Bewertung wird das relativ simple Schema aus [Hu13] benutzt, in dem die Teammitglieder eine feste Anzahl an unteilbaren Punkten auf die Teammitglieder verteilen müssen.

Das gesamte Schema ist über die vergangenen Jahre weitestgehend stabil geblieben und nur kleinere Anpassungen wurden vorgenommen.

So wurde bezüglich der Quelltextkommentare in [We16] nur das reine Enthaltensein von Kommentaren bewertet, was der heute flächendeckend üblichen Beschreibung von Schnittstellen beispielsweise in JavaDoc-Kommentaren nicht gerecht wird. Daher wurde seit 2019 in der Bewertung auf die laut [AR09] in Open-Source-Projekten durchschnittlichen 19% Kommentaranteil abgezielt und stärkere Abweichungen nach oben und unten führen zu weniger Punkten.

Im Bereich des Engagement wurde anfangs die Anzahl der Tage mit Git-Aktivität als eine Metrik genutzt, was rückblickend nicht die beste Lösung ist, da diese Metrik auch innerhalb von acht Wochen auf einen hohen Wert “gepuscht” werden kann. Daher wird seit 2018 mit der Anzahl der Wochen mit Git-Aktivität gearbeitet. Dabei wird in Wochen mit Commits/Pushes und Wochen mit geringerer Aktivität (Pull) unterschieden – letztere zählen nur als 0,5.

In diesem Paper betrachten wir ausschließlich die Benotung der Bachelorstudent:innen. Die Projekte werden jeweils zusätzlich durch 2–3 Masterstudent:innen begleitet, die diese Leistung im Rahmen einer Lehrveranstaltung Projektmanagementpraktikum absolvieren [WW09].

Metrik	Regel für Punktevergabe	Maximum
Programmierung		55
Git: Anzahl der Commits	1 Punkt pro 5 Commits	5
Git: LOC ₊ – LOC _–	1 Punkt für 100 LOC	10
Git: blamed LOC	1 Punkt für 50 LOC im Endprodukt	10
Git: Kommentare	14–24% = 5 Punkte, ab 49% 0 Punkte, linear skaliert	5
verbuchte Arbeitszeit Programmierung	1 Punkt für 7 Stunden	10
inspizierte Dateien	für drei Dateien berechnen sich anhand dem Anteil der Code-Ownership und dergröße der Datei bis zu 10 Punkte pro Datei	30
Dokumente		30
Zuarbeit Produktvision	0–6 Punkte gemäß Umfang und Qualität	6
Zuarbeit Technologierecherche	0–6 Punkte gemäß Umfang und Qualität	6
Dokumentation eines Sprints	0–16 Punkte gemäß Umfang und Qualität	16
Abschlussbericht	0–5 Punkte gemäß formaler Kriterien	5
Engagement		30
Gesamtarbeitszeit	1 Punkt pro 20 Stunden	8
Wochen mit Git-Aktivität	0,5 Punkte pro Woche, wobei Wochen mit geringer Aktivität nur halb zählen	8
Beiträge in Meetings	aus der Anzahl der Meetings mit substantiellen Redebeiträgen und der Gesamtanzahl der Beiträge werden bis zu 16 Punkte vergeben	16
Peer-Bewertung	durchschnittliche Bewertung: 5 Punkte, ohne Anerkennung der Peers: -5 Punkte, hohe Anerkennung: bis zu 10 Punkten, linear skaliert	10
Präsentationen		10
Code-Walkthroughs und Anfangsworkshops	wird aus Qualität und Länge des Beitrags errechnet	10
		125

Tab. 1: Übersicht über das Bewertungsschema (aktuelle Version).

3 Erfahrungen

Bei jährlich 7–9 Projekten mit insgesamt etwa 70–85 Teilnehmern fällt ein moderater Betreuungsaufwand während der zwei Semester an. Die Meetings und das begleitende Seminar für die Masterstudent:innen kann im Rahmen von insgesamt 8 SWS bewältigt werden. Hierfür wurden im Stundenplan pro Woche 4 SWS reserviert und zwei Lehrkräfte teilen sich die Termine auf. Dabei fällt fast ausschließlich Dokumentationsaufwand an. Kundengespräche wurden nicht in diesen reservierten Zeiten durchgeführt, da sich die Kunden aus allen Teilen der Hochschule wie externen Organisationen und Firmen zusammengesetzt haben.

Die eigentliche Benotung findet im Anschluss an die Projekte nach der Abgabe der

Abschlussberichte statt. Durch die Möglichkeit der automatisierbaren Auswertung von Projektdaten kann für die eigentliche Bewertung ein Aufwand von etwa 25–30 Minuten pro Student:in gerechnet werden. So ergibt sich bei 70–85 Prüflingen ein Gesamtzeitaufwand zwischen 30 und 43 Stunden.

Insbesondere bei der Analyse der Versionsverwaltung git können viele Aufgaben automatisiert durchgeführt werden. So werden die Metriken beispielsweise auch durch das Werkzeug gitinspector berechnet. Alternativ könnte auch, wie in [Bu20] geschildert, das Log von git mit eigenen Skripten bearbeitet werden, wobei für die Ermittlung der Urheberschaft von Codezeilen auf diesem Weg mehr Aufwand einzuplanen ist. Wesentlich tiefgreifendere Analysen des Git-Repositories sind denkbar, wie sie beispielsweise mit der Graphdatenbank Neo4J und der Software jQAssistant in [Ma17] gezeigt werden – die Techniken stammen dabei aus dem Buch [To15].

Die Verwendung von Werkzeugen zur Analyse des Git-Repositories birgt einige Schwierigkeiten, die hier kurz adressiert werden sollen:

- In zwei von 50 Projekten brach der gitinspector mit Fehlermeldungen ab. Vermutlich wurde in diesen Projekten Cherry-Picking bei der Übernahme von Commits genutzt, was zu den Problemen bei der Analyse führte. In diesen Fällen wurden die entsprechenden Daten manuell mit noch moderatem Mehraufwand erhoben.
- Bei der Verwendung von Frameworks müssen oft initiale, nicht selbst programmierte sehr große Dateien eingebunden werden. Diese sind jedoch leicht zu erkennen und werden dann im Weiteren nicht berücksichtigt.
- Am Ende zählen nur Programmierleistungen, die es in das Endprodukt geschafft haben. Diese müssen im entsprechend main- oder dev-Branch vorliegen. In Einzelfällen wurden große Programmierleistungen nicht berücksichtigt, weil sie in einem Feature-Branch verblieben sind.
- Für den vorherigen Anstrich wie auch für programmierte Dateien, die im Laufe der Entwicklung als Legacy-Code wieder entfernt wurden, wird den Projekten empfohlen, diese Dateien in einem zusätzlichen Ordner *removedcode* im Repository zu sichern, so dass dieser gegebenenfalls mit geringerer Wichtung berücksichtigt werden kann.
- Die Projekte zeichnen sich oft durch eine große Vielfalt an Programmiersprachen aus. Auch dies muss gegebenenfalls bei der Konfiguration der automatisierten Analyse berücksichtigt werden. Für neuere Sprachen wie Kotlin oder Rust werden die Dateiendungen ggf. noch nicht erkannt und die Identifikation von Kommentaren im Quelltext wird z.T. nicht automatisiert unterstützt.

Abbildung 1 zeigt, wie sich die Notenverteilung über die Jahre hinweg verändert hat. Dabei werden zwei Trends deutlich: Der Anteil an Noten “ungenügend” ging zurück und in den Anfangsjahren haben die sehr guten und guten Noten massiv zugenommen. Letztere haben sich dann weitestgehend stabil eingepegelt.

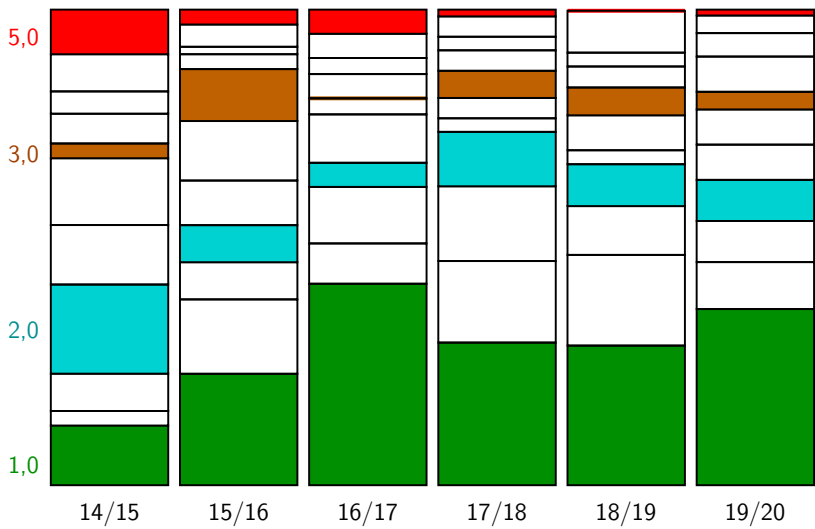


Abb. 1: Verteilung der Noten über die Jahrgänge.

Benotungsbereich	Mittelwert±sdv
Programmierung	43.55±13.65
Dokumente	23.44±5.83
Engagement	21.69±8.22
Präsentationen	6.95±2.86

Tab. 2: Durchschnittliche Punktzahl und Standardabweichung in den Bewertungskategorien (über alle 405 Studierende).

Bezüglich der zurückgehenden Durchfaller muss an dieser Stelle angemerkt werden, dass hier nur diejenigen Kandidat:innen erfasst sind, die bis zum Ende des Projekts aktiv waren und einen Abschlussbericht abgegeben haben. Während im Jahrgang 14/15 vermutlich noch viele ausprobieren wollten, ob es trotz geringer eigener Aktivität nicht doch zum Bestehen reicht, ist in den vergangenen Jahren zunehmend einen gezielten Abbruch des Projekts zu beobachten – oft verknüpft mit einem engagierten neuen Versuch im Folgejahr.

Die große Anzahl der sehr guten und guten Noten spiegelt unter anderem die zunehmende Akzeptanz des agilen Vorgehensmodells bei den Studierenden wider. Die projektleitenden Masterstudent:innen kennen dies zumeist aus eigener Werkstätigkeit oder Berufspraktika und können so die Bachelorstudent:innen gut auf den Prozess einstellen.

Der Beitrag der einzelnen Bereiche im Benotungsschema zur Gesamtpunktzahl ist in Tabelle 2 aufgeschlüsselt. Bei der Programmierung und den Dokumenten werden durchschnittlich etwas mehr als drei Viertel der möglichen Punkte erreicht. Das Engagement und die Präsentationen scheinen mit etwa 70% der Punkte geringfügig höhere Hürden darzustellen.

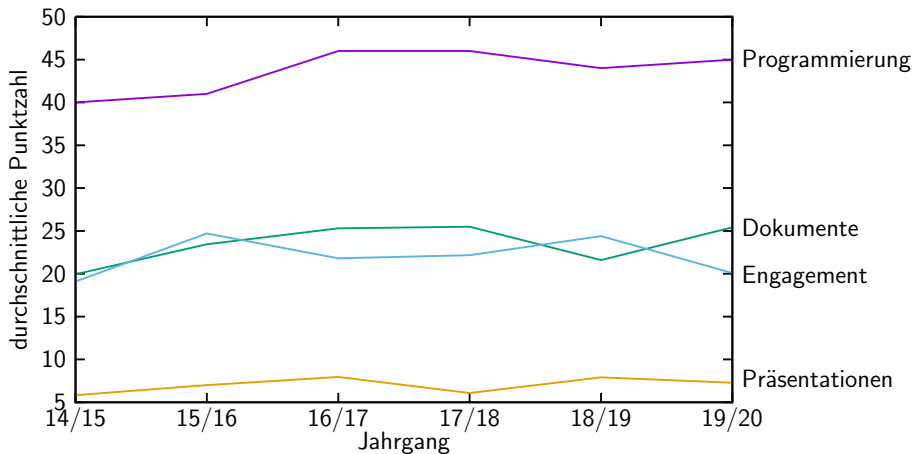


Abb. 2: Entwicklung der durchschnittlich erzielten Punkte in den einzelnen Bereichen.

Wenn man zu den einzelnen Bewertungsaspekten die Entwicklung in Abbildung 2 betrachtet, kann als Trend ebenfalls die Verbesserung der Programmierleistung beobachtet werden. Im Engagement und bei den Dokumenten zeigen die verschiedenen Jahrgänge unterschiedliche Stärken und Schwächen.

4 Reicht weniger Aufwand bei der Benotung?

Die Bewertung mit ihren vielen unterschiedlichen Kriterien ist ein aufwändiger Prozess für den Prüfer. Daher soll hier abschließend untersucht werden, ob man nicht mit weniger Kriterien und Metriken zu ähnlichen Entscheidungen bei der Notenfindung gelangen kann.

Hierfür wird in einer ersten Betrachtung die vergebene Note als “Wahrheit” aufgefasst und es wird errechnet, wie stark die Bereiche und die Metriken mit dieser Wahrheit korrelieren. Tabelle 3 enthält die Ergebnisse. Bei den einzelnen Bereichen zeigt sich deutlich, dass die höchste Korrelation zu den im Bereich der Programmierung erzielten Punkte besteht. Dies ist einerseits dadurch erklärbar, dass es sich um 55 von 125 möglichen Punkten handelt, und andererseits ist der Quelltext das Hauptartefakt der Projekte. Theoretisch könnten daher ähnliche Ergebnisse erwartet werden, wenn nur die Programmierleistung herangezogen wird. Allerdings liegt hier die Vermutung nahe, dass ohne die regulierenden Aspekte des Engagements und der Präsentationen viele Studierenden den Fokus von der Teamleistung zum reinen Produzieren von viel Code verschieben. Grundsätzlich wäre eine abgespeckte Variante der Bewertung denkbar, die lediglich die Programmierung und das Engagement berücksichtigt.

Die schwachen Korrelationen bei den Präsentationen würde ich mit der evtl. zu geringen Berücksichtigung (10 von 125 Punkten) der Codewalkthroughs bei der Notenfindung

erklären. Die Dokumente sind objektiv betrachtet für die schwächeren Studierenden eine Möglichkeit, Defizite in der Programmierkompetenz zu einem geringen Grad auszugleichen.

Die vier ausgewählten Einzelmetriken in Tabelle 3 zeigen eine moderate Korrelation mit der Note, was letztendlich allerdings auch unterstreicht, dass hier tatsächlich unterschiedliche Aspekte ermittelt werden. Dies wird belegt durch die in Tabelle 4 aufgezeigten Korrelationen zwischen den vier betrachteten Metriken. Die Werte sind insgesamt als geringe Korrelationen einzuordnen. Lediglich die Peer-Bewertung und die Anzahl der Wochen mit Git-Aktivität zeigen eine moderate Korrelation.

Zum Abschluss soll noch ein Vergleich zur Strategie (2) aus der Einleitung gezogen werden – der Vergabe einer Gesamtnote für das Projekt, welche dann für die einzelnen Teammitglieder entweder durch die Dozent:in auf Basis einer Peer-Bewertung angepasst wird oder direkt durch die Peers ausgehandelt wird (unter der Prämisse, dass die Durchschnittsnote gleich bleibt).

Wenn die Peer-Bewertung innerhalb jedes Teams die anderen Aspekte der Bewertung widerspiegeln würde, dann wäre über die Teammitglieder eines Teams eine hohe Korrelation zwischen der individuellen Note und der Einordnung in der Peer-Bewertung zu beobachten. Die Korrelation wurde für alle 50 Teams bestimmt und in Abbildung 3 zur Durchschnittsnote in Bezug gesetzt.

Mehr als 80% der Projekte zeigen dabei eine moderate bis hohe Korrelation zwischen der Note und der Peer-Bewertung, was anzeigt, dass dieses Bewertungsschema grundsätzlich zu ähnlichen Ergebnissen führen kann.

Ferner fällt auf, dass die kleineren Korrelationswerte nahezu ausschließlich bei Projekten mit sehr guten Durchschnittsnoten auftreten – mit einem prominenten Ausreißer bei Note 2,92. Zur Analyse der Gründe bei den Projekten im besseren Notenbereich, wurden für drei Projekte exemplarisch die Einzelbewertungen untersucht und zwei Auffälligkeiten als Gründe identifiziert:

Bereich bzw. Metrik	Korrelation
Punktzahl Programmierung	-0.9056
Punktzahl Dokumente	-0.5815
Punktzahl Engagement	-0.8198
Punktzahl Präsentationen	-0.5519
Punkte für blamed LOC	-0.7398
Punkte der Dateiinspektion	-0.7291
Wochen mit Git-Aktivität	-0.6932
Peer-Bewertung	-0.7312

Tab. 3: Korrelation der Note mit den Bewertungsbereichen und einigen ausgewählten Einzelmetriken (über alle 405 Studierende bzw. 219 Studierende bei der Git-Aktivität)

	blamed LOC	Dateiinspektion	Git-Wochen
Dateiinspektion	0.5994		
Git-Wochen	0.4011	0.5408	
Peer-Bewertung	0.4380	0.5207	0.6992

Tab. 4: Korrelation der einzelnen Metriken über der Datenbasis aller Studierender.

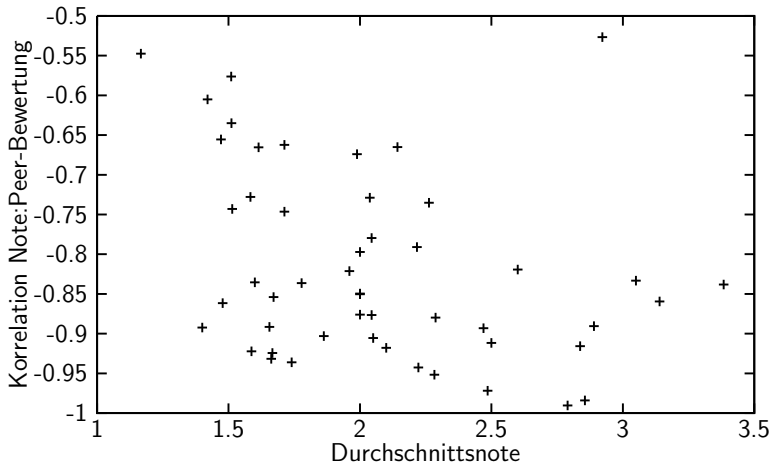


Abb. 3: Vergleich der Durchschnittsnote aller 50 Projekte mit der Korrelation der Peerbewertung im Projekt.

- Eine Einzelgänger:in trägt zwar umfassend zum Projekt bei (volle Punktzahl in der Programmierung), arbeitet aber an einem eher isolierten Teil der Software, was sich in nur wenigen Wochen mit Git-Aktivität und geringer Peer-Bewertung zeigt. Der geringeren Punktzahl im Bereich Engagement stehen allerdings volle Punktzahlen in allen Bereichen gegenüber, was in der Summe in einer sehr guten Note resultiert. (Beobachtet in zwei der Projekte).
- Die teamförderlichen Kompetenzen einer Student:in werden zwar von den Peers anerkannt, aber nicht in den Aspekten Programmierung, Dokumente und Präsentationen reflektiert. Dadurch wird nicht die Bestnote erreicht.

Bei den Projekten mit einer sehr guten Durchschnittsnote liegen häufig die Einzelnoten sehr dicht beieinander, was bei abweichender Peer-Bewertung auch zu schlechteren Korrelationswerten führt.

Der Ausreißer bei Note 2,92 entspricht einem Projekt mit vielen Schwierigkeiten. Die Mehrzahl der Student:innen hat nur schwer in die eigentliche Programmierung hineingefunden und eine Student:in hat sich als Hauptprogrammierer:in etabliert. Zudem ist die Peer-Bewertung stark durch Grüppchenbildung beeinflusst.

	1.0	1.3	1.7	2.0	2.3	2.7	3.0	3.3	3.7	4.0	5.0
beste Note	44	1	3	1	1						
schlechteste Note				3	4	2	4	9	7	10	11

Tab. 5: Verteilung der Projekte bezüglich der besten und der schlechtesten Note für die Teammitglieder.

Diese Beispiele in der Projektanalyse zeigen, dass die Peer-Bewertung gewisse Aspekte nicht abbilden kann, die letztendlich nur durch Dozentenbeobachtungen oder die Analyse des Git-Repositories ergänzt werden können.

In Bewertungsschemata mit einer gemeinsamen Projektnote und der Anpassung der Einzelnoten aufgrund einer Peer-Beurteilung neigen Projektteams manchmal zu Quick-and-Dirty-Hacks, da der Fokus auf dem Produkt liegt. Abweichungen der Einzelnoten bewegen sich häufig im Rahmen ± 1 Note. Demgegenüber werden die individuellen Noten mit dem hier vorgestellten Schema breiter gestreut, wie die besten und schlechtesten Noten der Projekte in Tabelle 5 zeigt. während in 90% der Projekte mindestens eine sehr gute Individualnote vergeben wurde, gab es in 11 Projekten durchgefallene Student:innen und in 17 Projekten als schlechteste Note eine nur ausreichende Bewertung.

5 Fazit und Ausblick

Die große Datenbasis aus sechs Jahrgängen zweisemestriger Software-Entwicklungsprojekte erlaubt tiefe Einblicke in die Gestaltung und Auswirkung des gewählten Bewertungsschemas. Aufwand und Ergebnisse haben einen stabilen Zustand erreicht, der demonstriert, dass diese Art der Bewertung routiniert durchgeführt werden kann. Dennoch besitzt es eine Modularität, die leicht auf andere Begebenheiten angepasst werden kann, wie auch das vorgestellte Schema immer wieder in Einzelmetriken leicht modifiziert wurde.

Ist ein Bewertungsschema den Prüflingen transparent, lenkt es deren Fokus – wie in der Physik beeinflusst auch hier der Beobachtungsprozess das Experiment. So wird beim vorgestellten Bewertungsschema den Student:innen klar vermittelt, dass der Prozess der Software-Entwicklung im Team benotet wird. Konsequenterweise können einzelne Beobachtungen in Bewertungen mit einem anderen Fokus gegebenenfalls nicht reproduziert werden.

Die Betrachtung der Einzelmetriken zeigt, dass die in den Projekten durchgeführte reine Peer-Bewertung das Gesamtpaket der Metriken nicht ersetzen kann. Es werden jeweils unterschiedliche, eher nicht korrelierte Aspekte gemessen. Während in diesem Paper dies im Rahmen eines komplexen Bewertungsschema gezeigt wurde, kam Buffardi [Bu20] mit einem orthogonalen Ansatz zum selben Resultat. Er hat innerhalb einer Peer-Bewertung mit der CATME-Technik [Oh12] festgestellt, dass Analysen des Git-Repositories die Peer-Bewertung anreichern können.

Literaturverzeichnis

- [AR09] Arafat, Oliver; Riehle, Dirk: The comment density of open source software code. In: 31st International Conference on Software Engineering, ICSE 2009, Companion Volume. IEEE, S. 195–198, 2009.
- [Bu20] Buffardi, Kevin: Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories. In: The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20). ACM, New York, S. 650–656, 2020.
- [Co06] Coppit, David: Implementing Large Projects in Software Engineering Courses. *Computer Science Education*, 16(1):53–73, 2006.
- [Hu13] Hummel, Oliver: Transparente Bewertung von Softwaretechnik-Projekten in der Hochschul-lehre. In (Spiller, A.; Lichter, H., Hrsg.): Tagungsband 13. Workshop SEUH Software Engineering im Unterricht der Hochschulen. S. 103–114, 2013. <http://ceur-ws.org/Vol-956/>.
- [Ma17] Mahler, Dirk: Shadows Of The Past: Analysis Of Git Repositories. <https://jqassistant.org/shadows-of-the-past-analysis-of-git-repositories/>, 2017.
- [Oh12] Ohland, Matthew W.; Loughry, Misty L.; Carter, Rufus L.; Bullard, Lisa G.; Felder, Richard; Finelli, Cynthia J.; Layton, Richard; Schmucker, Douglas G.: The Comprehensive Assessment of Team Member Effectiveness: Development of a Behaviorally Anchored Rating Scale for Self- and Peer Evaluation. *Academy of Management Learning & Education*, 11(4):609–630, 2012.
- [To15] Tornhill, Adam: *Your Code As a Crime Scene*. O'Reilly, Sebastopol, CA, 2015.
- [We16] Weicker, Karsten: A Metric-Based Point System for Grading Individual Performance in Software Engineering Projects. In (Hagel, Georg; Mottok, Jürgen, Hrsg.): *European Conference on Software Engineering Education ECSEE*. Shaker Verlag, S. 231–244, 2016.
- [WW09] Weicker, Karsten; Weicker, Nicole: Von Häuptlingen und Indianern – Bachelor/Master als Chance. In (Jaeger, Ulrike; Schneider, Kurt, Hrsg.): *Software Engineering im Unterricht der Hochschulen SEUH 11*. dpunkt.verlag, Heidelberg, S. 61–743, 2009.

Software Engineering lehren (Teil 2)

Teaching during the Covid-19 Pandemic — Online Programming Education

Sandro Speth,¹ Niklas Krieger,¹ Georg Reißner,¹ Steffen Becker¹

Abstract: Teaching programming skills is an essential part of the first-semester computer science curriculum. During the Covid-19, it became particularly challenging as, in particular, the essential exercise lessons had to be implemented virtually. Before the pandemic, we implemented didactic concepts like objects-first and gamification with the hamster simulator. They had to be transferred to the virtual setting as good as possible. In this paper, we report on the implementation of our virtual course programming and software development. In particular, we report on the lessons we learnt from our course implementation, which, overall, was relatively successful.

Keywords: Digital Lectures; Programming Education; Intelligent and Automated Tutoring System

1 Introduction

In every computer science curriculum, students have to gain programming skills in the first semester. At the University of Stuttgart, we usually teach programming in a course with 180 minutes lecture time and 90 minutes exercise time per week. On average, about 750 students participate in the course. The lecture takes place in a large lecture hall, and the exercises are split into approximately 35 exercise groups taught by paid students as tutors. However, when the Covid-19 pandemic started, we had to move all activities in a virtual setting.

Virtualising such a large and complex setting is a challenging task. We faced several issues we had to address. How to deliver the lecture to the students while still keeping interactive elements such as live discussion breaks during the lecture and allowing for questions? How to compensate for the lack of peer programming in the exercise? How to offer the same amount of tutor time while facing the fact that every activity takes longer in a virtual setting than a physical meeting? How to replace the mock exam the students are required to pass before they are allowed to participate in the actual exam? How to deal with students who had to use our computer lab because of a lack of dedicated hardware? Moreover, how to keep in contact with the students and monitor their learning progress?

In our course, we already had implemented a couple of modern didactic methods. We taught objects first, as characterised by B. Meyer in his book “Touch of Class” [Me09]. We used

¹ University of Stuttgart, Institute of Software Engineering, Universitätsstraße 38, 70569 Stuttgart, Germany
{sandro.speth,niklas.krieger,georg.reissner,steffen.becker}@iste.uni-stuttgart.de

our variant of the hamster simulator², a mini programming world to implement the objects first approach in Java, including BlueJ at the beginning of the course.

In this paper, we report how we faced the characterised challenges. We outline in detail our implemented measures to countermeasure each issue. The implemented actions include the use of (1) a dedicated interactive and video-based e-Learning book for the lecture, (2) virtual exercises in Microsoft Teams with breakout rooms including the use of CodeTogether, (3) the use of the ArTEMiS exercise management system [KS18] which we extended towards becoming an intelligent tutoring system, and (4) a YouTube live stream to keep the connection to the students alive. We described the concrete hardware and software setup with costs in a GitHub wiki³. Our lessons learned are many-fold. Several implemented countermeasures worked surprising well. Others did not succeed as planned. In the end, we deem that many lessons learnt can be reused for hybrid or blended learning setups.

The remainder of the paper details our implemented countermeasures (cf. Section 2). In Section 3 we outline our main lessons learnt before we conclude the paper in Section 4.

2 Transformations

Transformation I: Lecture Due to the Covid-19 pandemic and the resulting need for students to attend lectures from home, traditional lectures had to be converted into digital formats. Consequently, many lecturers decided to conduct their lectures virtually via conferencing tools such as Zoom or Microsoft Teams. However, it is challenging for many students to attend an entire lecture concentrated on a video call on the screen without losing attention and missing essential details. Therefore, other lecturers tended to pre-record their lectures and make the videos available to students for self-study from home. Nevertheless, the quality of the recordings can vary greatly. Additionally, students have difficulties finding a specific topic in a long video. For this reason, we have broken down the content of our lectures into short videos of around 5–10 minutes in length. To ensure that the amount of videos is available to the students in a structured way and that they can quickly find the content they are looking for, we use the *learning module* plugin of the ILIAS e-learning platform. In an ILIAS learning module, it is possible to create chapters and pages similar to a book. A lecturer can fill the pages with content such as text, images or videos. Furthermore, the learning module shows the progress through colour coding, making it easier for the student to identify which pages have not yet been viewed. Moreover, there is a page at the end of each chapter for checking the learning status with quizzes which, if passed, unlocks the next chapter and can be repeated as often as desired. We built a setup with a green screen, 4k video camera, external screen, podcast microphone, and various lighting in a soundproof basement for the video recording. The videos were recorded with Open Broadcast System (OBS) and then post-edited with Adobe Premiere and Adobe After Effects to improve the image and sound quality.

² <https://github.com/SQAHamster/plain-java-hamster>

³ <https://github.com/spethso/Remote-Teaching-Setup/wiki>

Transformation II: Central Lecture Exercise In addition to the lecture, students could voluntarily attend the weekly lecture exercise before the pandemic, in which we explained selected topics from the lecture in detail. Another significant part of the lecture exercise consisted of shorter tasks to discuss the respective topics and the opportunity to ask questions centrally. For the content parts, we created videos and an ILIAS learning module similar to the lecture. However, the question and discussion part, in particular, depends on the active participation of the students, which is why an asynchronous alternative was needed. Since the overall number of students in a lecture exercise usually exceeds the maximum number of participants of Microsoft Teams, Cisco Webex, and Zoom Calls, we decided to stream the synchronous part of the lecture exercise via YouTube for one lecture hour per week. The streaming was done with OBS from the video room, and the students could ask anonymous questions via chat which were discussed on the iPad in a digital whiteboard by us. Moreover, several tutors managed the active stream of questions in the chat and answered some pending questions. Since YouTube automatically records live streams, students could also watch them at any time later.

Transformation III: Introduction Lecture A significant disadvantage of digital teaching is that first-year students can no longer see their lecturers in person and get to know them inside the auditorium. However, this is important for most students so that the lecturer no longer remains the unknown magical being behind the lessons. Furthermore, students usually ask many organisational questions in the first lecture. In a purely asynchronous lecture setting, these can only be asked via a forum or email. To overcome both challenges, we streamed two lectures live via YouTube at the beginning of the semester. The first live stream served to explain the organisational framework and clarify questions. In the second live stream, “Meet-your-Prof”, students were able to ask us various questions, including funny ones, to get to know us better. The event thus served as an ice breaker so that students could experience university life in the online setting during the pandemic and motivate them to ask questions.

Transformation IV: Exercises The most important part of learning programming is practice. However, traditional on-site group exercises supervised by a tutor are not possible in a digital setting. For this reason, we conducted our synchronous live exercise sessions virtually via Microsoft Teams, including breakout groups with two students per team, through which the tutor could iterate. We decided on Teams as it is free for universities and schools. Both students in a breakout group work on the programming tasks together through pair programming. Since participants could not switch between groups independently in the breakout feature of Microsoft Teams, which we required for various discussion activities, we decided to simulate breakout rooms artificially in Microsoft Teams via a team and several channels. Additionally, as the students do not sit together in front of a computer as they would in a face-to-face exercise, they could use the collaboration tool CodeTogether⁴. Using

⁴ <https://www.codetogether.com/>

CodeTogether, one team member hosted an IDE session, and the other member joined via a shared link either in their IDE using a plugin or the browser. CodeTogether allowed both students to work on the same project simultaneously. One student transferred his IDE as a screen share to enable the tutor to identify functional and stylistic errors, thus, enabling the tutor to view the current work without interference. Furthermore, the students could check their tasks on their own using ArTEMiS as a virtual tutor, which is explained in Sect. 2.

To successfully be allowed to participate in the course's final exam, the students were required to attend and pass the exercises and succeed in four quizzes on ILIAS as a replacement for a mock exam written before the end of the semester to prepare students for the actual exam. They were released regularly over the semester and checked that all students understood the topics from the lectures. The students had a window of two hours for each of the quizzes on the day they could work on it. During these two hours, they could start the quiz exactly once, from which point they had 30 minutes to complete and submit it. To pass a quiz, the students had to achieve at least 50% of the available points, which between ~87–100% of those who attempted the quiz did over the four quizzes.

To also provide the computer science teaching qualification students with valuable reflective pedagogical practice [BBF20], we implemented the described extra tasks and separate tutorial groups for all pre-service teacher students in the course.

Transformation V: ArTEMiS as Virtual Tutor Live tutor feedback is one of the critical aspects of every exercise. Despite all efforts spent on online tutoring, there was less interaction between students and tutors than offline exercises. Therefore, we used ArTEMiS⁵, an automatic assessment management system developed at the Technical University Munich [KS18], to give additional feedback on exercises. Our setup consists of the main system, a GitLab plugin to store the student repositories, and a Jenkins plugin to automatically evaluate the students' code. After the students registered an account and joined our course on our ArTEMiS website, they could see the list of already existing exercise sheets. When starting an exercise sheet, the system creates a git repository, which the student can clone to work on the exercises. After each push to the remote repository, ArTEMiS automatically executes tests, and the student can see how many tests succeeded for each exercise. If tests failed, we also provided feedback on the cause of the failure. In order to provide an additional incentive to use ArTEMiS, we decided to provide tests both for non-graded and some graded exercises. This allows students to get automatic feedback on their exercises before submitting the final solution to ILIAS without a tutor manually reviewing the code. However, we made sure that the students were not able to see the source of the tests by using the Ares library⁶, a JUnit extension. This library also prevented the students from executing several types of malicious code on our test servers by i.a., preventing network or file system access and adding hard time limits to test cases. Additionally, the test results helped us identify the student's open fields of learning.

⁵ <https://github.com/lslintum/ArTEMiS>

⁶ <https://github.com/lslintum/Ares>

Transformation VI: Online IDE In regular (presence) semesters, students, who did not own powerful enough computing hardware or any computing hardware which is compatible with a JDK or one of the used IDEs, had the opportunity to participate in those exercises held in a computer pool at the university and use the provided hardware. As this was not a possibility when the university's buildings were not accessible to students, an alternative way of providing compatibility for lower-spec devices was needed. The original plan to use "Microsoft Visual Studio Codespaces", a VS Code instance hosted in an Azure Cloud usable with any Microsoft Account, was not realizable due to Microsoft renaming the product to "GitHub Codespaces" and moving it into an (at that point in time) closed Beta. As a replacement, we created a similar application called "SQA CodeOnline" which we now host for free for all course students. It is based on the open-source product "code-server"⁷ which hosts a VS Code Server and provides access to the GUI through the browser. After a student logs in on the welcome page, an individual docker container, based on a custom image in which code-server and all needed tools are installed, is created and assigned a virtual filesystem of 150MB for the student's files. Within those constraints, the students can use their containers for regular development. As the hamster simulator which we use in the course initially relies on a JavaFX GUI and the container the students are constrained to do not provide a window server, an alternative user interface was added. An extension for VS Code, which is pre-installed in the docker image, connects to a local HTTP server which the hamster simulator automatically opens if it detects that it is running on SQA CodeOnline⁸. The GUI in the extension is mainly a web view displaying a browser version of the GUI (including all controls). This gives the student the full ability to work on the exercises from any device (even mobile ones), just like others who work on VS Code locally. They can install extensions etc., to customize their editing experience. Because it stores the files on a remote server, they are synchronized between all used devices.

3 Lessons Learned

As demonstrated in the sections above, it is very well possible to transform even a more extensive lecture course into a purely online course without losing too much of the value for the students. Some formats even have a better reception by students than the traditional presence course. For one, the short videos, structured in the ILIAS e-Learning Module, were rated more positively by the students. Even contrary to many experiences in other online courses, the interactivity was (for some of the formats) even higher than what was experienced in presence. The most interaction occurred on the YouTube streams of the central lecture exercises where students were asking questions and interacting more freely than what they would in a lecture hall in their presence. Less interaction and challenges how to help the students arose, however, in the tutor exercises. Learning programming needs social interaction. However, students often join the meetings without saying or writing much. Of course, all the added tools and support cannot replace that and engage all those

⁷ <https://github.com/cdr/code-server>

⁸ <https://github.com/SQAHamster/VSCodeHamster-Mirror>

involved in teaching. While ArTEMiS provided valuable feedback, several students had difficulties using git. That is why some students, esp. those needing feedback the most, decided not to use the system. However, this effect can be mitigated by providing a better IDE integration to make using ArTEMiS easier. Therefore, we are integrating Orion⁹, an IntelliJ IDEA extension for ArTEMiS, into the lecture for winter semester 2021/2022.

4 Conclusion and Outlook

In this paper, we present the implementation of our virtual introductory course to programming. We outline how we tackle the various challenges of the virtual setting. Finally, we report the lessons we have learnt during our journey.

The lessons we learned can help course instructors implement hybrid programming courses in a blended learning approach. In doing so, instructors can combine the best of the virtual teaching environment with the physical setup.

In the future, we will try to reorganize our programming course and move it closer towards a sound blended learning setup. Furthermore, we work on the implementation of our intelligent tutoring system. This intelligent tutoring system (named IT-REX) should better track the gathered skills of the students, adapt the exercised to the progress and give better hints to typical programming mistakes.

References

- [BBF20] Becker, S.; Bescherer, C.; Fest, A.: Reflective Pedagogical Practice on and in Introduction to Programming and Software Engineering. In: Tagungsband des 17. Workshops “Software Engineering im Unterricht der Hochschulen” 2020, Innsbruck, Österreich, 26. - 27.02.2020. Vol. 2531, CEUR-WS.org, pp. 7–10, 2020.
- [KS18] Krusche, S.; Seitz, A.: ArTEMiS: An Automatic Assessment Management System for Interactive Learning. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. SIGCSE ’18, Association for Computing Machinery, pp. 284–289, 2018.
- [Me09] Meyer, B.: Touch of Class. Learning to Program Well with Objects and Contracts 51/, 2009, URL: <https://link.springer.com/book/10.1007/978-3-540-92145-5>.

⁹ <https://github.com/lslintum/Orion>

Virtuelle Präsenz? – Fallbeispiel eines Flipped Classrooms während der Covid-19-Pandemie

Christin Voigt,¹ Nicole Draxler-Weber,² Uwe Hoppe³

Abstract: Die Covid-19-Pandemie gilt als Treiber digitaler Lehre. Zugleich stellt sie das hybride Lehrformat Flipped Classroom (FC) vor die Herausforderung, dessen interaktive Präsenzphase in den virtuellen Raum zu verlagern. In diesem Beitrag wird ein Fallbeispiel dieser virtuellen Präsenz zu Zeiten der Covid-19-Pandemie präsentiert und die Stärken und Schwächen des Konzepts herausgearbeitet. Die Evaluation zeigte auf, dass zukünftig eine Mischung aus Präsenz- und Distanzlehre bevorzugt wird, wenngleich im Fallbeispiel der Präsenzphase in räumlicher Synchronität (Präsenzlehre) oder Asynchronität (Distanzlehre) unterschiedliche Vorteile nachgesagt wurden.

Keywords: Flipped Classroom; Präsenzlehre; Distanzlehre; Covid-19-Pandemie; Fallbeispiel

1 Einleitung

Zur Eindämmung des Coronavirus SARS-CoV-2 wurde im Frühjahr 2020 das öffentliche Leben weltweit eingeschränkt. Der Großteil der Hochschulen wurde geschlossen, sodass gewohnte Präsenzformate nicht mehr stattfinden konnten. Um den Lehrbetrieb aufrecht zu erhalten, musste auf digitale Formate zurückgegriffen werden. Eines der Formate ist das Konzept des Flipped Classrooms (FC), das aus einer Onlinephase zum eigenständigen Wissenstransfer und einer Präsenzphase zur gemeinsamen Wissensvertiefung besteht [HS17]. Studien zeigen, dass das Konzept des FC mit der Pandemie eine stärkere Nutzung erfahren hat [Co21]. Jedoch handelt es sich beim FC im ursprünglichen Sinne um ein hybrides Format, bestehend aus einer räumlich asynchronen Onlinephase und einer räumlich synchronen Präsenzphase im Plenum. Mit der Schließung der Hochschulen konnte eine gemeinsame Präsenzphase nicht mehr stattfinden. Stattdessen war eine alternative Umsetzung der Präsenzphase in digitaler Form notwendig [SDA20]. Folgende Forschungsfragen (FF) sollen in dieser Arbeit beantwortet werden:

FF1: Wie kann eine Veranstaltung im Rahmen des FC-Konzepts mit Online- und Präsenzphase in vollständiger Distanzlehre aussehen?

FF2: Welche Vor- und Nachteile gehen mit einem FC in vollständiger Distanzlehre einher? Zur Beantwortung von FF1 wird zunächst ein Fallbeispiel eines FC während der Covid-19-Pandemie aufgezeigt, das an einer Universität im Sommersemester 2021 in der dargestellten Form stattgefunden hat. Im Rahmen von FF2 werden die Stärken und Schwächen dieses

¹ Universität Osnabrück, BOW, Katharinenstraße 1-3, 49074 Osnabrück, christin.voigt@uni-osnabrueck.de

² Universität Osnabrück, BOW, Katharinenstraße 1-3, 49074 Osnabrück, nicole.draxler-weber@uni-osnabrueck.de

³ Universität Osnabrück, BOW, Katharinenstraße 1-3, 49074 Osnabrück, uwe.hoppe@uni-osnabrueck.de

Konzepts aus Studierendensicht qualitativ erhoben und diskutiert, woraufhin die Frage nach zukünftigen Veranstaltungsformaten aufgeworfen wird. Hierzu wird zunächst qualitativ erhoben, ob die Präsenzphase des FC aus Sicht der Studierenden zukünftig in Distanz- oder Präsenzlehre gestaltet werden sollte. Zusätzlich werden die Teilnehmenden der Veranstaltung qualitativ befragt, welche Bedingungen der zeitlichen Synchronität sie als Vorteil bewerten würden. Die Ergebnisse unserer Arbeit zeigen auf, wie ein FC-Konzept bei vollständiger Distanzlehre ausgestaltet werden kann und wie Studierende diese Art der Umsetzung erleben.

2 Theoretische Grundlagen

Die voranschreitende Digitalisierung veränderte nicht nur das tägliche Leben, sondern zunehmend auch die Bildung an Schulen und Universitäten [HS17]. Daraus entstanden hybride Bildungskonzepte wie der FC [HS17; Ka20]. Im FC wird die klassische Struktur der Lehr-Lern-Aktivitäten umgedreht. Während in klassischen Lernumgebungen die Wissensvermittlung beispielsweise in Vorlesungen gemeinsam erfolgt, wird im FC der gemeinsamen Zeit eine Onlinephase vorgelagert. In dieser Onlinephase werden den Studierenden digitale Medien wie Videos, Podcasts oder Selbsttests zur Wissensvermittlung zur Verfügung gestellt, die oftmals in ein Lernmanagementsystem (LMS) eingebunden sind [BS12; HS17; KO19]. Die Lernenden erarbeiten somit die theoretischen Inhalte selbstständig, bevor sie mit dem Lehrenden in der Präsenzphase zusammentreffen. Die frei gewordene gemeinsame Zeit kann nun unter Anwendung interaktiver und teilnehmeraktivierender Methoden für die Erreichung höherer Lernziele genutzt werden, etwa durch Projekt- und Gruppenarbeiten oder der Bearbeitung praktischer Fallstudien [BS12; HS17]. Da Lehrende und Lernende während der Onlinephase zu einer unterschiedlichen Zeit an unterschiedlichen Orten lehren und lernen, findet sie asynchron statt, wohingegen die Präsenzphase typischerweise zur selben Zeit am selben Ort als synchrone Lehre durchgeführt wird [MM08]. Der FC vereint die Vorteile der synchronen und asynchronen Lehre, weshalb er auch als eine Form hybrider Lehre bezeichnet wird [Ka20]. Dabei werden dem FC in der Literatur bereits einige positive Effekte zugeschrieben. Einerseits ermöglicht er durch die Bereitstellung von Online-Materialien ein zeit- und ortsunabhängiges Lernen mit einer höheren Flexibilität [BV13]. Andererseits können positive Auswirkungen auf den Lernerfolg beobachtet werden [GKC14]. Durch die Covid-19-Pandemie und die damit verbundenen Schließungen der Hochschultore erfahren digitale Lehrformate zwar einen starken Aufwind, doch der FC wurde vor eine Herausforderung gestellt [SPT20]. Denn die politischen Reaktionen auf die Ausbreitung des Coronavirus verhinderten eine physische Anwesenheit im Hörsaal und zwangen Lehrende damit, die Präsenzphase des FC mit ihren interaktiven Unterrichtsmethoden in den virtuellen Raum zu verlagern [AAA20; Yu20]. Ob die Präsenzphase des FC nach einer zeitlichen und räumlichen Synchronität verlangt, bleibt unklar. Laut Myrach und Montandon ist lediglich die zeitliche Synchronität entscheidend [MM08]. Bergman und Sams implizieren durch ihre Abgrenzung der Zeit „in the classroom“ (S.13) von der Zeit „at home“ (S.13) auch eine örtliche Synchronität [BS12]. Stöhr et al. unterscheiden

hingegen zwischen einem konventionellen FC mit einer zeitlich und räumlich synchronen Präsenzphase und einem virtuellen FC mit einer zeitlich synchronen und einer räumlich asynchronen Präsenzphase [SDA20]. Eine Studie von Hew et al. vergleicht das Engagement der Studierenden in einem konventionellen und virtuellen FC und kann keine signifikanten Unterschiede feststellen [He20]. Handke und Sperl betonen grundsätzlich die Bedeutung interaktiver Gruppenarbeiten für das Gelingen des FC, die durch eine gemeinsame Diskussion mit dem Sitznachbarn gewährleistet werden können [HS17]. In der reinen Distanzlehre, wie sie zu Zeiten der Covid-19-Pandemie stattfand, liegt die Herausforderung des FC demnach darin, die interaktiven Unterrichtsmethoden in eine Art „virtuelle Präsenz“ zu überführen.

3 Fallbeispiel

Um FF1 zu beantworten, wird im Folgenden eine universitäre Veranstaltung vorgestellt, die im Sommersemester 2021 als virtueller FC durchgeführt wurde. Zu dieser Zeit wurde an der Universität vollständig auf Präsenzveranstaltungen verzichtet. Inhaltlich wurden u.a. verschiedene Organisationstheorien, Bedingungen und Gestaltungsparameter der Organisationsstruktur sowie der organisatorische Wandel thematisiert. Besucht wurde die Veranstaltung hauptsächlich von Studierenden der Wirtschaftswissenschaften und -informatik im Bachelor sowie von Studierenden der Rechtswissenschaften. Für diese Studiengänge gilt sie als Pflichtveranstaltung. Mit ca. 300 Teilnehmenden im Sommersemester 2021 ist sie eine universitäre Großveranstaltung [Bo17]. Studierende der Wirtschaftswissenschaften und -informatik legten am Ende des Semesters eine 60-minütige benotete Klausur ab, während die 80 Studierenden der Rechtswissenschaften eine Teilnahmebescheinigung erhielten. Die Struktur der Veranstaltung wird in Abbildung 1 dargestellt.

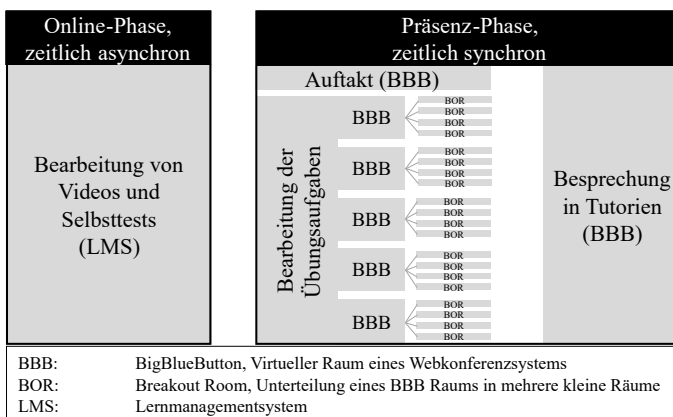


Abb. 1: Struktur der Veranstaltung

Im Rahmen der Onlinephase wurden den Studierenden im universitätseigenen Lernmanagementsystem (LMS) innerhalb der virtuellen Lernumgebung „Courseware“ verschiedene

Materialien zur eigenen, zeitlich asynchronen Bearbeitung angeboten. Zu Beginn wurden ein Erklärvideo zur technischen Nutzung sowie Informationen über den Ablauf der Veranstaltung bereitgestellt. Anschließend erfolgte die wöchentliche Freischaltung verschiedener Materialien. Dazu wurden die Vorlesungsinhalte in insgesamt 32 Videos aufgeteilt, wovon eines von einem Praxispartner bereitgestellt wurde. Die Videos waren jeweils zwischen zehn und 40 Minuten lang. Zu jedem Video wurden die Lernziele kommuniziert und mehrere Selbsttests angeboten, mit denen die Studierenden ihr Wissen überprüfen konnten. Studierende der Rechtswissenschaften erhielten durch die Teilnahme an den Selbsttests ihre Teilnahmebescheinigung der Veranstaltung und konnten darüber hinaus freiwillig an der Präsenzphase teilnehmen. Für Studierende der Wirtschaftswissenschaften und -informatik galt das Anschauen der Videos als Voraussetzung für die Teilnahme an der zeitlich synchronen Präsenzphase. Zusätzlich waren sie in einem wöchentlichen Tutorium eingeschrieben. Es wurden insgesamt zehn 90-minütige Tutorien angeboten. Die Präsenzphase wurde online über das Webkonferenzsystem „BigBlueButton“ (BBB) durchgeführt. Ziel der Präsenzphase war die Bearbeitung von Übungsaufgaben in Kleingruppen, die inhaltlich auf den Videos der Courseware aufbauten. Grundlage für die Übungsaufgaben stellte eine fiktive, für die Veranstaltung entwickelte Fallstudie dar. Zunächst wurde als *Auftakt* in der Großgruppe das aktuelle Übungsblatt inklusive dessen Lernziele vorgestellt und erste Fragen wurden beantwortet. Anschließend wechselten die Studierenden in fünf BBB-Gruppen mit jeweils 30-50 Personen. Jede der fünf BBB-Gruppen wurde wieder in kleinere Breakout-Räume (BOR) unterteilt. In den BOR formierten sich somit Kleingruppen mit 3-5 Personen, die in der vorgegebenen Zeit die Aufgaben bearbeiteten. Die Teilnehmer der BOR-Gruppen variierten dabei wöchentlich. Während der Bearbeitungszeit besuchten die Dozenten der Veranstaltung abwechselnd alle Kleingruppen und boten Hilfestellungen an. Mit Ablauf der Bearbeitungszeit wurden die Abgaben der Kleingruppen im LMS hochgeladen und dem Tutor zugänglich gemacht. In insgesamt zehn Tutorien mit einer Größe von bis zu 30 Personen wurden schließlich die Abgaben besprochen und Bonuspunkte verteilt, die sich die Studierenden auf die abschließende Prüfungsleistung in Form einer Klausur anrechnen lassen konnten. Den Tutoren wurden eine Musterlösung und Bewertungshinweise bereitgestellt. Diese Musterlösungen wurden allerdings nicht an die Studierenden weitergegeben. Stattdessen wurde individuell auf die Abgaben eingegangen. Am Ende des Semesters fand eine Online-Klausur statt, die auf den Lernzielen der Veranstaltung aufbaute und sowohl Inhalte der Online- als auch der Präsenzphase beinhaltete.

4 Analyse

4.1 Datenerhebung und Methodik

Zur Beantwortung der FF2 wurde eine Befragung durchgeführt. Die Befragung fand in dem letzten zeitlich synchronen Zusammentreffen innerhalb des *Auftakts* (vgl. Abb.1) der Veranstaltung statt, sodass sichergestellt war, dass die Studierenden den FC vollständig kennengelernt haben. Sie wurde online über LimeSurvey durchgeführt und enthielt sowohl

offene als auch geschlossene Fragen. Insgesamt nahmen 96 Studierende an der Befragung teil, deren demografische Daten in Tabelle 1 zusammengefasst werden. Es wurden drei

Tab. 1: Demografische Daten

Charakteristik				Charakteristik			
		Absolut	Prozent			Absolut	Prozent
Geschlecht	Männlich	53	55,2	Semester	1. Semester	1	1,0
	Weiblich	36	37,5		2. Semester	11	11,5
	n.a.	7	7,3		3. Semester	1	1,0
	Gesamt	96	100		4. Semester	63	65,6
Alter	19	6	6,3		6. Semester	5	5,2
	20	20	20,8		> 6. Semester	8	8,3
	21	20	20,8		n.a.	7	7,3
	22	8	8,3		Gesamt	96	100
	23	16	16,7	Studiengang	Informatik	1	1,0
	24	8	8,3		Wirtschaftsinformatik	15	15,6
	25	4	4,2		Wirtschaftswissenschaften	68	70,8
	27	1	1,0		Wirtschaftsrecht	4	4,2
	43	1	1,0		Sonstige	1	1,0
	n.a.	12	12,5		n.a.	7	7,3
	Gesamt	96	100		Gesamt	96	100

offene qualitative Fragen verwendet. Die Befragten wurden in den ersten beiden Fragen zunächst aufgefordert, positive und negative Aspekte der virtuellen zeitlich synchronen Präsenzphase zu nennen. Insgesamt konnten jeweils bis zu neun positive und negative Aspekte genannt werden. Neben der Nennung und Beschreibung dieser Kriterien wurde auch eine Wertung nach Relevanz für jedes Kriterium gefordert, die von 1 (ist mir sehr unwichtig) bis 5 (ist mir sehr wichtig) reichte und in einer Matrix eingetragen wurde: *Was hat Ihnen an der synchronen Online-Vorlesung gefallen bzw. nicht gefallen? Bitte nennen und erläutern Sie wenigstens ein Kriterium und bewerten Sie Ihre Aussagen nach Relevanz von 1 bis 5.* Die dritte offene Frage lautete: *Nehmen Sie an, das aktuelle Veranstaltungskonzept der synchronen Online-Vorlesung würde auf die Präsenzvorlesung im Hörsaal übertragen werden. Welche Unterschiede vermuten Sie zwischen einer Präsenz- und einer Online-Vorlesung?* Die Studierenden haben dabei eine Beschreibung des Unterschieds, eine Wertung, ob der genannte Unterschied für Präsenz- oder Distanzlehre vorteilhaft ist, sowie eine Relevanzbewertung von 1 bis 5 vorgenommen. Die offenen qualitativen Fragen wurden nach der qualitativen Inhaltsanalyse nach Mayring und der induktiven Kategorienbildung ausgewertet und folgten den Schritten Paraphrasierung, Generalisierung, Reduktion und Subsumtion [MF14]. Es wurden einheitliche Kodierregeln festgelegt, die die Nennungen in Unter- und Oberkategorien einteilten. Neben den offenen Fragen wurden zwei geschlossene Fragen gestellt und deskriptiv ausgewertet: *Welche der folgenden Optionen würden Sie sich für zukünftige Veranstaltungen wünschen: Präsenzlehre, Distanzlehre oder eine Mischung aus beidem?* und *Hätte man die synchrone Online-Vorlesung im Hörsaal durchgeführt, hätte ich mich wohler gefühlt.* Während die erste quantitative Frage durch Auswahlboxen zu beantworten war, wurde für die zweite Frage eine 5-stufige Likert Skala verwendet.

4.2 Ergebnisse

Die zweite Forschungsfrage (FF2) adressiert zunächst die Vor- und Nachteile der virtuellen Präsenz im vorgestellten Fallbeispiel. Um FF2 zu beantworten, wurden die qualitativen Fragen ausgewertet. Dabei wurden die Studierenden aufgefordert, positive und negative Aspekte der virtuellen Präsenzphase zu benennen und diese anschließend auf einer Skala von 1 bis 5 nach Wichtigkeit zu bewerten. Je höher die Wertung, desto wichtiger wurde der Aspekt empfunden. Tabelle 2 fasst die positiven Nennungen zusammen. Unter Σ werden die Anzahlen der absoluten Nennungen zusammengefasst, M beschreibt die durchschnittliche Wertung der Nennung und SD die Standardabweichungen aller Wertungen der Unterkategorien. Auffällig ist, dass insgesamt mehr positive als negative Kriterien genannt wurden. So stehen 102 negativen Nennungen 152 positive Nennungen gegenüber. Die Nennungen wurden in die Oberkategorien Aufbau, Online Gruppenarbeit, Zeitmanagement, Übungsaufgaben, Kontakt zu Lehrenden und LMS aufgeteilt. Unter der Struktur der Veranstaltung wurde neben dem Aufbau der Themen positiv bewertet, „*dass die Inhalte zu den Videos in der Courseware passten*“. Auch die Verständlichkeit und der zuverlässige Upload der Inhalte im LMS wurden positiv für die Bearbeitung in der Präsenzphase angesehen. Die Qualität der Veranstaltung beinhaltete nicht nur die allgemeine Lehrqualität, sondern auch die gute Tonqualität der Videos und Online-Konferenzen. Die Gruppenarbeit und die Übungsaufgaben dominierten in den Nennungen der Teilnehmenden. Am häufigsten insgesamt wurden der soziale Kontakt und die virtuelle Gruppenarbeit positiv bewertet. Begünstigt wurde der soziale Kontakt, der den Austausch mit anderen förderte, durch die Zufälligkeit der Gruppenzusammensetzung. Darüber hinaus wurde die Verbindung aus Flexibilität bei der Bearbeitung und festem Zeitplan gelobt, denn „*regelmäßige Abgabetermine helfen, am Ball zu bleiben*“. Die höchste durchschnittliche Bewertung (4,6) erhielt die Unterstützung durch die Lehrenden bei der Bearbeitung der Übungsaufgaben. Die zweithöchste Wertung erhielten mit 4,5 zugleich die angemessene Schwierigkeit der Aufgabenblätter und die Struktur der Courseware im LMS. Auch der Anreiz zur Teilnahme über die Bonuspunkte wurde mit 16 Nennungen häufig als positiver Aspekt aufgezählt. Die negativen Aspekte der Veranstaltungen werden in Tabelle 3 dargestellt. Die Oberkategorien Aufbau, Online Gruppenarbeit, Zeitmanagement und Übungsaufgaben tauchten sowohl bei den positiven als auch bei den negativen Nennungen auf. Darüber hinaus wurde der Wunsch nach weiteren Materialien geäußert. Zunächst wurde der frühe Vorlesungsbeginn um 8:15 Uhr genannt, der mit einem Mittelwert von 3,8 die drittstärkste durchschnittliche Wertung aller Unterkategorien aufzeigte. Am stärksten diskutiert wurde die Online-Gruppenarbeit als negativer Aspekt. Die Gruppenarbeit wurde von vier Personen grundsätzlich abgelehnt, die eine Einzelarbeit bevorzugten. Entgegen der zuvor genannten Möglichkeit, technische Tools bei der virtuellen Gruppenarbeit nutzen zu können, empfanden fünf Personen die Bearbeitung im virtuellen Raum insbesondere bei längeren Aufgaben als anspruchsvoll. Auch technische Probleme traten beispielsweise durch schlechte Mikrofone auf und erfuhren die zweithöchste Wertung. Am häufigsten insgesamt wurde jedoch der Trittbrettfahrer-Effekt genannt, denn „*viele haben wenig bis gar nichts gesagt, aber natürlich schreibt man auch diese mit auf die Zettel, man ist ja nett*“. Zusätzlich wurde die Abhängigkeit von den Gruppenmitgliedern bemängelt. Zugleich

Tab. 2: Positive Nennungen

Oberkategorien	Unterkategorien	Σ	M	SD	Ankerbeispiel
Aufbau	Struktur der Veranstaltung	4	3,8	0,4	„ich fand insgesamt die Kombination aus Videos, Übung und Tutorien hilfreich“
	Verständlichkeit	4	4	0,7	„Ausführliche Informationen zum Ablauf“
	Qualität	4	4	1,2	„Gute Qualität [der Veranstaltung]“
	Zuverlässiger Upload	4	3,3	1,1	„Pünktlicher & zuverlässiger Upload“
Online Gruppenarbeit	Sozialer Kontakt	21	3,9	1,3	„Es war/ist die einzige Veranstaltung, wo man aktiv im Austausch mit anderen Studierenden war“
	Zufälligkeit der Gruppen	3	3,7	1,3	„Zufällige Gruppen“
	Gruppenarbeit generell	18	3,7	1	„Bearbeitung in Gruppen“
	Unterschiedliche Sichtweisen	4	4	1,1	„Ideen mehrerer Personen gehört, die man [...] an dem Punkt noch nicht hatte“
Zeitmanagement	Fester Zeitplan	10	4,2	0,6	„Dass man so gezwungen ist, sich regelmäßig mit den Inhalten zu beschäftigen“
	Flexible Bearbeitung	12	4,3	0,9	„Die Bereitstellung aller Materialien, zur selbständigen Erarbeitung“
	Geringer Zeitaufwand	3	3,3	0,5	„Zeitersparnis“
Übungsaufgaben	Anwendung der Theorie aus dem LMS	13	3,9	1,1	„Man hat das [G]elernte direkt anwenden können in den Aufgaben“
	Praxisbezug	8	3,9	0,8	„Praxisnahe Aufgaben und Bezug auf ein Fallbeispiel“
	Anreiz durch Bonuspunkte	16	4,1	1,3	„Erwerb von Klausurpunkten, Einigen Studierenden fällt das online lernen schwerer, daher fair“
	Feedback über Leistung	1	2	-	„Dass man Feedback bekommt über seinen Leistungsstand“
	Schwierigkeit	2	4,5	0,5	„Angemessenes Niveau der Aufgaben“
Kontakt zu Lehrenden	Auftakt	2	4	1	„Vorherige Vorstellung des jeweiligen Aufgabenblattes“
	Unterstützung bei der Bearbeitung	8	4,6	0,7	„Direkte Beantwortung von Fragen“
LMS	Struktur der Courseware	2	4,5	0,5	„Die gute Struktur in Courseware“
	Videos	7	2,7	1,3	„Sehr ausführliche und gute Videoaufzeichnungen“
	Selbsttests	4	1,5	0,8	„Selbsttest am Ende jedes Videos“

wurde vermutet, dass ein Fehlverhalten von Trittbrettfahrern virtuell schwerer zu überwachen sei. Diese fehlende Kontrolle wurde mit einer durchschnittlichen Bewertung von 5 insgesamt am bedeutsamsten bewertet. Daneben wurde einerseits bemängelt, dass teilweise per Zufallsprinzip ähnliche Gruppenzusammensetzungen zustande kamen, andererseits wurde auch der Wunsch nach gleichen Gruppen geäußert.

Tab. 3: Negative Nennungen

Oberkategorien	Unterkategorien	Σ	M	SD	Ankerbeispiel
Aufbau	Frühes Aufstehen	6	3,8	0,7	„Zu früher Beginn der Veranstaltung“
Online Gruppenarbeit	Generelle Abneigung	4	3	1,6	„Arbeiten in der Gruppe“
	Online Bearbeitung	5	3,2	1,2	„Aufgaben teils schwer online in Gruppenarbeit umsetzbar“
	Technische Probleme	3	4	0,8	„Oft Studenten mit schlechten Mikrofonen“
	Sozialer Kontakt	7	3,4	1,3	„Einsamkeit“
	Abhängigkeit	3	2,7	1,7	„Man ist auf die Gruppe angewiesen“
	Trittbrettfahrer	16	3,8	1,3	„teilweise haben sich Studierende an der Bearbeitung wenig bis gar nicht beteiligt“
	Fehlende Kontrolle	2	5	-	„Freifahrerverhalten schwierig kontrollierbar“
	Gleiche Gruppen	3	3,7	0,5	„Teilweise gleiche Gruppen“
	Zufällige Gruppen	2	3	1	„Zufällige Gruppenzuteilung“
Zeitmanagement	Zu hoher Zeitaufwand	8	3,3	1,3	„Ich finde die Vorlesung ziemlich zeitaufwendig (...)“
	Zeitdruck bei Bearbeitung	15	3,3	1,1	„Manchmal war die Zeit etwas knapp“
Übungsaufgaben	Aufgabenstellung	10	3	1,1	„Aufgaben waren teilweise zu (a)llgemein formuliert“
	Variierender Arbeitsaufwand	5	2,6	1	„steigender Schwierigkeitsgrad der Aufgabenblätter“
Weitere Materialien	Musterlösung	4	3,8	0,4	„keine wirklichen Lösungen zu den Aufgaben“
	Skript	9	3,7	1,4	„Vorlesungsfolien wurden nicht hochgeladen“

Am zweithäufigsten wurde ein Zeitdruck bei der Bearbeitung genannt. Auch der generelle Arbeitsaufwand wurde von acht Studierenden als zu hoch eingeschätzt. Zehn Studierende bemängelten die Aufgabenstellung und berichteten von „*Verständnisschwierigkeiten*“. Zuletzt kam der Wunsch nach weiteren Materialien auf. Die Videos waren mit vielen Grafiken und Bildern versehen, sodass für das Verständnis ein Anhören der Audiospur notwendig war. Es

wurde bewusst darauf verzichtet, ein Vorlesungsskript sowie Musterlösungen zur Verfügung zu stellen. Damit sollte ein Weiterreichen der Materialien über mehrere Semester hinweg verhindert werden. Stattdessen wurden die eingereichten Lösungen individuell besprochen. Neun Studierende bemängelten das fehlende Vorlesungsskript bei der Klausurvorbereitung und vier Studierende wünschten sich eine Musterlösung für die Übungsaufgaben.

Ob die Teilnehmenden der Veranstaltung zukünftig die Distanz- oder Präsenzlehre bevorzugen, wird ebenfalls in FF2 untersucht. Die deskriptive Auswertung zeigt zunächst eine leichte Tendenz zur Präsenzlehre im Vergleich zur Distanzlehre. Die Mischung aus Präsenz- und Distanzlehre wird allerdings von ca. 68 % aller Befragten bevorzugt.

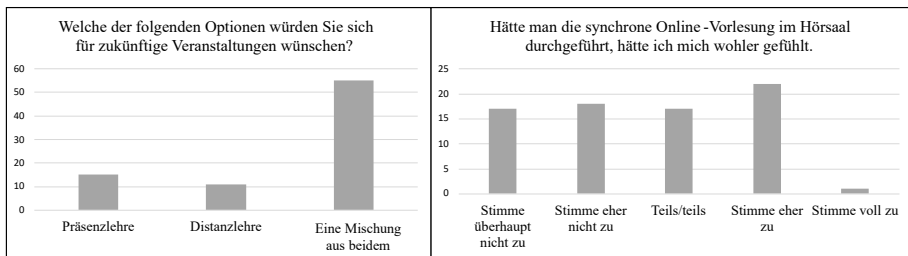


Abb. 2: Vergleich Präsenz- und Distanzlehre

Darüber hinaus wird deutlich, dass die Ansichten darüber, ob die Präsenzphase im Hörsaal das Wohlbefinden erhöht, schwanken. Es ist jedoch zu beachten, dass aktuell auch Faktoren wie die Angst vor einer Ansteckung die Antworten dieser Frage beeinflussen könnten. Abbildung 2 lässt daher bereits erahnen, dass es auf die Frage, welche Form der räumlichen Synchronität bevorzugt wird, keine allgemeingültige Antwort gibt. Welche Aspekte in welchem Veranstaltungsformat präferiert werden, wurde wie in Tabelle 4 dargestellt, in einer qualitativen Befragung geprüft. Dabei verdeutlicht + *Distanzlehre* die Vorteile in Distanzlehre und + *Präsenzlehre* die genannten Vorteile der Präsenzlehre. Sowohl virtuell als auch in Präsenz wurde der Faktor *Zeit* von den Befragten genannt. Neben einem verbesserten Zeitmanagement aufgrund der höheren Flexibilität wurde in einer virtuellen Präsenzphase ein geringer Zeitaufwand vermutet. Vor allem die fehlende Fahrzeit und die erleichterte Teilnahme an der Konferenz führe zu einem „weniger frühen Aufstehen“. Im Hörsaal mit örtlicher Synchronität wurde hingegen ein schnelleres Arbeiten und eine verbesserte Kommunikation als Grund für einen verringerten Zeitaufwand genannt. Dieser äußere sich auch durch eine höhere Beteiligung in den Gruppen. Insgesamt wurde in 14 Nennungen eine bessere Beteiligung in den Gruppenarbeiten vermutet. Zurückzuführen sei sie auf eine flüssigere Zusammenarbeit und auf eine leichtere Absprache untereinander. Daneben sei in Präsenz auch die „Kooperationsbereitschaft höher“. So wurde vermutet, dass in Präsenz weniger Trittbrettfahrer-Effekte auftreten. Auch die Kontrolle der Mitarbeit wurde in zwei Nennungen in der Präsenzlehre als einfacher bewertet, wohingegen ein Studierender die Kontrolle in Distanzlehre als einfacher vermutete. Dem gegenüber stehen acht Nennungen für eine bessere Beteiligung im virtuellen Raum, wobei die interaktive Gruppenarbeit und eine höhere Anwesenheit von Studierenden als Gründe genannt werden. Mit jeweils sieben

Personen sprachen sich gleich viele Befragte für eine bessere Zusammenarbeit im virtuellen und im physischen Raum aus, wobei die Gründe dafür variierten. Die Zusammenarbeit sei insbesondere bei der Bearbeitung anspruchsvoller Aufgaben wie das Erstellen aufwändiger Grafiken in Präsenz einfacher. Im virtuellen Raum werde die Zusammenarbeit hingegen durch technische Tools unterstützt, sodass beispielsweise alle Studierende an einem geteilten Dokument arbeiten können. Während drei Befragte die virtuelle Kommunikation als Vorteil

Tab. 4: Vergleich: Präsenzphase des FC in Präsenz- und Distanzlehre

	+ Distanzlehre	Σ	M	SD	+ Präsenzlehre	Σ	M	SD
Zeit	Zeitaufwand	7	3,6	1,3	Zeitaufwand	4	3,5	0,5
	Zeitmanagement	3	4	0,8				
Gruppen	Beteiligung in Gruppen	8	3,9	0,9	Beteiligung in Gruppen	14	4,2	0,7
	Kontrolle	1	4	-	Kontrolle	2	2	0
	Zusammenarbeit	7	3,1	1,1	Zusammenarbeit	7	4,1	0,6
	Anonymität	1	2	-				
Sozial	Kommunikation	3	4,3	0,5	Kommunikation	10	4,7	1,4
					Face to Face Kontakt	15	3	1,4
Zugriff	Nachschlagen von Vorlesungsinhalten	12	3,8	0,1	Skript	1	4	-
Raum	Einfache Gruppeneinteilung	17	3,1	1,2	Echter Raum	1	3	-
	Eigener Schreibtisch	2	3	1				
	Mehr Ruhe	2	3,5	1,5				
	Motivation	2	4,5	0,5	Motivation	3	4,7	0,8

nannten („Kommunikation über BBB sehr gut umsetzbar“), sprechen sich zehn Befragte für eine bessere Kommunikation in Präsenz („bessere Diskussionen“) aus. Sowohl in Präsenz als auch virtuell wurde dabei der Vorteil der Kommunikation als überdurchschnittlich wichtig bewertet. Insbesondere der Kontakt von Angesicht zu Angesicht zeigte mit 15 Nennungen den größten Mehrwert der Präsenzlehre. Darüber hinaus galt mit zwölf Nennungen der einfachere Zugriff auf das LMS während der Bearbeitung als Vorteil der virtuellen Gruppenarbeit, wohingegen von einer Person vermutet wurde, dass in Präsenz ein Skript bereitgestellt wird. Am häufigsten wurde darüber hinaus der verminderte Organisationsaufwand bei der Gruppenbildung als Vorteil der virtuellen Gruppenarbeit genannt. Daneben sei das Arbeiten am eigenen Schreibtisch und die Ruhe vorteilhaft in der virtuellen Lehre. Demgegenüber steht der Wunsch nach einem „echten Raum“. Zwei Befragte sehen eine höhere Motivation beim virtuellen Arbeiten als Vorteil, da es ihnen in Distanzlehre mit aktuellem Veranstaltungsaufbau leichter fallen würde, „dran zu bleiben“. Für drei Befragte würden dagegen u.a. die sozialen Kontakte in der Präsenzlehre zu einer höheren Motivation

führen. Darüber hinaus gab es vier Nennungen, die unabhängig von der Art der örtlichen Synchronität als relevant für das Gelingen der Präsenzphase angesehen wurden. Drei davon richteten sich an das Engagement der Kommilitonen ($M=3,3$; $SD=0,9$) und eine weitere an die Möglichkeit des persönlichen Kontakts ($M=4$).

5 Fazit

Die Covid-19-Pandemie hat die sofortige Umstellung der Präsenzformate in digitale Lehrformate gefordert, um den Lehrbetrieb an Hochschulen weiter aufrecht erhalten zu können. Somit haben Formate wie der FC an Bedeutung gewonnen. Das konventionelle FC-Konzept verbindet die Onlinephase klassischerweise mit einer Präsenzphase in örtlicher Synchronität. Diese Art der Umsetzung war während der Pandemie jedoch nicht möglich, sodass der virtuelle FC in den Vordergrund gerückt ist. In diesem Beitrag wurde aufgezeigt, wie eine Veranstaltung im Rahmen des FC-Konzepts bei vollständiger Distanzlehre aussehen kann. Die Vor- und Nachteile einer solchen Umsetzung wurden ferner untersucht. Der soziale Kontakt in der Online Gruppenarbeit wurde am positivsten bewertet. Mit dem vorgestellten FC-Konzept ist es gelungen, trotz der physischen Distanz eine Nähe zu und zwischen den Studierenden aufzubauen. Dies wurde durch die zufällige Gruppeneinteilung und die Unterstützung seitens der Lehrenden bei der Bearbeitung erreicht. Demgegenüber stand der Zeitdruck bei der Bearbeitung der Online Gruppenarbeit und der generelle Aufwand der Veranstaltung. Zudem konnte ein Trittbrettfahrer-Effekt festgestellt werden, indem sich vereinzelt Gruppenmitglieder nicht an der Lösung der Aufgaben beteiligt haben. Aus Sicht der Studierenden sind solche Trittbrettfahrer-Effekte eine Besonderheit der Umsetzung des FC-Konzepts bei vollständiger Distanzlehre. Im konventionellen FC wäre dies aufgrund der gemeinsamen Bearbeitung der Aufgaben in physischer Präsenz nicht möglich. Somit muss der FC in der Distanzlehre zukünftig an der Überwachung von Fehlverhalten gearbeitet werden, um solche Fälle zu vermeiden. Bei der Frage, ob sich die Studierenden in Zukunft die Veranstaltung in Präsenz- oder Distanzlehre wünschen würden, gab die Mehrheit der Befragten an, dass sie eine Mischform bevorzugen würden. Das konventionelle FC-Konzept sieht diese Mischung vor und vereinbart somit die positiven Aspekte aus Präsenz- und Distanzlehre. Eine virtuelle Präsenz kann zwar im Rahmen der FC-Konzepts umgesetzt werden, birgt aber Herausforderungen mit sich, die sich mit einer zusätzlichen örtlichen Präsenz lösen lassen. Dennoch hat unsere Untersuchung gezeigt, dass der FC auch in Krisensituationen, wie es die Covid-19-Pandemie war, ein Konzept darstellt, das eine Umsetzung der Distanzlehre ermöglicht.

Literatur

- [AAA20] Aji, W. K.; Ardin, H.; Arifin, M. A.: Blended Learning During Pandemic Corona Virus: Teachers' and Students' Perceptions. IDEAS: Journal on English Language Teaching and Learning, Linguistics and Literature 8/2, S. 632–646, 2020.

- [Bo17] Boevé, A. J.; Meijer, R. R.; Bosker, R. J.; Vugteveen, J.; Hoekstra, R.; Albers, C. J.: Implementing the flipped classroom: an exploration of study behaviour and student performance. *Higher Education* 74/6, S. 1015–1032, 2017.
- [BS12] Bergmann, J.; Sams, A.: Flip your classroom: Reach every student in every class every day. International society for technology in education, 2012.
- [BV13] Bishop, J.; Verleger, M. A.: The flipped classroom: A survey of the research. In: 2013 ASEE Annual Conference & Exposition. S. 23–1200, 2013.
- [Co21] Collado-Valero, J.; Rodríguez-Infante, G.; Romero-González, M.; Gamboa-Ternero, S.; Navarro-Soria, I.; Lavigne-Cerván, R.: Flipped Classroom: Active Methodology for Sustainable Learning in Higher Education during Social Distancing Due to COVID-19. *Sustainability* 13/10, S. 5336, 2021.
- [GKC14] Giannakos, M. N.; Krogstie, J.; Chrisochoides, N.: Reviewing the flipped classroom research: reflections for computer science education. In: Proceedings of the computer science education research conference. S. 23–29, 2014.
- [He20] Hew, K. F.; Jia, C.; Gonda, D. E.; Bai, S.: Transitioning to the “new normal” of learning in unpredictable times: pedagogical practices and learning performance in fully online flipped classrooms. *International Journal of Educational Technology in Higher Education* 17/1, S. 1–22, 2020.
- [HS17] Handke, J.; Sperl, A.: Das Inverted Classroom Model: Begleitband zur ersten deutschen ICM-Konferenz. Walter de Gruyter GmbH & Co KG, 2017.
- [Ka20] Kantereit, T.: Hybrid-Unterricht 101: Ein Leitfaden zum Blended Learning für angehende Lehrer: innen. Visual Ink Publishing, 2020.
- [KO19] Kauffeld, S.; Othmer, J.: Handbuch Innovative Lehre. Springer, 2019.
- [MF14] Mayring, P.; Fenzl, T.: Qualitative inhaltsanalyse. In: Handbuch Methoden der empirischen Sozialforschung. Springer, S. 543–556, 2014.
- [MM08] Myrach, T.; Montandon, C.: Blended Learning Kombinationen von Präsenzlehre und E-Learning. *Moderne Personalentwicklung: Mitarbeiterpotenziale erkennen, entwickeln und fördern*/, S. 191, 2008.
- [SDA20] Stöhr, C.; Demazière, C.; Adawi, T.: The polarizing effect of the online flipped classroom. *Computers & Education* 147/, S. 103789, 2020.
- [SPT20] Siripongdee, K.; Pimdee, P.; Tuntiwongwanich, S.: A blended learning model with IoT-based technology: effectively used when the COVID-19 pandemic? *Journal for the Education of Gifted Young Scientists* 8/2, S. 905–917, 2020.
- [Yu20] Yulia, H.: Online learning to prevent the spread of pandemic corona virus in Indonesia. *English Teaching Journal (ETERNAL)* 11/1, 2020.

Auf dem Weg zum neuen Normal — Herausforderungen für die Lehre im post-pandemischen Zeitalter

Veronika Thurner,¹ Axel Böttcher²

Abstract: Die Covid-19-Pandemie hat der hochschulischen Lehre in Deutschland einen massiven Digitalisierungsschub gebracht. Dafür wurden neue, virtuelle Lehr-Lernformate entwickelt und mit Hilfe einer Vielzahl von technischen Tools umgesetzt und durchgeführt. Insgesamt hat sich dadurch der verfügbare Methodenbaukasten deutlich vergrößert und ein Raum an neuen Möglichkeiten eröffnet. Gleichzeitig wurden jedoch auch die Grenzen bzw. Herausforderungen von Online-Lehre offensichtlich. Diese gilt es nun gezielt zu thematisieren, damit sie bei der Systematisierung der während der Pandemie geschaffenen Möglichkeiten zu einem neuen Status quo des virtuell unterstützten Lehrens und Lernens angemessen berücksichtigt und adressiert werden können.

Keywords: Lehrentwicklung; Lehr-Lerntechnologien; Didaktische Architektur; Lehr-Lernprozess; Hochschul-Governance

1 Motivation

Die Covid-19 Pandemie war in hohem Maße disruptiv für die etablierten Prozesse und Strukturen des Lehrens und Lernens an Hochschulen: Traditionell überwiegend auf Präsenz ausgerichtete, ggf. mit virtuellen Elementen angereicherte Lehr-Lernangebote mussten innerhalb kürzester Zeit auf rein virtuelle Formate umgestellt werden, um einerseits die Infektionsrisiken aller unmittelbar am Lehr-Lernprozess Beteiligten zu reduzieren und andererseits gleichzeitig den Lernenden soweit wie möglich eine Weiterentwicklung ihrer Kompetenzen zu ermöglichen, so gut es angesichts der bestehenden Situation eben machbar war.

Vielerorts wurden unter enormem persönlichen Einsatz quasi über Nacht virtuelle Angebote geschaffen, die angesichts der Größe der Herausforderung erstaunlich gut funktioniert haben. Ohne diese Leistungen herabwürdigen zu wollen bleibt dennoch festzustellen, dass viele der entwickelten Ergebnisse mit sehr heißer Nadel gestrickt waren und daher entsprechend als „Emergency Remote Teaching“ zu klassifizieren sind. Auf dem Weg ins neue Normal des Lehrens und Lernens im post-pandemischen Zeitalter lohnt es sich daher, systematisch auf diese Entwicklungen zu blicken und mögliche Verbesserungspotenziale zu heben, bevor sich der Status quo des Lehrens und Lernens mit hohen digitalen Anteilen als neuer Standard etabliert.

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

² Hochschule München, Fakultät für Informatik und Mathematik, D-80335 München, axel.boettcher@hm.edu

Die erarbeiteten neuen Möglichkeiten waren die Grundvoraussetzung dafür, Lehren und Lernen während der Pandemie überhaupt aufrecht erhalten zu können. Gleichzeitig wirft die schiere Vielzahl an neuen Möglichkeiten eine Reihe von Herausforderungen auf, für alle Beteiligten des hochschulischen Lehrens und Lernens. Diese müssen explizit thematisiert werden, damit sie auf dem Weg ins „neue Normal“ entsprechend berücksichtigt werden können.

- Die *Studierenden* müssen sich in der Vielfalt der neuen Angebote zurecht finden, einschließlich der Einarbeitung in die für deren Nutzung erforderlichen Werkzeuge. Des Weiteren benötigen Sie eine entsprechende technische Infrastruktur, um die Technologie-basierten Angebote überhaupt nutzen zu können. Darüber hinaus erfordert ein Lernprozess mit hohen digitalen Anteilen und vergleichsweise wenigen Präsenzanteilen andere Schlüsselkompetenzen im Sinne einer „digitalen Studierfähigkeit“, insbesondere ein höheres Maß an Selbstorganisation und Selbstmotivation.
- Für uns als *Lehrende* hat sich durch die zunehmende Digitalisierung des Lehrens und Lernens unsere alltägliche Arbeit um eine Dimension erweitert. Diese eröffnet uns neue Möglichkeiten – stellt uns jedoch gleichzeitig vor die Herausforderung, aus der Vielfalt an Möglichkeiten eine Selektion zu wählen, die für das, was wir in der Lehre gerade erreichen wollen, zielführend ist; und diese Selektion dann auch in der verfügbaren Zeit und mit den verfügbaren Ressourcen praktisch umzusetzen.
- *Hochschulleitungen und Entscheider:innen* müssen innerhalb der verfügbaren Ressourcen und der bestehenden Rechtslage ihre Gestaltungsspielräume nutzen zur Schaffung von Rahmenbedingungen, die Raum geben um die neuen – und alten – Optionen bestmöglich zu nutzen. Auf der Meta-Ebene erfordert dies auch eine Auseinandersetzung damit, was „bestmöglich“ hier überhaupt heißt, welche Gütekriterien also anzulegen sind und wie deren Erreichen eingeschätzt werden kann. Angesichts der Komplexität und Vielfältigkeit von Lehren und Lernen ist das eine echte Herausforderung. „Billige“, also leicht zu erfassende Kennzahlen wie die Anzahl der Spielminuten von auf die Streaming-Server der Hochschule hochgeladenem Video-Material werden dieser Aufgabe in keiner Weise gerecht – erst recht nicht, wenn diese nicht zwischen gezielt produzierten Erklärvideos und reinen Mitschnitten von synchron gehaltenen Veranstaltungen differenzieren.
- Die *politische Ebene* schließlich ist gefordert, um Mittel und Stellen bereitzustellen, die für die adäquate Umsetzung von (teil-)digitalisierter Lehre erforderlich sind. Des Weiteren muss sie für ihr Controlling der sinnhaften Ressourcenverwendung angemessene Metriken entwickeln. Das reine Zählen von Hochschülerstsemestern einerseits und von Studienabbrüchen andererseits war dafür schon in der Vergangenheit nur bedingt geeignet. Anzustrebendes Ziel ist letztlich so etwas wie die Maximierung des Bildungszuwachses der Lernenden, was aber über Kennzahlen schwer zu erfassen ist, siehe oben.

Um in dieser komplexen Gemengelage einen kleinen Schritt weiterzukommen und die Vielfalt und Komplexität von Möglichkeiten und Unterstützungsbedarfen sichtbar zu machen haben wir Modellierungstechniken aus dem Software Engineering verwendet, um die vielfältigen Konzepte und Buzz-Words zu strukturieren in eine Art Landkarte im Sinne einer didaktischen Architektur. Diese ergänzen wir mit einer Prozesssicht, welche die operative Umsetzung von Lehr-Lerneinheiten adressiert. Auf dieser Grundlage konkretisieren und diskutieren wir Erkenntnisse zu den oben genannten Bedarfen und Herausforderungen, die sich daraus für die verschiedenen Stakeholder-Gruppen ergeben, mit besonderem Fokus auf die Studierenden und die Lehrenden.

2 Überblick über die Gesamtaufgabe des Lehrens

Gute Lehre ist eine hochkomplexe Tätigkeit, in der eine Vielfalt von Konzepten dynamisch zusammenspielt. Entsprechend differenzieren wir eine statische Sicht für die strukturellen Aspekte und eine dynamische Sicht für den Prozess der Erstellung und Verwendung.

2.1 Strukturelle Sicht

Um Orientierung zu schaffen strukturiert Abbildung 1 diese statischen Konzepte in eine didaktische Architektur (siehe [TB21] für eine detaillierte Beschreibung) und dokumentiert diese auf feingranularer Ebene mittels der Notation des Klassendiagramms der UML. Im Kern steht dabei das Konzept des Constructive Alignment, als Grundlage einer modernen konstruktivistischen Didaktik [Bi96]. Es schlägt die Brücke zwischen Lehr- und Lernzielen (dargestellt in grün), Lehr-Lernaktivitäten und den dahinter liegenden didaktischen Methoden (in rot) und den Prüfungen bzw. Lernstandserhebungen (in blau), die das Erreichen der angestrebten Lernziele validieren, summativ oder formativ. Jegliche Lernstandserhebung erfordert eine studentische Aktivität, zu erbringen im Umfeld einer Aufgabe (in lila). Für die Umsetzung sowohl der Lehr-Lerneinheiten als auch der Prüfungselemente braucht es mediale Artefakte (in orange) sowie entsprechende Werkzeuge (in türkis), die abhängig sind vom Durchführungsmodus. Ergänzend dargestellt sind die organisatorische Spezifikation von Studiengängen und Modulen (in schwarz), Aspekte der Evaluation (in pink) sowie die Lehrperson mit ihrer Lehrhaltung (in gelb).

Im Folgenden erläutern wir knapp die Kernelemente dieser einzelnen Bereiche. Eine ausführlichere Beschreibung einer Vorversion dieser Architektur findet sich in [TB21]. Ein Prozessmodell Anwendung des Modells sowie konkrete Beispiele für dessen Umsetzung im Lehr-Lernalltag werden in [TB22] diskutiert.

Ein Studiengang wird realisiert über einen oder mehrere Studienpläne, die sich wiederum aus Modulen zusammensetzen (schwarzer Bereich). Jedes Modul adressiert Lernziele (grün), wird realisiert über Lehr-Lerneinheiten (rot), beinhaltet Leistungserhebungen (blau)

Abb. 1: Klassendiagramm der didaktischen Architektur (als Erweiterung von [TB21])

und wird durchgeführt von einer Lehrperson (gelb). Des Weiteren wird die Qualität der Durchführung eines Moduls über eine Evaluation erhoben (pink).

Jedes Lernziel (grün) adressiert einen Inhalt auf einer bestimmten Kompetenzebene, deren Bedeutung von einer entsprechenden Lernziel-Taxonomie definiert wird, wie z.B. die jeweils sechsstufigen Taxonomien von Bloom [Bl56], deren Überarbeitung von Anderson et al. [An01] oder die jeweils dreistufigen Taxonomien von Metzger und Nüesch [MN04] oder Schneider und Hauer [Sc02].

Der rote Bereich kapselt Lehr-Lernaktivitäten und die dahinter liegenden didaktischen Methoden. Ausgangspunkt ist die Lehr-Lerneinheit, die sich in der Regel um ein zentrales inhaltliches Thema und die damit verbundenen Kompetenzen rankt. Die Lehr-Lerneinheit untergliedert sich in Learning Nuggets [Ro11], also nicht sinnvoll weiter unterteilbare und damit atomare Einheiten. Jedes Nugget zielt auf genau eine didaktische Funktion ab, wie z. B. "Faktenwissen darstellen" oder "Studierende aktivieren". Die Organisation didaktischer Funktionen in eine sinnvolle Reihenfolge wird unterstützt durch didaktische Entwurfsmuster (Patterns) wie z. B. Just-in-time Teaching [No99], der daraus abgeleitete Inverted Classroom [Ha17], Problem Based Learning oder AVIVA. Bei der Umsetzung einer didaktischen Funktion kommen didaktische Methoden zum Einsatz, wie z. B. ein Monolog der Lehrperson oder eine Murmelgruppe.

Ein Spezialfall von Methoden stellen ausführungsorientierte Methoden dar, in deren Rahmen Studierende selbst aktiv werden. Typische Beispiele sind Quiz, Laboraufgabe oder Quelltext-Diktat. Jede ausführungsorientierte Methode umfasst mindestens eine Aufgabenstellung, die von den Studierenden zu bearbeiten ist (lila Bereich). Die ausführungsorientierten Methoden können summativ, formativ oder auch schlichtweg zum Üben zum eingesetzt werden.

Ausführungsorientierte Methoden sind zudem eng verknüpft mit Prüfungen bzw. Lernstandserhebungen (blauer Bereich). Sie sind zentrale Elemente der Nuggets zur Lernstandserhebung, aus denen sich wiederum Prüfungen und andere Instrumente zur Lernstandserhebung zusammensetzen.

Spätestens seit Corona wissen wir, dass sowohl Lehren und Lernen als auch eine Lernstandserhebung in sehr unterschiedlichen Formen durchgeführt werden kann (türkiser Bereich). Insbesondere differenzieren wir zeitliche und örtliche Aspekte, unterscheiden also einerseits zwischen synchronen und asynchronen Formaten sowie andererseits zwischen rein virtuellen, hybriden oder rein in Präsenz durchgeführten Formaten. Je nach Durchführungsform kommen unterschiedliche Werkzeuge zum Einsatz.

Lehren und Lernen involviert fast immer mediale Artefakte (oranger Bereich). Diese können unterschiedlich elaboriert sein, von wenigen Stichworten auf einem Post-it oder Bieruntersetzer bis hin zu ausgeklügelten Erklärvideos. Auch die Lernstandserhebung sowie die Evaluation von Lehr-Lernveranstaltungen erfordern in der Regel entsprechende mediale

Artefakte. Die Erstellung der Medien, und teilweise auch deren späterer Einsatz, erfordert wiederum entsprechende Werkzeuge.

Schließlich spielen noch die Lehrenden (gelber Bereich) eine zentrale Rolle im Lehr-Lernprozess. Sie führen die Module nebst den zugehörigen Lehr-Lernveranstaltungen hauptverantwortlich durch, mit allem was dazu gehört. Wie sie dabei agieren und welche Entscheidungen sie dabei fällen, beispielsweise bei der Auswahl geeigneter Lehr-Lernmethoden, ist maßgeblich geprägt durch die Haltung bzw. das von diesen Personen präferierte Lehr-Lernparadigma.

2.2 Prozesssicht

Ergänzend zu dieser statischen Sicht der Architektur skizziert Abbildung 2 eine Prozessstruktur als Beschreibung der dynamischen Aspekte, die bei der Anwendung der didaktischen Architektur für deren Operationalisierung in einem spezifischen Lehr-Lern-Kontext durchzuführen sind. Dieses Prozessmodell greift den PDCA-Zyklus von Deming [MN06] auf, mit seinen Phasen „plan, do, check“ und „act“ und verbindet diesen mit der Prozessvorstellung der klassischen Medienproduktion, mit ihren Phasen „Präproduktion“, „Produktion“, „Postproduktion“ und „Verteilung“ [Ko18].

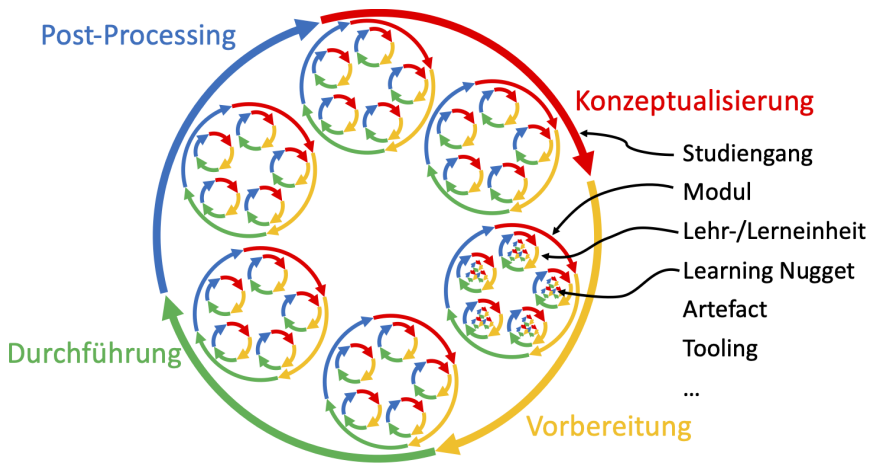


Abb. 2: Auf mehreren Granularitätsebenen verzahnte Prozesse zur Umsetzung der Lehre nach dem Architekturmodell [TB22].

Die Phase der *Konzeptualisierung* fasst alle Aktivitäten zusammen, die der Strukturierung und der Planung von Lehr-Lernaktivitäten dienen. Dabei sind insbesondere die Ausgangsbasis sowie die Ziele zu klären. Die darauf folgende Phase dient der *Vorbereitung* und Entwicklung von benötigtem Material, der Installation der benötigten Werkzeuge und umfasst des Weiteren ggf. die Einrichtung von Organisationsstrukturen und Prozessen,

die für die Durchführung eines Lehr-Lernangebotes erforderlich sind. Die dritte Phase *Durchführung* kennzeichnet das eigentliche Lehren und Lernen in einem bestimmten Durchführungsmodus, also synchron oder asynchron, sowie in Präsenz, blended, hybrid oder rein virtuell. Bei dieser Durchführung kommen die zuvor vorbereiteten Elemente bzw. Materialien zum Einsatz. Die letzte Phase dient der *Nachbereitung* des Lehr-Lernangebotes und fokussiert die Auswertung von Feedback jeglicher Art, beispielsweise durch Prüfungen, formale Evaluation, die dialogische Auseinandersetzung zwischen Lehrenden und Lernenden und Vieles mehr. Die gewonnenen Erkenntnisse fließen ein in den nächsten Prozesszyklus zur Weiterentwicklung des Lehr-Lernangebotes.

Je nach Kontext und konkreter Zielsetzung adressiert der Prozess unterschiedliche Elemente des Architekturmodells und damit sehr unterschiedliche Granularitätsebenen, vom Studiengang bis hinunter zu einzelnen Learning Nuggets. Entsprechend wird der Prozess mehrfach, rekursiv und auf verschiedenen Granularitätsebenen instanziiert, für unterschiedliche didaktische Elemente oder Artefakte.

3 Erfahrungen aus der Pandemie

Diverse Studien haben die Erfahrungen aus der Pandemie zusammengetragen und schaffen so eine systematische Basis für den Weg ins neue Normal und damit die Lehre im post-pandemischen Zeitalter.

Bosse und Würmseer unterscheiden die Mikroebene einzelner Lehrveranstaltungen, die Mesoebene zur Entwicklung von Studienprogrammen einerseits und Lehrkompetenzen andererseits und die Makroebene zur Struktur- und Prozessentwicklung [BW21]. Auf der Mikroebene ist während der Pandemie eine Fülle guter Lösungen entstanden, die meist ad-hoc entwickelt wurden für einen konkreten Kontext. Die dabei gewonnenen Erfahrungen müssen nun generalisiert und in die Breite getragen werden, so dass sie auch für diejenigen Lehrenden nutzbar werden, die von den Neuerungen während der Pandemie eher überfordert waren.

Das Architekturmodell hilft dabei, diese Einzellösungen in einen größeren Kontext des Lehrens und Lernens einzuordnen und so effektiv auf einer breiteren Basis nutzbar zu machen. Gleichzeitig eröffnet es den Lehrenden die Möglichkeit, gezielt nach bestimmten Elementen zu suchen, die in einem konkreten Lehr-Lernsetting grade benötigt werden – beispielsweise ein Learning Nugget für die didaktische Funktion der physischen Aktivierung von Studierenden in einem virtuellen, synchronen Durchführungsmodus.

Des Weiteren trägt das Modell dazu bei, Orientierung zu schaffen innerhalb der Vielfalt von Aspekten, die bei der Erbringung von Lehren und Lernen zu berücksichtigen sind. Im virtuellen Raum (und dort insbesondere bei asynchronen Formaten) fallen Wahrnehmungskanäle zwischen Lehrenden und Lernenden weg. Während bei synchronen Lehr-Lerneinheiten im Hörsaal eine Stimmung im Raum wahrnehmbar ist, die signalisiert, ob beispielsweise

aufseiten der Lernenden Schwierigkeiten besetzen, muss eine äquivalente Erkenntnis im virtuellen Raum durch explizite Maßnahmen erhoben werden. Diese sind gezielt und klar einzuplanen. Viele Dinge, die bei synchroner Präsenzlehre also eher implizit ablaufen brauchen bei virtuellen Formaten ein bewusstes, explizites Hinschauen, damit sie überhaupt umsetzbar werden. (Und natürlich wird auch die synchrone Präsenzlehre davon profitieren, wenn eher intuitiv getroffene Verhaltensentscheidungen der Lehrperson hinterfragt und ggf. systematisiert werden.) Das Architekturmodell trägt dazu bei, hier einen Überblick zu schaffen, beispielsweise über die Vielfalt und das Zusammenspiel von didaktischen Funktionen, etablierten Entwurfsmustern zur Gestaltung von Lehren und Lernen oder für didaktische Methoden.

Das Prozessmodell hilft, deutlich zu machen dass sich in unterschiedlichen Lehr-Lernszenarien die zeitlichen Hauptaufwände für die Leistungserbringung zwischen den Phasen verschieben. Beispielsweise liegt bei auf selbst gedrehten Erklärvideos basierendem Just-in-Time-Teaching ein erheblicher Teil des Aufwandes für die Lehrperson in der Vorbereitungsphase (Produktion von Videos und zugehörigen Aufgaben). Die eigentliche Durchführung des Lehrens und Lernens beinhaltet dagegen einen hohen Anteil an studentischer Selbstlernzeit (zur Rezeption des JiTT-Materials und Bearbeitung der Aufgaben). Aus Perspektive der Lehrperson betrachtet nehmen die synchronen Lehr-Lerneinheiten dabei in der Regel insgesamt einen geringeren Anteil am Gesamtaufwand ein als bei traditioneller Hörsaal-Lehre. Bestehende Abrechnungsmodelle für Lehrverpflichtungen spiegeln diese Verschiebung derzeit nur bedingt wider.

Die politische Ebene ist daher gefordert, Regelungen zu Schaffen für die Abrechnung der Lehrverpflichtung, welche die digitale Lehre an sich sowie idealer Weise auch den Prozess der Digitalisierung selbst angemessen berücksichtigt. Hier weisen die Regelungen der einzelnen Bundesländer große Unterschiede auf. Während die Lehrverpflichtungsverordnung LVVO des Landes Niedersachsen in der Version von 2018 beispielsweise in §14 Abs. 5 festlegt „Die Erstellung und Betreuung von Multimediaangeboten kann in einem dem Zeitaufwand entsprechenden Umfang bei der Erfüllung der Lehrverpflichtung berücksichtigt werden.“, so ergänzt das Pendant LUFV des Landes Bayern in der Version von 2019 in §3 Abs. 9 einen Anrechnungsfaktor und führt gleichzeitig eine Deckelung auf 25% ein: „Die Erstellung und Betreuung von Multimedia-Angeboten kann in einem dem Zeitaufwand entsprechenden Umfang auf die Lehrverpflichtung angerechnet werden, jedoch höchstens bis 25 v.H. der festgelegten Lehrverpflichtung. Eine Lehrveranstaltungsstunde (Anrechnungsfaktor 1) entspricht drei Arbeitsstunden.“ Gleichzeitig kennt die LUFV bereits die „moderne, insbesondere internetbasierte Ausgestaltung“ von Lehre (siehe LUFV §14 Abs. 2 (1)).

4 Zusammenfassung, Fazit und Ausblick

Die systematische Übertragung und Ausweitung der während der Covid-19 Pandemie gewonnenen Erfahrungen und entwickelten Lösungen im Bereich der digital unterstützten Lehre auf die Zeit nach der Pandemie bringt eine Fülle von Herausforderungen mit sich, die

auf dem Weg ins neue Normal angegangen werden muss. Dazu braucht es sowohl einen Überblick über die verschiedenen Elemente und Konzepte, die hierbei zusammenspielen, sowie ein Verständnis für darauf basierenden Prozesse beim praktischen Einsatz.

Die in dieser Arbeit vorgestellten Modelle für die statische Sicht (Architektur) und die dynamische Sicht (Prozess) tragen dazu bei, die Komplexität von Lehren und Lernen sichtbar zu machen und Handlungs- bzw. Investitionsbedarfe zu identifizieren.

Literaturverzeichnis

- [An01] Anderson, Lorin W.; Krathwohl, David R.; Airasian, Peter W.; Cruikshank, Kathleen A.; Mayer, Richard E.; Pintrich, Paul R.; Rath, James; Wittrock, Merlin C., Hrsg. A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Longman, New York, 1. Auflage, 2001.
- [Bi96] Biggs, John: Enhancing teaching through constructive alignment. Higher education, 32(3):347–364, 1996.
- [Bl56] Bloom, B.; Engelhart, M.; Furst, E.; Hill, W.; Krathwohl, D.: Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain. David McKay Company, 1956.
- [BW21] Bosse, Elke; Würmseer, Grit: Vom Krisenmodus zur strategischen Lehrentwicklung. Magazin für Hochschulentwicklung – Vom Krisenmodus zur strategischen Lehrentwicklung, (1):6–7, 2021.
- [Ha17] Handke, J.: Handbuch Hochschullehre Digital: Leitfaden für eine moderne und mediengerechte Lehre. Tectum-Sachbuch. Tectum Wissenschaftsverlag, 2017.
- [Ko18] Koester, Jonas: Video in the Age of Digital Learning. Springer International Publishing, 2018.
- [MN04] Metzger, C.; Nüesch, C.: Fair prüfen: Ein Qualitätsleitfaden für Prüfende an Hochschulen, 2004.
- [MN06] Moen, Ronald; Norman, Clifford: Evolution of the PDCA cycle. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.5465&rep=rep1&type=pdf>, 2006.
- [No99] Novak, G.M.; Gavrín, A.; Patterson, E.; Christian, W.: Just-in-time teaching: blending active learning with web technology. Prentice Hall series in educational innovation. Prentice Hall, 1999.
- [Ro11] Robes, Jochen: Learning Nuggets – Wunsch und Wirklichkeit. Personalführung, (2):50–53, 2011.
- [Sc02] Schneider, W.: Foliensatz "Einführung in die Wirtschaftspädagogik", 2002.
- [TB21] Thurner, Veronika; Böttcher, Axel: Designing an Architecture for Structuring Didactic Concepts, Methods and Tools. In: ICL2021 – 24th International Conference on Interactive Collaborative Learning. S. 864–875, 9 2021.
- [TB22] Thurner, Veronika; Böttcher, Axel: An Architectural Concept for Didactics that integrates Technology into Teaching, Learning and Assessment. In: Learning with Technologies and Technologies in Learning. Springer, 2022.

Ergänzende Themen

A study on the quality mindedness of students

Steffen Dick,¹ Stefan Schulz,² Christoph Bockisch³

Abstract: Awareness of software quality is a skill generally agreed to be very important working in the industry, but we have observed that it receives little attention in the first-year programming education at universities. Besides preparing students for work life, we assume that good knowledge of software quality also helps computer science students to study more successfully. In this paper, we present a method for determining the quality-awareness, based on a diagnostic assignment and a questionnaire. Using the method we establish a baseline measurement in two courses that students typically follow in their first year, showing that quality awareness correlates with good grades. According to the baseline, the level of quality-mindedness of approximately half the students is not satisfactory.

Keywords: code quality awareness, students, learners, survey, evaluation

1 Introduction

It is obvious that writing quality software is very important in practice, improving among others the maintainability of software. As quality attributes, in our context we especially consider readability, following conventions, documentation and sufficient testing. At the same time, it is a known problem that early programming courses at universities do not pay very much attention to software quality because they focus on basic programming concepts and typically only use small exercises. Due to the small size of code to be written and because this code is usually not further developed after it is submitted, the necessity of writing high quality code is not obvious to students.

Our expectation, however, is that early programming education also sets the direction of how students and graduates treat software quality. Furthermore, we expect that, if writing good quality software becomes their second nature, this will positively influence their further success, e.g., because it will improve their capability of working in teams. As part of the Erasmus+ project *QPED* (<https://qpед.eu>), we research approaches of strengthening software quality in early programming courses and investigate how they impact the quality awareness in students. In this paper, we report on establishing a baseline measurement of the students' quality awareness and resulting indications that our expectations could be true. In the future, we plan to adapt the syllabus of our first year programming education and repeat these measurements to evaluate changes in quality awareness and overall programming abilities of our students.

¹ Philipps Universität Marburg, Germany dickst@informatik.uni-marburg.de

² Philipps Universität Marburg, Germany schulzs@informatik.uni-marburg.de

³ Philipps Universität Marburg, Germany bockisch@acm.org

For that purpose, we devised a study comprised of two parts. Firstly, we developed a diagnostic assignment to assess the students' knowledge of software quality as part of the final exam of an early programming course. Secondly, we developed a survey that lets students in a higher semester rate the importance of different quality criteria. We also used this to gather information on the programming background before their university education.

We elaborate on the results of this survey in the following sections. As one of the main results, we were able to confirm that good knowledge of software quality has a strong positive correlation with success in an early programming course. The questionnaire has revealed that students do already consider software quality rather important—even more so when working in a team—but the quality-awareness still needs to be improved. It has also shown that for most of the students the university courses are the first formal programming education, showing the importance of laying a solid foundation here.

2 Diagnostic Assessment

We wanted to see how knowing about software quality relates to the final grade of early learners. Therefore, we included a task within the final exam about the quality of a given, flawed code snippet. Thereby, the included flaws are no actual bugs, but rather bad style or missing parameter validation. The students were supposed to write a sufficient number of adequate tests and to name and fix a flaw within that code snippet. This was done for the first exam and also for the repeat exam, though we exchanged the code snippet.

For their first exam, the students received a flawed tree-based implementation of the Perrin-sequence which is similar to the Fibonacci sequence that the students were already familiar with. For the repeat exam, we chose a register for citizens which had similar flaws to the implementation of the Perrin-sequence. Over 130 students participated in the first exam and about 60 participated in the repeat exam. Some of these 60 students had already taken the first exam and failed, the others elected to directly participate in the repeat exam. For both exams we used a tool [Dr19] to calculate the item-test-correlation according to Pearson for both sub-tasks. The result is a value between -1 and +1, whereby the +1 is a perfect correlation, 0 is no correlation at all and -1 is a perfect negative correlation. This means students with no quality awareness would have the best grades and vice versa.

For the results of the first exam, we calculated correlations with the final grade of 0.55 for writing the tests, 0.41 for naming and fixing a flaw and 0.61 for the task as a whole. A value above 0.4 is already considered a good correlation [MoosbruggerKelava2012]. With smaller sample size of only about 60 students, the item-test-correlation could still be calculated to 0.34 for writing the tests. The second part, naming and fixing a flaw, was skipped by most participants of the repeat exam leading to an item-test-correlation of 0.01 and bringing the correlation of the task as a whole down to 0.25. The high correlation of the first exam, nevertheless, suggests that further research into a causation between quality mindedness and good grades is promising.

3 Surveying the Students

We concluded our evaluation by handing out a survey to participants of a course that is usually situated between the second and third semester. The goal of this survey was to see how much attention students normally pay to software quality when writing source code on their own or within a group. To achieve this, we needed to include some metrics or concepts of software quality within the survey. Going over every metric in existence would have blown up the survey and might have caused less students willing to participate. Because of this, we selected 9 metrics to be included in our survey.

One of the criteria for selecting the metrics is the knowledge of the students. As different cadences of lectures are possible, we had to choose the least common denominator of lectures that we can expect to be completed by the participants. From this, we can expect that students are familiar with the concept of *Magic Numbers* (constants that are not declared as such) and the “Don’t Repeat Yourself” principle, which is heavily involved with the metric *Duplicated Code*, so we included both in the survey. The second criterion we considered for selecting the metrics was their naming: We wanted to include those with self-explanatory names so that, even if they hadn’t heard of them, the students would be able to have an inkling of what they were about. Metrics such as *Logical Lines of Code*, *Number of Fields*, *Too Large Methods* and *Characters per Line* were included with this reasoning. Our last criterion for selecting a metric was testing. Because testing source code is one of the main pillars of software quality and is taught in both beginner lectures, testing metrics were included. In combination with the second criterion, we chose *Number of Passed Tests*, *Number of Failed Tests* and *Test-Code Coverage*.

Out of 110 students 40 participated in the survey, so the results may be biased, although we do not have any evidence that a specific group of students (e.g., especially good or bad students) stands out in the survey. About 87% of participants identified as male, whereas only 8% identified as female which is below our university’s average of about 15-20% female students in computer science. And 5% skipped the question all together.

Of those who participated, 50% were in their second, and about 29% in their third semester. Furthermore, 9% of participants were in their fourth, 8% in their fifth semester and 2% in their tenth semester. This confirms our assumption that the majority of participants were still at the beginning of their university education.

	Compulsory Classes	Optional Classes	Private Means	Industry
None	66%	47%	10.5%	66%
little	5%	10%	18%	10%
some	18%	26%	34%	3%
much	3%	8%	21%	8%
very much	5%	8%	16%	5%

Tab. 1: Participants rating the source of their knowledge about CS before university

We also asked participants to quantify their prior knowledge of computer science before their first semester from various likely sources, the results are shown in Tab. 1. Participants who skipped one or more of these questions are not represented in the table, which is why some columns do not add up to 100%. From the table, we can see that only about 50% of students received formal education on computer science before starting university and that most prior knowledge comes from private means. Almost no student received prior knowledge through working in the industry. So the first formal knowledge half of the students receive comes from university, so it is the foundation for their future careers.

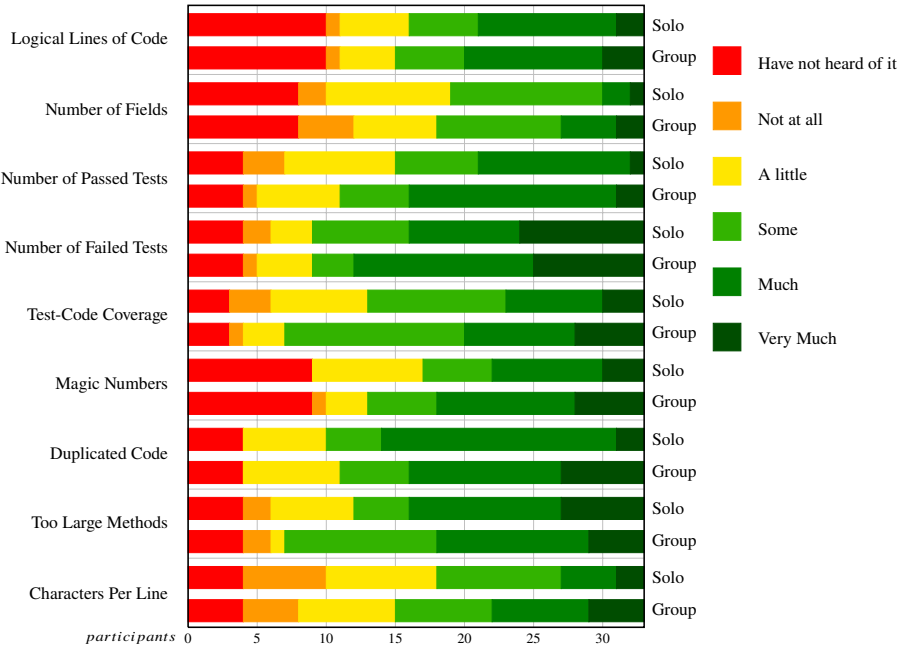


Fig. 1: Participants rating how much attention concepts receive when coding.

Furthermore, we asked the students to rate how much attention the concepts behind the 9 aforementioned metrics receive when they are writing code. They were supposed to rate the concepts when writing code on their own and when within a group of fellow students. The results are shown in Fig. 1. For every concept there are two bars in the chart, with the bottom one being the results when in a group and the top one when writing code on their own. If a student stated that they had not heard of the concept working alone or in a group, the other answer for that concept was changed to *Have not heard of it*. Three answers were demoted this way.

Overall, at least 10% of participants did not know a given metric. Roughly 30% of participants had not even heard about *Magic Numbers*, though it is part of the compulsory curriculum. Only 50% on average stated that they paid *much* attention to the metrics, with *Number of Passed Tests*, *Number of Failed Tests* and *Duplicated Code* receiving the most attention.

4 Related Work

Over the past two decades, a lot of research has been done on the effects of teaching test-driven development concepts on various levels of CS education [DJS08; Ga20; Me20]. Edwards [Ed03] discusses the problem, that many CS students struggle to adopt analysis and comprehension skills, which are crucial for software development. To combat this, they suggest to shift curricula of junior-level courses to a test-first approach and present an automated testing tool, capable of providing helpful feedback to students. In a comparison between courses with and without TDD [Ed04], they observed a reduction of 45% regarding defects in the student's code.

Matthies et al. [MTU17] propose Prof. CI, a novel approach to programming exercises and teaching test-driven development. It leverages modern online courses, designed to teach programming and provides a modern IDE-based tool-chain for the exercises. While the authors did not evaluate the code quality of the student's work, they observed that the students were more likely to write more tests in future projects. An important takeaway is the fact that extra care is required when designing tool-enhanced programming exercises, especially when the exercise requires the students to write tests.

Similar to these works, our goal is to teach better code quality standards to junior-level students by shifting the curriculum of our university's entry level courses. This shift will mainly be achieved by remodelling the mandatory exercises during the semester, using web- and IDE-based tool-support to provide quality-related feedback for the submissions.

5 Conclusion and Future Work

To assess the quality-mindedness of students in their early education, we have performed a study in two courses typically followed in the first and second semester, respectively. This firstly comprised a diagnostic assignment in the final exam which was taken by 130 students on the first exam and 60 students on the repeat exam. An item-test-correlations of 0.6 and 0.25 (in the first and repeat exam, respectively) proves a high correlation between understanding software quality and getting a good grade in the final exam.

Secondly, we made a survey with students in a lab course, most of whom were at the end of their second or third semester. In this survey, 40 students self-appraised how much attention they pay on software quality by rating the importance of nine different quality metrics when developing alone or in a team. In this survey, each quality concept was not known to 10–30% of the students. And on average only half of the students paid much attention to the quality concepts. This shows that it is a good idea to increase the efforts to teach early learners about software quality to improve their quality-mindedness.

In the future, we will update the syllabus of our introductory course on object-oriented programming to strengthen the topic of software quality and to make students more aware

of the importance of paying attention to software quality. We also plan to repeat the study presented in this paper with students who followed the updated course to verify our assumption that by changing the syllabus we can increase the quality-mindedness of students. Furthermore, we will investigate if the changed syllabus will improve the achievements in the exam regarding software quality, and whether an increased quality-mindedness also leads to better overall grades. If this can be shown, it would allow us to conclude that quality-mindedness is not only correlated with good grades but also a reason for them.

Acknowledgements

This work is partly funded by the Erasmus+ project *Quality-focussed Programming Education (QPED)*, 2020-1-NL01-KA203-064626.

References

- [DJS08] Desai, C.; Janzen, D.; Savage, K.: A survey of evidence for test-driven development in academia. *ACM SIGCSE Bulletin* 40/2, 2008.
- [Dr19] Dreyer, T.: Automatische Gütebewertung von Informatikklausuren, Bachelor Thesis, Sept. 2019.
- [Ed03] Edwards, S. H.: Rethinking computer science education from a test-first perspective. In: *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. 2003.
- [Ed04] Edwards, S. H.: Using software testing to move students from trial-and-error to reflection-in-action. In: *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 2004.
- [Ga20] Garousi, V.; Rainer, A.; Lauvås Jr, P.; Arcuri, A.: Software-testing education: A systematic literature mapping. *Journal of Systems and Software* 165/, 2020.
- [Me20] Melo, S. M.; Moreira, V. X.; Paschoal, L. N.; Souza, S. R.: Testing Education: A Survey on a Global Scale. In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*. 2020.
- [MTU17] Matthies, C.; Treffer, A.; Uflacker, M.: Prof. CI: Employing continuous integration services and Github workflows to teach test-driven development. In: *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017.

Emotionale Veränderung von Studierenden bedingt durch die digitale Transformation der Lehr- und Lernformate

Eine empirische Untersuchung unter Covid-19-Bedingungen

Jonas Kötter,¹ Agnes Mainka,² Natallia Kukharenka³

Abstract: Die weltweite Covid-19 Pandemie hatte zur Folge, dass bisherige Präsenzveranstaltungen an Universitäten ad hoc in digitale Lehr- und Lernformate transformiert werden mussten. In der vorliegenden Panelstudie werden die emotionalen Auswirkungen dieser Maßnahme aus Sicht von Studierenden im 1. und 3. Semester an der Universität Osnabrück untersucht. Mithilfe einer etablierten Methode aus der Trauerforschung wurde die emotionale Veränderung der Studierenden während des Semesters betrachtet. Zusätzlich wurden 690 Antworten identifiziert, die zu einer Veränderung der emotionalen Lage der Studierenden beigetragen haben. Diese konnten in einem induktiven Verfahren neun Dimensionen zugeordnet werden. Insgesamt kann festgehalten werden, dass die Proband:innen keine starken emotionalen Veränderungen aufweisen und dass die Lehr- und Lernformate als ein wesentlicher Einflussfaktor durch die Studierenden benannt wurde.

Keywords: Emotionale Veränderung; Universität; digitale Lehre; Lehr- und Lernformate

1 Einleitung

Über 1,7 Milliarden Studierende waren weltweit von Lockdown-Maßnahmen zur Prävention der Bevölkerung während der anhaltenden Corona-Pandemie beginnend im Jahr 2020 betroffen [Du20]. Dies hatte zur Folge, dass sie ihr Studium ad hoc digital bestreiten mussten. In vielen Fällen fand die digitale Transformation der Lehr- und Lernformate häufig als Notlösung statt [HWO21]. Diese unvorhergesehene drastische Veränderung wird für Lehrende und Studierende gleichermaßen als „Schock“ bezeichnet [Ha21]. Um sich aus einem Schockzustand wieder herauszubewegen, werden verschiedene emotionale Phasen durchlaufen. Dieser Bewältigungsprozess ist unter anderem aus der Trauerforschung bekannt [Ke21]. Ursprünglich betrachtet man die Verabreichung eines Verlustes. Im Falle der Studierenden sprechen wir von dem Verlust der physischen Lern- und Lehrformate, die den Studierenden aus ihrer Schulzeit und dem Studium sehr vertraut sind. Studierende sind es

¹ Universität Osnabrück, BWL/Organisation und Wirtschaftsinformatik, Katharinenstraße 1-3, 49074 Osnabrueck, Deutschland, jonas.koetter@uni-osnabrueck.de

² Universität Osnabrück, Institut für Informatik, Wachsbleiche 27, 49090 Osnabrueck, Deutschland, agnes.mainka@uni-osnabrueck.de

³ Universität Osnabrück, Didaktik der Humandienstleistungsberufe, Nelson-Mandela-Str. 13, 49069 Osnabrueck, Deutschland, natallia.kukharenka@uni-osnabrueck.de

gewohnt, wenn sie mit neuen Problemstellungen konfrontiert werden, dass eine Lehrperson sie in einer physischen Lernumgebung beim Lernen unterstützt. Genau diese persönliche Betreuung vermissen Studierende in einem virtuellen Lehr- und Lernszenario [DFR21]. Wenn neue Lernmaterialien zu kompliziert sind oder Problemstellungen nicht selbst bewältigt werden können, führt das bei einigen zu Desorientierung und Isolation. Dies wiederum wirkt sich negativ auf die aktive Teilnahme und Lernbereitschaft aus. Negative und positive Emotionen haben einen Einfluss auf Lernende und sollten bei der Untersuchung digitaler Lehr- und Lernformate im Kontext der Hochschullehre berücksichtigt werden. Zembylas [Ze08] hat sich mit verschiedenen Studien zu Emotionen in der online Lehre beschäftigt und bestätigt, dass es einen Zusammenhang zwischen positiven und negativen Emotionen mit entsprechenden positiven bzw. negativen Auswirkungen auf den Lernprozess gibt. Ebenfalls sind Emotionen ausschlaggebend für den Lernerfolg.

Nur wenige Studien behandeln den emotionalen Verlauf der Studierenden während eines gesamten Semesters [Be04] und keine uns bekannte Studie hat die emotionalen Phasen, die Studierende in der Studieneingangsphase durchlaufen, systematisch untersucht. Dieser Beitrag füllt diese Lücke, indem der emotionale Verlauf im digitalen Wintersemester HEI (Higher Education Institutions) Vollzeit-Studierender an einer deutschen Universität während der anhaltenden Corona-Pandemie empirisch erhoben und untersucht wird. Hinzu kommt in diesem besonderen Semester, dass die Studierenden aus einem „Schock“ heraus die vielerorts ad hoc digital transformierten Lehr- und Lernformate nutzen müssen. Aus dieser Problemstellung ergeben sich folgende Forschungsfragen:

FF1: Wie sieht die emotionale Veränderungskurve im Wintersemester bei Studierenden im ersten und dritten Semester aus?

FF1a: Welchen Phasen können Studienanfänger:innen und Studierende in höheren Semestern zugeordnet werden?

FF1b: Welcher Phasenverlauf kann bei den Studierenden festgestellt werden?

FF2: Können Einflussfaktoren identifiziert werden, welche zu einer Veränderung der emotionalen Lage beigetragen haben?

In diesem Beitrag wird zunächst die Messung von Emotionen im Kontext der digitalen Hochschullehre anhand aktueller Literatur dargelegt und ein mögliches Modell zur Messung der Emotionen näher beschrieben. Im Abschnitt Methodik wird die empirische Erhebung der Daten für diese Studie aufgezeigt. Darauf aufbauend erfolgt im vierten Kapitel sowohl die Auswertung der Daten als auch die Analyse der Ergebnisse. Das fünfte Kapitel beinhaltet die Interpretation der Phasenverläufe und der damit verbundenen identifizierten Einflussfaktoren. Im Fazit folgen eine Zusammenfassung, Lessons Learned und resultierende, offene Forschungsfragen.

2 Emotionen in der digitalen Hochschullehre

Die Corona-Pandemie hat wesentlich dazu beigetragen, dass die emotionale Situation Studierender in der digitalen Lehre in den Fokus von Wissenschaftler:innen gerückt ist. Abstandsregelungen und Lockdown-Maßnahmen haben dazu geführt, dass die digitale Lehre vielerorts die einzige Möglichkeit geworden ist, Lehre weiterhin anzubieten. Somit konnte auch vermehrt Forschung in der digitalen Lehre durchgeführt werden. Zum Einsatz kamen verschiedene Messinstrumente zur Erfassung der Emotionen [PSR20], [Zh20], [Qa20], [CC20], [Li21], [DSB21]. Zusätzlich wurden spezifische Corona-Fragen entwickelt [Li21] oder eine Betrachtung der Prüfungsangst vorgenommen [ACS21].

Anhand verschiedener Studien konnte belegt werden, dass sich positive Emotionen positiv auf die Selbstwirksamkeit als auch auf den Lernerfolg auswirken und zudem die Kreativität und flexible Problemlösungskompetenz fördern [Al20], [EAT15], [RMD14]. Eine Studie, die während der Corona Pandemie Studierende über einen längeren Zeitraum betrachtet hat, konnte folgende Gefühle bei den Proband:innen während der Lockdown-Maßnahmen feststellen: Angst, Trauer, Wut, Langeweile, Schuld, Einsamkeit und Freude [Gi20]. Überwiegend wurden negative Emotionen durch die Studierenden benannt. Ein Aspekt, der viele Studierende erfreut hat, war die gewonnene Zeit mit der Familie. Emotionen in der digitalen Hochschullehre haben bereits eine zunehmende Bedeutung in der wissenschaftlichen Literatur erhalten, aber es gibt kaum Beiträge, die sich über einen längeren Zeitraum mit der emotionalen Veränderung der Studierenden befassen oder gar diese in Zusammenhang zu den angebotenen Lehr- und Lernformaten setzen.

Eine Möglichkeit, Veränderungen der emotionalen Entwicklung von Betroffenen messen zu können hat Elisabeth Kübler-Ross, eine Psychiaterin, im Jahr 1969 in ihrem Beitrag *Ön Death and Dying* "konzipiert [ET02]. Mithilfe von Gesprächen bzw. Interviews mit krebserkrankten Menschen hat sie die Reaktionen der kranken Menschen in Form einer Trauerforschung analysiert. Aus diesen Ergebnissen hat die Autorin fünf verschiedene emotionale Phasen (Verleugnung, Wut, Verhandlung, Depression und Akzeptanz) der Veränderung abgeleitet [Ku73]. In der Literatur können neben dem ursprünglichen Fünf-Phasen-Modell von Kübler-Ross auch Sechs-, [Bl08] Sieben-, [St97] oder Acht-Phasen-[Fr18] Modelle der emotionalen Veränderung identifiziert werden. Nach Frei [Fr18] besteht die Gemeinsamkeit aller Modelle darin, dass diese mit einem einschneidenden Ereignis wie dem Tod oder dem Arbeitsplatzverlust beginnen und entsprechende Phasenverläufe aufweisen. In weiterführenden Forschungen konnte bewiesen werden, dass positive und negative Veränderungen zu ähnlich starken Emotionen führen. Eine Beförderung kann bspw. als Schock vom Beteiligten aufgenommen werden. Je nach Situation und Art der Veränderung können sowohl die Höhen und Tiefen der Kurven als auch die benötigte Zeit für den Wandel individuell ausfallen [Wi08]. Die beschriebenen Phasen dienen als Rahmen zum Umgang mit der emotionalen Veränderung und müssen nicht sequenziell durchlaufen werden, wie im ursprünglichen Modell von Kübler-Ross [Ku73].

Der idealtypische Prozess nach Kübler-Ross beginnt mit der Verleugnungsphase, die in

Folge eines Schocks auftritt. Dieser kann bspw. durch den drohenden Arbeitsplatzverlust oder eine andere organisatorische Veränderung eintreten. In der folgenden Phase der Wut reagiert der Betroffene mit starken emotionalen Reaktionen. Diese Reaktionen können bspw. aus einem Gefühl von Verrat resultieren. Der Betroffene versucht in der dritten Phase (Verhandlung) bspw. den drohenden Arbeitsplatzverlust oder die organisatorische Veränderung abzuwenden bzw. zu verhindern. Die Depression bildet die vorletzte Phase. Innerhalb dieser Phase stellen sich negative Gefühle wie bspw. Traurigkeit ein. Der Betroffene ist nicht mehr wütend und verarbeitet die Veränderung. Ausgehend von der Depression folgt die Akzeptanz der Situation. Der Betroffene sieht ein, dass er seinen Arbeitsplatz verlieren wird und findet sich mit der Situation ab [ET02]. Blau hat 2008 zahlreiche Studien im Bereich der Berufsberatung und der Theorie des Arbeitsplatzverlustes analysiert und kam zu dem Ergebnis, dass eine weitere Phase der Erkundung in den identifizierten Studien als Erweiterung mit aufgenommen wurde. Innerhalb dieser Phase wird durch die Betroffenen nach einer möglichen Umsetzung der Veränderungen gesucht. Für die Ausarbeitung wird als Grundlage die überarbeitete und bereits validierte Variante des Sechs-Phasen-Modells der emotionalen Veränderung nach Blau verwendet [Bl08] und entsprechend mit dem Fokus im Bereich HEI (Higher Education Institutions) adaptiert.

3 Methodik

Die Untersuchung des emotionalen Verhaltens der Bachelor-Studierenden im digitalen Wintersemester 2020/21 wurde unter Corona-Bedingungen an der Universität Osnabrück im Rahmen einer Panelstudie durchgeführt. Das sogenannte Tagebuchstudien-Format wurde gewählt, um das Erleben und Verhalten der Studierenden im digitalen Semester über einen längeren Zeitraum zu beobachten und zu analysieren. Diese Art der Untersuchung liefert realitätsnahe Daten, da die Wahrnehmung der Proband:innen zeitnah unter realen Bedingungen erfasst wird und somit der Effekt der Retrospektive verringert wird [BDR03], [Oh10]. Um eventuelle Unterschiede des emotionalen Erlebens in Bezug auf das Online-Studium zwischen den Studierenden aus der Studieneingangsphase sowie aus den höheren Semestern festzustellen, wurden jeweils 45 Proband:innen aus dem ersten sowie aus dem dritten Semester aus unterschiedlichen Fachrichtungen der Universität befragt. Insgesamt wurden 44 weibliche, 44 männliche und 2 diverse Teilnehmer:innen in die Studie aufgenommen. Die Teilnehmer:innen haben sich freiwillig für diese Studie angemeldet. Als Gegenleistung wurde Geld ausgezahlt oder Versuchspersonenstunden ausgestellt. Die Proband:innen sollten drei Monate lang an einer wöchentlichen Befragung teilnehmen, in der sie drei selbst gewählte, aktuell besuchte Lehrveranstaltungen evaluierten und über Interaktion und Kommunikation mit den Lehrenden, Kommiliton:innen sowie Service- und Beratungsstellen der Universität berichteten. Darüber hinaus mussten die Studierenden einmal pro Monat einen zusätzlichen Fragebogen im Rahmen der Erforschung ihres emotionalen Zustandes ausfüllen. Insgesamt fanden zehn wöchentliche und drei monatliche Erhebungen statt. Die emotionale Veränderung der Studierenden wurde in Bezug auf die während der Corona-Pandemie geänderten Lehr- und Lernformate untersucht. Der Fragenkomplex

beinhaltete sechs Aussagen, die sechs Phasen der emotionalen Entwicklung abbilden, wie *Verleugnung*, *Wut*, *Verhandlung*, *Depression*, *Erkundung* und *Akzeptanz*. Die Aussagen wurden mit Antwortmöglichkeiten in Form von sechsstufigen Ratingskalen von 1 = „Stimme gar nicht zu“ bis zu 6 = „Stimme voll und ganz zu“ versehen. Die Proband:innen sollten folgende Fragen beantworten:

1. Ich glaube nicht, dass aktuell interessante Lehr- und Lernformate bedingt durch die Corona-Pandemie entfallen (Phase: *Verleugnung*).
2. Ich bin wütend, dass interessante Lehr- und Lernformate bedingt durch die Corona-Pandemie entfallen sind (Phase: *Wut*).
3. Es sollte ein Kompromiss gefunden werden, dass interessante Lehr- und Lernformate bedingt durch die Corona-Pandemie beibehalten werden (Phase: *Verhandlung*).
4. Ich finde es sehr schade, dass interessante Lehr- und Lernformate bedingt durch die Corona-Pandemie entfallen sind (Phase: *Depression*).
5. Es können aus der Corona-Pandemie positive Gelegenheiten entstehen, interessante Lehr- und Lernformate kennenzulernen (Phase: *Erkundung*).
6. Ich werde mich auf die durch die Corona-Pandemie entstandenen neuen Lehr- und Lernformate einstellen (Phase: *Akzeptanz*).

Alle Daten wurden über das online Umfragetool LimeSurvey der Universität erhoben. Die Datensatz von 70 Befragten standen am Ende vollständig zur Auswertung bereit. Zur Clusterung von Gruppen wird ein exploratives, hierarchisches Vorgehen (agglomerative Clusteranalyse) angewendet, um Ähnlichkeiten, im Besonderen Distanzen zwischen den Studierenden, zu identifizieren. Als Clustermethode wurde das Average Linkage Verfahren (Verlinkung innerhalb der Gruppen) angewendet, da besonders kleine Gruppen bzw. einzelne Studierende vermutet werden, die sonst im Rahmen anderer Clusteranalysen wie bspw. dem Ward-Verfahren in andere Gruppen mit einsortiert werden würden. Ein weiterer Vorteil ist, dass sich das Verfahren für Korrelationen eignet [Ha09]. Die Daten wurden mithilfe von IBM SPSS Statistic – Version 26 ausgewertet. Darüber hinaus konnten zusätzliche Faktoren festgestellt werden, die das emotionale Befinden der Studierenden beeinflusst haben. Die Faktoren wurden mithilfe von zwei separaten Inhaltsanalysen in Anlehnung an Mayring identifiziert (**FF2**). Die Inhaltsanalysen basieren auf zwei offenen Fragestellungen („Was hat Ihnen diese Woche gut gefallen?“ und „Was hat Ihnen diese Woche nicht gut gefallen?“), welche die Studierenden wöchentlich beantwortet haben. In einem deduktiven Vorgehen wurden die Dimensionen zunächst von den Forschern definiert. Anschließend wurden diese mithilfe eines Kreuzcodierungsverfahren miteinander verglichen und zusammengeführt.

4 Ergebnisse

Aus der Clusteranalyse konnten insgesamt 13 Gruppen (Studienanfänger = Gruppe A; Studierende in höheren Semestern = Gruppe B) unterschieden werden (vgl. Tab. 1). 23 der 32 Studierenden bei den Studienanfängern können zwei Gruppen (A5 und A7) zugeordnet werden. Über 50 Prozent der Studierenden in höheren Semestern können einer Gruppe (A6)

zugeordnet werden. Die Tabelle 1 stellt sowohl die Gruppen A1 bis A7 als auch die Gruppen B1 bis B6 zu den unterschiedlichen Zeitpunkten der Messung Z1 bis Z3 in Abhängigkeit zur jeweiligen Phase dar. Bei den Ergebnissen handelt es sich um aggregierte Mittelwerte der Studierenden innerhalb einer Gruppe. Zur besseren Übersicht wurde die Tabelle in unterschiedlichen Graustufen eingefärbt. Je dunkler eine Zelle eingefärbt ist, desto höher ist die Zustimmung zu einer der genannten Phasen und je heller eine Zelle eingefärbt ist, desto stärker ist die Ablehnung zu dieser Phase. Zudem wird in den Spalten mit den Überschriften (E) die Entwicklung bzw. Veränderung der Zustimmung oder Ablehnung zum vorherigen Messzeitpunkt durch Pfeile aufwärts, abwärts oder durch ein Gleichheitszeichen deutlich gemacht.

Die Veränderung der emotionalen Lage wird in Zusammenhang mit den Aussagen betrachtet, was den Studierenden in der untersuchten Woche besonders gut bzw. gar nicht am Studium gefallen hat. Die insgesamt 690 Antworten (332 positive und 358 negative) umfassen die nachfolgend dargestellten neun Dimensionen: Lehr- und Lernformate (n=215), Studien- und Prüfungsleistungen (n=135), Arbeitsbelastung (n=128), Individuum (n=80), Kommunikation (n=73), Technik (n=30), Organisation (n=12), Arbeitsplatz (n=11) und Lebensumfeld (n=6). Die größte Dimension bilden die Lehr- und Lernformate ab. Zu den positiven Einflussfaktoren zählen unter anderem interessante Lehrmethoden wie Labore, Offline-Meetings und Live-Vorlesungen oder Kurzvorträge. Hingegen gehören zu den negativen Einflussfaktoren langweilige Vorlesungen, eine ungünstige Kombination von synchronen und asynchronen Elementen in der Vorlesung oder ein unregelmäßiger Upload von Vorlesungsunterlagen. Die zweite Dimension bildet die Studien- und Prüfungsleistung, die sich im Positiven durch Fairness in den Prüfungen und frühzeitige Prüfungsvorbereitung auszeichnet. Negative Einflussfaktoren sind bspw. schlechte Organisation der Prüfungen oder technische Probleme. Als positive Einflussfaktoren der Arbeitsbelastung zählen das Zeitmanagement der Studierenden oder einfache Inhalte. Im Gegensatz dazu konnten physische Überlastung und Stress als negative Einflussfaktoren identifiziert werden. Zu den positiven Einflussfaktoren in der Dimension Individuum zählt die Weihnachtszeit und das erwartete Ende des Semesters. Negative Einflussfaktoren in dieser Dimension sind die mangelnde Motivation oder Krankheiten. Neben der Teamarbeit zählt die online Interaktion zu den positivsten Einflussfaktoren in der Dimension Kommunikation. Der mangelnde Austausch unter den Studierenden bspw. in Gruppenarbeiten zählt zu den negativen Einflussfaktoren. Die Technik bildet die 6. Dimension. Zu den positiven Einflussfaktoren zählt bspw., dass die Lehrenden entsprechende technische Fähigkeiten besitzen. Negative Einflussfaktoren sind bspw. Videos mit schlechter Qualität oder Ausfallzeiten des Lernmanagementsystems. Positive Einflussfaktoren bei der Dimension Organisation sind bspw. Informationsveranstaltungen zu Auslandsstudiengängen. Negative Einflussfaktoren, die diese Dimension prägen, sind fehlende Fristsetzungen seitens des Prüfungsamtes oder zu wenig Informationen. Die Möglichkeit auch von Zuhause aus zu arbeiten wird als positiver Einflussfaktor bei der Dimension Arbeitsplatz genannt. Der eingeschränkte Zugriff auf digitale Bücher wird als negativer Faktor dieser Dimension erwähnt. Das Lebensumfeld bildet die neunte Dimension, deren positive Einflussfaktoren durch das Campusleben in Form von

G	A	Z	I	E	II	E	III	E	IV	E	V	E	VI	E
A1	1	Z1	6		1		5		1		6		5	
		Z2	5	↓	1	=	5	=	1	=	5	↓	5	=
		Z3	6	↑	1	=	4	↓	1	=	4	↓	5	=
A2	3	Z1	3		1,33		4,33		4		4,67		5	
		Z2	5,33	↑	1	=	4	↓	1	↓	5,33	↑	5,67	↑
		Z3	3	↓	1	=	4	=	3,33	↑	5,33	=	5	↓
A3	2	Z1	3,5		1,5		4		1,5		5		5,5	
		Z2	4	↑	2,5	↑	4,5	↑	4	↑	5,5	↑	5,5	=
		Z3	5,5	↑	2	↓	4,5	=	5,5	↑	4,5	↓	5,5	=
A4	2	Z1	4,5		1,5		5,5		5		6		6	
		Z2	3	↓	1,5	=	4,5	↓	4,5	↓	6	=	5,5	↓
		Z3	2	↓	3	↑	4,5	=	5,5	↑	5	↓	5	↓
A5	10	Z1	1,2		4,3		4,8		5,2		4,4		5,1	
		Z2	1	↓	3,8	↓	4,9	↑	5,3	↑	4,1	↓	5,1	=
		Z3	1,3	↑	3,8	=	5,3	↑	5	↓	4,5	↑	5,3	↑
A6	1	Z1	1		4		2		4		5		3	
		Z2	5	↑	2	↓	4	↓	4	=	4	↓	3	=
		Z3	6	↑	3	↑	4	=	5	↑	4	=	4	↑
A7	13	Z1	2,15		3,15		3,77		4,08		4,46		4,38	
		Z2	2,08	↓	3,38	↑	3,62	↓	4,46	↑	4	↓	4,69	↑
		Z3	1,92	↓	3,54	↑	4	↑	4,46	=	3,85	↓	3,92	↓
B1	5	Z1	1,6		5		4,8		5,4		5		5,2	
		Z2	1,2	↓	4,8	↓	5	↑	5,6	↑	4,8	↓	5,4	↑
		Z3	1,2	=	5	↑	5,2	↑	5,4	↓	5,8	↑	5,6	↑
B2	6	Z1	1		5		5,33		5,33		2,83		4,5	
		Z2	1,5	↑	4,83	↓	5	↓	5,5	↑	3	↑	4,67	↑
		Z3	1	↓	5,67	↑	4,67	↓	5,67	↑	2	↓	3,33	↓
B3	1	Z1	2		1		4		1		6		6	
		Z2	6	↓	1	=	6	=	1	=	6	=	6	=
		Z3	6	=	1	=	6	=	1	=	6	=	6	=
B4	3	Z1	5,33		1		3,67		1		5,33		5,33	
		Z2	3,67	↓	2	↑	4	↑	2	↑	4,67	↓	4,67	↓
		Z3	3,67	=	1	↓	4,67	↑	2,33	↑	5	↑	5	↑
B5	1	Z1	2		3		5		5		4		4	
		Z2	5	↑	1	↓	5	=	1	=	2	=	2	↓
		Z3	4	↓	1	=	5	=	1	=	2	=	4	↑
B6	19	Z1	2,68		3,11		4,63		4,16		4,32		5,05	
		Z2	2,89	↑	3	↓	4,37	↓	4	↓	4,32	=	4,74	↓
		Z3	2,37	↓	2,89	↓	4,63	↑	4,53	↑	4,58	↑	4,74	=

I = Verleugnung; II = Wut; III = Verhandlung; IV = Depression; V = Erkundung; VI = Akzeptanz;
 G = Gruppe; A = Anzahl; E = Entwicklung; Z = Zeitpunkt; ↑ = steigend; ↓ = fallend; “=” = gleichbleibend

Tab. 1: Darstellung des emotionalen Wandels der Studierenden

gemeinsamen Mittagessen geprägt wird. Negativ werden die fehlenden Möglichkeiten an sozialen Events, wie Weihnachtsmärkten, teilzunehmen genannt.

5 Diskussion

Im Folgenden werden die emotionalen Phasen in Bezug zu den negativen und positiven Aussagen der Studierenden diskutiert. Aufgrund des begrenzten Platzes werden die beiden größten Gruppen der Studienanfänger:innen (A5, n=10, i. V. m. A7, n=13) und der fortgeschrittenen Studierenden (B6, n=19) vergleichend betrachtet, da diese über 50 Prozent der Proband:innen darstellen.

Gruppe A5 & A7: Diesen Gruppen ist bewusst, dass für sie interessante Lehr- und Lernformate bedingt durch die Corona-Pandemie entfallen. Somit befinden sie sich zu allen Messzeitpunkten nicht in der Phase *Verleugnung*. Als interessante Lehrveranstaltungen, die entfallen sind, wurden z.B. Laborpraktika genannt, die durch Videos ersetzt werden mussten. Zu der Phase *Wut* kann eine stärkere Zustimmung über alle drei Messzeitpunkte festgesetzt werden. Besonders zu Beginn des Semesters waren die Studierenden wütender darüber, dass einzelne Lehr- und Lernformate ausfallen und auch nicht in digitale Formate transformiert wurden. Vom Zeitpunkt Z1 bis Z3 steigt die Zustimmung der Studierenden aus beiden Gruppen zur Phase *Verhandlung*. Die Zustimmung der Gruppe A7 fällt marginal gerniger aus als bei Gruppe A5. Beide Gruppen wünschen sich einen Kompromiss, dass interessante Lehr- und Lernformate beibehalten werden. So wurden im Laufe des Semesters einige Formate kontinuierlich ins Digitale überführt. Positiv bewertet wurden technische Möglichkeiten, wie Sprungmarken in den Vorlesungsvideos, welche die Studierenden für die Prüfungsvorbereitung nutzen können. Es kann festgestellt werden, dass die Studierenden beider Gruppen eine hohe Zustimmung zur Phase *Depression* aufweisen. Die Gruppe A7 weist ein niedrigeres Niveau der Zustimmung auf und die Zustimmung nimmt über die Messzeitpunkte Z1 bis Z3 zu. Hingegen sinkt die Zustimmung bei der Gruppe A5 zum Ende des Semesters. Die Gruppen A5 und A7 finden es schade, dass interessante Lehr- und Lernformate entfallen oder stattdessen digital stattfinden, wie Musiklehre oder Laborarbeit. Ebenfalls ist eine Zustimmung zur Phase *Erkundung* beider Gruppen erkennbar. Zu Beginn der Messung waren die Studierenden noch offener für neue Lehr- und Lernformate. In der Gruppe A5 wurde benannt, dass erst während des Semesters eine digitale Transformation der Lehr- und Lernformate stattgefunden hat und somit erst zum Ende des Semesters diese Formate genutzt werden konnten. Eine geringere Zustimmung der Gruppe A7 kann begründet werden durch unregelmäßig, geballten oder gar nicht bereitgestellte Lehrmaterialien. Zudem bemängelten die Studierenden bspw. die Vorlesegeschwindigkeit in den Videos oder die unzureichende Qualität dieser. Bedingt durch die Corona-Pandemie und die digital geprägten Lehr- und Lernformate sowie die elektronischen Prüfungen am Ende des Semesters, mussten die Studierenden sich zwangsläufig bis zum Zeitpunkt Z3 auf die neuen Regelungen einstellen. Dies zeigt sich durch die dauerhafte Zustimmung zur Phase *Akzeptanz*.

Gruppe B6: Für die fortgeschrittenen Studierenden der Gruppe B6 kann im Vergleich zu den Studienanfänger:innen (Gruppe A5 und A7) eine etwas stärker ausgeprägte Zustimmung zur Phase *Verleugnung* festgestellt werden. Die Gruppe B6 hat den Schock nicht gänzlich verabreitet, dass einzelne Veranstaltungen ausfallen. Eine sinkende Zustimmung vom Zeitpunkt Z1 bis zu Z3 kann für die Phase *Wut* identifiziert werden. Am Anfang waren die Studierenden noch wütender (*Wut*) darüber, dass einige interessante Veranstaltungen ausgefallen sind, jedoch konnten diese im Laufe des Semesters transformiert werden und somit doch stattfinden. Zum Ende des Semesters stellten sich bei den Studierenden der Gruppe B6 Routinen ein und es entstand eine Art Monotonie beim Ansehen der Videos. Die Studierenden wurden während und zum Ende des Semesters immer unzufriedener mit den digitalen Lehr- und Lernformaten und die Motivation mitzuarbeiten hat nachgelassen. Für die Phase *Verhandlung* zeigt sich gleich hohe Zustimmung zu den Zeitpunkten Z1 und Z3. Zum Zeitpunkt Z2 ist die Zustimmung etwas geringer. Während des Semesters, zum Zeitpunkt Z2, haben die Studierenden aus Gruppe B6 sich die Lehr- und Lernformate in Präsenz mehr zurückgewünscht als zu Beginn. Eine positive Auswirkung auf die Emotionen der Studierenden hatte zum Ende des Semesters ein zusätzliches Angebot an Fragestunden, um auf Prüfungsfragen der Studierenden einzugehen. Eine leicht zunehmende Zustimmung über die drei Messzeitpunkte hinweg ist für die Phase *Depression* zu vermerken. Die Studierenden finden es schade, dass interessante Lehr- und Lernformate entfallen sind. Jedoch konnten die Studierenden zum Ende des Semesters feststellen, dass bedingt durch die Covid-19-Pandemie positive Gelegenheiten entstanden sind, neue Lehr- und Lernformate kennenzulernen. Dies zeigt sich durch die hohe Zustimmung in der Phase (*Erkundung*). In der Phase *Akzeptanz* wird deutlich, dass die Studierenden in kommenden Semestern eher mit Präsenzvorlesungen rechnen, aber sich auf digitale Lehr- und Lernformate eingestellt haben bzw. einstellen würden.

Insgesamt konnten bei den identifizierten Einflussfaktoren für die hier diskutierten Gruppen A5, A7 und B6 mehr negative als positive Aussagen festgestellt werden, sodass bei der reinen Anzahl deutlich wird, dass die Studierenden die Präsenzvorlesungen vermissen. Dies ist nicht verwunderlich, da Sie sich für eine Präsenzuniversität entschieden haben.

6 Fazit

Die kurzfristige Umstellung der Lehre an Universitäten hat zu einem Erfahrungsgewinn sowohl für die Studierenden als auch für die Lehrenden geführt, die ein völlig digitales Semester mit teilweise neuen digitalen Lehr- und Lernformaten kennengelernt haben. Während des Semesters hatten die beiden Gruppen Zeit, Erfahrungen zu sammeln, bis zum Ende des Semesters aus diesen zu lernen und die neue, digital geprägte Normalität zu akzeptieren. Grundlage für die Erforschung der emotionalen Veränderungen von Studierenden bildet eine umfassende Literaturanalyse. Die Ergebnisse der Auswertung belegen, dass es verschiedene Möglichkeiten bzw. Bewertungsinstrumente zur Messung von Emotionen bei Studierenden gibt, jedoch beziehen sich die meisten der identifizierten Fallstudien auf

einen Untersuchungszeitpunkt, statt auf einen Untersuchungszeitraum. Zusammenfassend lässt sich festhalten, dass die veränderten digitalen Lehr- und Lernformate zu den größten Einflussfaktoren zählen, die bei Studierenden während des Semesters festgestellt wurden und deren Emotionen beeinflusst haben. Aufgrund der positiven und negativen Ergebnisse innerhalb der Gruppen, können jedoch keine großen Veränderungen bei den Emotionen der Studierenden festgestellt werden, da die Mehrzahl der befragten Studierenden auch mit dem digitalen Semester und den damit verbundenen Veranstaltungen zufrieden war. Neben den digitalen Lehr- und Lernformaten spielten vor allem die Prüfungen und die zunehmende Arbeitsbelastung eine große Rolle für die Studierenden. Zudem zeigten sich nach einem Semester bei einer Gruppe (B5) bereits die körperlichen Folgen bei falscher Ausstattung im Home-Office. Mithilfe des gewählten Untersuchungsdesigns konnten Veränderungen der Emotionen während der Pandemie gesammelt werden. Zukünftig kann diese Methode in ausgewählten Lehr- und Lernformaten als Pilotprojekt umgesetzt werden. Gerade in größeren Veranstaltungen kann dies eine effektive Möglichkeit darstellen, um überforderte Studierende zu identifizieren und zielgerichtet mit besonderen Dienstleistungen zu unterstützen. Der durchgeführte Fragebogen der emotionalen Veränderung stellt eine zuverlässige Methode dar, mit der nachgewiesen werden kann, wie Studierende, in unserem Fall vor allem Studienanfänger:innen und Studierende in höheren Fachsemestern, die Situation des Beginns oder die Fortführung eines völlig digitalen Studiums bewältigen. Die Limitationen der Arbeit resultieren zum einen aus den Untersuchungsgegenständen in Form der Studierenden, die freiwillig an der Studie teilgenommen haben. Es wurden nur Studierende berücksichtigt, die neben ihrem Studium Zeit für die Umfrage hatten und engagiert genug waren diese auszufüllen. Des Weiteren konnten keine Studierenden berücksichtigt werden, die während des Semesters das Studium abgebrochen haben. Insgesamt nahmen 70 Proband:innen an der qualitativen Studie teil. Zusätzlich sollte die Covid-19-Pandemie als Limitation gesehen werden, die sowohl im universitären als auch im privaten Umfeld der Studierenden zu einer Veränderung der Situation geführt hat und somit auch Emotionen ausgelöst hat. Die vorgestellte Messung der Emotionen der Studierenden könnte als Reflexionsmethode in der Lehre verwendet werden, um negative Einflussfaktoren der Lehr- und Lernformaten zu minimieren und die positiven Einflussfaktoren zu verstärken. Außerdem könnte die Studie robuster gemacht werden, indem bspw. verschiedene Altersgruppen und verschiedene Universitäten diese Methode durchführen.

Literatur

- [ACS21] Arora, S.; Chaudhary, P.; Singh, R. K.: Impact of coronavirus and online exam anxiety on self-efficacy: The moderating role of coping strategy. *Interactive Technology and Smart Education* 18/3, S. 475–492, 2021.
- [AI20] Alemany-Arrebola, I.; Rojas-Ruiz, G.; Granda-Vera, J.; Mingorance-Estrada, Á.C.: Influence of COVID-19 on the perception of academic self-efficacy, state anxiety, and trait anxiety in college students. 11/, 2020.

- [BDR03] Bolger, N.; Davis, A.; Rafaeli, E.: Diary methods: Capturing life as it is lived. *Annual Review of Psychology* 54/1, S. 579–616, 2003.
- [Be04] Becker, L.; Beukes, L. D.; Botha, A.; Botha, A.; Botha, J. J.; Botha, M.; Cloete, D. J.; Cloete, J. L.; Coetzee, C.; De Beer, L. et al.: The impact of university incorporation on college lecturers. *Higher Education* 48/2, S. 153–172, 2004.
- [Bl08] Blau, G.: Exploring antecedents of individual grieving stages during an anticipated worksite closure. *Journal of Occupational and Organizational Psychology* 81/3, S. 529–550, 2008.
- [CC20] Cuschieri, S.; Calleja Agius, J.: Spotlight on the shift to remote anatomical teaching during Covid-19 pandemic: Perspectives and experiences from the University of Malta. *Anatomical Sciences Education* 13/6, S. 671–679, 2020.
- [DFR21] Del Arco, I.; Flores, Ò.; Ramos-Pla, A.: Structural model to determine the factors that affect the quality of emergency teaching, according to the perception of the student of the first university courses. *Sustainability* 13/5, 2021.
- [DSB21] Dikaya, L.; Shipitko, O.; Borokhovski, E.: Psychological microclimate of student groups, studying in different instructional formats. In: *E3S Web of Conferences*. Bd. 258, EDP Sciences, 2021.
- [Du20] Duong, V.; Luo, J.; Pham, P.; Yang, T.; Wang, Y.: The ivory tower lost: How college students respond differently than the general public to the covid-19 pandemic. In: *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, S. 126–130, 2020.
- [EAT15] El-Abbasy, K.; Angelopoulou, A.; Towell, A.: Affective computing to enhance e-Learning in segregated societies. In: *2015 Imperial College Computing Student Workshop (ICCSW 2015)*. Bd. 49, OpenAccess Series in Informatics, S. 13–20, 2015.
- [ET02] Elrod, P. D.; Tippet, D. D.: The “death valley” of change./, 2002.
- [Fr18] Frei, M.: *Change Management für Führungskräfte: Eine Praxisanleitung zur betrieblichen Transformation*. Vahlen, 2018.
- [Gi20] Giusti, L.; Salza, A.; Mammarella, S.; Bianco, D.; Ussorio, D.; Casacchia, M.; Roncone, R.: Everything will be fine. Duration of home confinement and “all-or-nothing” cognitive thinking style as predictors of traumatic distress in young university students on a digital platform during the COVID-19 Italian lockdown. *Frontiers in Psychiatry* 11/1398, 2020.
- [Ha09] Hair, J. F.; Black, W.; Babin, B. J.; Anderson, R. E.: *Multivariate Data Analysis*. 7. Auflage. Pearson, 2009.
- [Ha21] Harouni, H.: Unprepared humanities: A pedagogy (forced) online./, S. 1–16, 2021.

- [HWO21] Hagedorn, R. L.; Wattick, R. A.; Olfert, M. D.: “My entire world stopped”: College students’ psychosocial and academic frustrations during the COVID-19 pandemic. In: *Applied Research in Quality of Life*. Springer Nature, S. 1–22, 2021.
- [Ke21] Kee, C. E.: The impact of COVID-19: Graduate students’ emotional and psychological experiences. *Journal of Human Behavior in the Social Environment* 31/1-4, S. 476–488, 2021.
- [Ku73] Kuebler-Ross, E.: *On Death and Dying*. Routledge, 1973.
- [Li21] Liu, C.; McCabe, M.; Dawson, A.; Cyrzon, C.; Shankar, S.; Gerges, N.; Kellett-Renzella, S.; Chye, Y.; Cornish, K.: Identifying predictors of university students’ wellbeing during the COVID-19 pandemic - A data-driven approach. *International Journal of Environmental Research and Public Health* 18/13, 2021.
- [Oh10] Ohly, S.; Sonnentag, S.; Niessen, C.; Zapf, D.: Diary studies in organizational research. *Journal of Personnel Psychology* 9/2, 2010.
- [PSR20] Polujanski, S.; Schindler, A.-K.; Rothhoff, T.: Academic-associated emotions before and during the COVID-19-related online semester - a longitudinal investigation of first-year medical students. *GMS Journal for Medical Education* 37/7, 2020.
- [Qa20] Qanash, S.; Al-Husayni, F.; Alemam, S.; Alqublan, L.; Alwafi, E.; Mufti, H. N.; Qanash, H.; Shabrawishi, M.; Ghabashi, A.: Psychological effects on health science students after implementation of COVID-19 quarantine and distance learning in Saudi Arabia. *Cureus* 12/11, 2020.
- [RMD14] Ribeiro, S.; Moreira, A.; Da Silva, C. P.: *Digital Storytelling: Emotions in Higher Education*. International Association for Development of the Information Society, 2014.
- [St97] Streich, R. K.: Veränderungsprozessmanagement. In (Reiß, M.; Rosenstiel, L. v.; Lanz, A., Hrsg.): *Change-Management: Programme, Projekte und Prozesse*. Bd. 1, Schäffer-Poeschel, S. 237–254, 1997.
- [Wi08] Wiggins, L.: Managing the ups and downs of change communication. *Strategic Communication Management* 13/1, S. 20–23, 2008.
- [Ze08] Zembylas, M.: Adult learners’ emotions in online learning. *Distance Education* 29/1, S. 71–87, 2008.
- [Zh20] Zhang, K.; Wu, S.; Xu, Y.; Cao, W.; Goetz, T.; Parks-Stamm, E. J.: Adaptability promotes student engagement under COVID-19: The multiple mediating effects of academic emotion. In: *Frontiers in Psychology*. Bd. 11, Frontiers Media SA, 2020.

Methodenvermittlung des Software Engineerings als Hebel für die Digitale Transformation der Öffentlichen Verwaltung

David Zellhöfer¹

Abstract: Die Öffentliche Verwaltung erfährt aktuell einen hohen Digitalisierungsdruck. Hierbei ist abzusehen, dass es nicht reicht, ein agiles Mindset zu entwickeln. Vielmehr müssen technologische Kompetenzen in der Breite aufgebaut werden, um an der Schnittstelle Fachlichkeit/Software-Entwicklung bestehen zu können, damit nutzerorientierte, digitale Dienste entwickelt werden können.

Der vorliegende Artikel benennt Kernherausforderungen der Digitalen Transformation für die Öffentliche Verwaltung und präsentiert einen auf die grundständige Lehre bezogenen Lösungsansatz am Beispiel der Verwaltungsausbildung an der Hochschule für Wirtschaft und Recht Berlin, um diesen Anforderungen langfristig gerecht zu werden. Hierbei werden Teilbereiche des Software und Usability Engineerings vermittelt, um die Kommunikation im Software-Entwicklungsprozess zu vereinfachen, damit zukünftige in der Verwaltung tätige Personen erfolgreich an deren Digitalen Transformation mitwirken können.

Keywords: Digital Transformation; User-Centered Design; Agile; Professional Competencies

1 Einführung

Der Druck auf die Öffentliche Verwaltung (ÖV), sich den Herausforderungen der Digitalen Transformation zu stellen, nimmt stetig zu: sei es durch den Gesetzgeber in Form des Onlinezugangsgesetzes (OZG)² oder Verordnungen wie der BITV 2.0³, Open-Data-Initiativen der Zivilgesellschaft, wie *bund.dev* [Bu21] oder durch sich weiterentwickelnde Bedürfnisse und Anforderungen von Nutzer*innen – zuletzt auch durch die Corona-Pandemie.

Besonders stark zeigte sich der Digitalisierungsdruck Ende der 80er Jahre in den Einrichtungen des GLAM-Bereichs⁴, welche zum großen Teil in der öffentlichen Hand liegen. Dies kann besonders am Beispiel wissenschaftlicher Bibliotheken illustriert werden. Mit dem Verfügbarwerden leistungsfähiger Hard- und Software im Back- und Frontendbereich, digitalisierte man hier Arbeitsprozesse wie die Medienkatalogisierung oder die Magazinsteuering bereits frühzeitig. Dieser Transformationsprozess hält seitdem mit den Entwicklungen disruptiver Technologien wie dem World Wide Web Schritt und spiegelt sich auch in sich

¹ Hochschule für Wirtschaft und Recht, Digitale Innovation in der öffentlichen Verwaltung, Alt-Friedrichsfelde 60, 10315 Berlin, david.zellhoefer@hwr-berlin.de

² Gesetz zur Verbesserung des Onlinezugangs zu Verwaltungsleistungen

³ Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0)

⁴ Galleries, libraries, archives and museums

parallel entwickelnden Berufsbildern wie der/dem Systembibliothekar*in wider, welche an der Schnittstelle zwischen Bibliothek und Software-Entwicklung arbeiten und Kompetenzen beider Berufsfelder vereinen.

Es ist deshalb kaum verwunderlich, dass moderne Bibliotheken kaum mehr von ihren primär Software-basierten Prozessen trennbar sind, wie Hanson treffend darstellt:

Because the library has become software, it is no longer viable for our services to exist separately from our software. Our best opportunities for intervention, for reference, and for instruction are within our software. Our users succeed every day in information seeking online, as they shop, listen to music, register for courses, and interact through social media. In none of these cases do our users expect that services vital to the endeavor exist outside of the software in front of them. Rather, the software is the service. [Ha15]

Die dauerhafte und nutzerorientierte Erbringung von Dienstleistungen ist demnach eng mit typischen Feldern des Software Engineerings wie dem Usability und Requirements Engineering, dem User-Centered [GL87] oder agilen [BG01] Design verbunden, was sich auch in der Ausbildung von Bibliothekar*innen und Bibliothekswissenschaftler*innen zeigen lässt [HU17].

Das einführende Beispiel zeigt, dass es möglich ist, Berufsbilder fortzuentwickeln und Schnittstellen zwischen der Fachlichkeit und dem Software Engineering, welches die Digitale Transformation maßgeblich gestaltet, zu schaffen. In der Breite der ÖV sind diese Entwicklungen jedoch noch nicht zu beobachten. Hier fehlt es an zentralen Kompetenzen, sowohl bei den Beschäftigten als auch in der Ausbildung [MBH21].

Das Hauptaugenmerk des vorliegenden Artikels ist deshalb auf Lehrinhalte gerichtet, die Studierende der ÖV zukünftig dazu zu befähigen sollen, besser während der Digitalen Transformation und deren verwaltungsspezifischen Herausforderungen bestehen zu können.

2 Kernherausforderungen für die Öffentliche Verwaltung

Obwohl sich Deutschland im “Digital Quality of Life-Index” [Su21] von 2021 auf Platz neun von 110 betrachteten Ländern findet und sich damit um sieben Plätze binnen eines Jahres verbessert, lohnt sich eine tiefere Befassung mit den zugrundeliegenden Einzelindikatoren. Beim Indikator “Electronic Government”, welcher die Reife digitaler Verwaltungsleistungen abbildet, erreicht Deutschland noch den 20. Platz. Hierbei ist anzumerken, dass diese Platzierung im Wesentlichen auf die Gewichtung des “Artificial Intelligence Readiness Index” (Platz vier im diesbezüglichen Ranking) zurückzuführen ist, welcher hauptsächlich durch die strategischen Förderprogramme der Großen Koalition bis 2021 beeinflusst wird.

Betrachtet man nur den “Online Service Index”, welcher nutzerfreundliche, digitale Verwaltungsdienstleistungen repräsentiert und damit die Ziele des OZG widerspiegelt, erreicht Deutschland nur den 56. Platz im internationalen Ranking.

Auch wenn das OZG und der IT-Planungsrat [IT18] klare Vorgaben machen, dass digitale Dienstleistungen nutzerorientiert zu entwickeln sind, muss davon ausgegangen werden, dass die dafür nötigen Kompetenzen in der Breite der ÖV nicht zur Verfügung stehen. Dies liegt zum einen darin begründet, dass Digitalisierungsthemen häufig als reines IT-Thema verstanden wurden, andererseits darin, dass sich der Outsourcing-Trend in der ÖV, insbesondere im IT-Bereich, zum Innovationshemmnis durch die verstärkte Abhängigkeit von (IT-)Dienstleistern entwickelt hat [Sch21]. Hinzu kommt die schiere Komplexität der zu digitalisierenden Leistungen. Aktuell benennt der OZG-Umsetzungskatalog 575 Verwaltungsdienstleistungen, die in 14 Themenfelder, wie “Bildung” oder “Steuern & Zoll” unterteilt sind. Hier gilt es, entsprechende Kompetenzen zur Erhebung fachlicher wie technischer Anforderungen und zur Begleitung nutzerzentrierter digitaler Transformationsprozesse an der Schnittstelle zur Informatik bereitzustellen bzw. zu entwickeln, um auf Augenhöhe kommunizieren zu können, wie es auch eine aktuelle Studie [MBH21] belegt.

Betrachtet man die Diskussion innerhalb der ÖV, so wird schnell deutlich, dass der Begriff der Agilität, der im Software-Engineering nun seit 20 Jahren bekannt ist [BG01], quasi als neuer Allheilsbringer im Rahmen der Digitalen Transformation der ÖV betrachtet wird [MGW21]. Hierbei wird häufig verkannt, dass es jedoch mehr als einem agilen Mindset bedarf, um an der technischen Schnittstelle zur Software-Entwicklung zu bestehen und in einem interdisziplinären Team nutzerorientierte Services zu entwickeln. So zeigt z.B. [Ze21] auf, welche Herausforderungen sich durch agiles Arbeiten in hierarchischen Organisationen der ÖV ergeben. Dabei benennt der Autor, neben weit verbreiteten Problemen, wie dem Projektanbahnungs- und Multiprojektmanagement, auch Aspekte des Haushalts- und Beschaffungsrechts sowie den Weiterbildungsbedarf für die Fachlichkeit, um etablierte, agile Rollen wie die des Product Owners auszufüllen, welche aufgrund des stets steigenden Aktualisierungs- und Innovationsdrucks kaum durch die IT alleine, z.B. durch Proxy Product Owner, auszufüllen sind.

Die oben genannten Beispiele bilden die Basis für die Kernherausforderungen der digitalen Transformation der ÖV, die wie im folgendem kurz umrissen werden können:

1. *Komplexität*: Die Komplexität und der Umfang der zu digitalisierenden Verwaltungsdienstleistungen ist enorm.
2. *“Professional” Digital Literacy*: Durch das massive IT-Outsourcing innerhalb der ÖV sind Kompetenzen aus dem Bereich der *“professional” digital literacy* (s.u.) kaum verfügbar bzw. sind Schnittstellen hin zur Software-Entwicklung und zum IT-Betrieb unterentwickelt. Digitalisierte Dienstleistungen werden häufig als Aufgabe für die IT gesehen; eine Identifikation der Fachabteilungen mit den IT-basierten Diensten fehlt.
3. *Nutzerorientierung*: Verwaltungsdienstleistungen sollen zukünftig nutzerorientiert

gedacht werden [IT18] und sich dynamisch an die Bedürfnisse der Nutzenden anpassen. Verwaltungsinterne Prozesse sind deshalb nicht mehr primäre Quelle für Anforderungen.

4. *Strukturelle Herausforderungen:* Strukturelle Gegebenheiten, wie die vorherrschende hierarchische Arbeitsweise, stehen transparentem, agilem Arbeiten und dem Erwartungsmanagement im Projekt- und Multiprojektmanagement entgegen.

Bereits anhand des Umfangs der zu digitalisierenden Verwaltungsdienstleistungen und der damit verbundenen Komplexität wird deutlich, dass die ÖV diese Aufgabe nicht allein mit verwaltungswissenschaftlichen und juristischen Kompetenzen bestreiten können wird. Der wachsende Innovationsdruck ist zeitgleich nicht allein von “der IT” zu bewältigen. Es bedarf folglich einer Weiterqualifizierung an der Schnittstelle ÖV/IT, welche ebenso die Bedürfnisse von Bürger*innen und Unternehmen nicht aus dem Blick verliert und deren Standpunkt aktiv in den Entwicklungsprozess miteinbezieht.

3 Schnittstellenkompetenzen in der Verwaltungsbildung

Im Rahmen der Neuausrichtung der Studiengänge Öffentliche Verwaltung bzw. Recht in der Öffentlichen Verwaltung an der Hochschule für Wirtschaft und Recht Berlin wurde deshalb das Modul “Digitalisierung in der Verwaltung” etabliert, in dem technologische Kompetenzen vermittelt werden, die es zukünftigen Beschäftigten in der ÖV – im Sinne einer “*professional digital literacy*” – ermöglichen sollen, effektiv und auf Augenhöhe mit Software-Entwickler*innen zusammenzuarbeiten. Hierbei werden sowohl theoretische und praktische Ansätze des Software- und Usability-Engineerings als auch der konkrete Umgang mit einschlägigen Kollaborationswerkzeugen innerhalb eines seminaristischen Lehrangebots vermittelt. Ein besonderes Augenmerk wird hierbei auf Prinzipien des User-Centered Designs gelegt, da der Gesetzgeber explizit die Nutzerorientierung als Zielstellung für die erfolgreiche Digitalisierung von Verwaltungsdienstleistungen im Rahmen des Onlinezugangsgesetzes benennt [IT18].

Auch wenn eine der Kernherausforderungen die enorme Komplexität der Digitalen Transformation der ÖV darstellt, findet sich dazu bei den im folgenden vorgestellten Kursinhalten kein expliziter Lösungsansatz. Diese Kernherausforderung soll vielmehr implizit durch die Vermittlung von Schnittstellenkompetenzen adressiert werden, unter deren bisherigem Mangel die Digitale Transformation leidet. Es soll ein Hebel im Rahmen der Ausbildung geschaffen werden, der die Digitale Transformation der ÖV langfristig erleichtert.

3.1 Aktuelles Lehrangebot

Der aktuelle, seit dem Wintersemester 2020/21 semesterweise angebotene, Fächerkanon umfasst drei Kurse mit jeweils 2 ECTS, von den mindestens zwei zu wählen sind, um

das insgesamt 5 ECTS-Credits umfassende Modul zu absolvieren. Hinzu kommt eine Kurzreflexion im Umfang von mindestens 500 Worten, welche mit 1 ECTS angerechnet wird, um die Kursinhalte untereinander zu verknüpfen und deren Relevanz im Bereich der ÖV einzuordnen.

Im Rahmen dieser Arbeit wird der dritte, im gleichen Turnus angebotene Kurs “Digitalisierungsprozesse und -technologien” keine Betrachtung finden, da er sich primär im Spannungsfeld Digitalisierungspolitik, Digital Literacy, Datenschutz und Ethik bewegt. Hinzu kommen anlassbezogene Kursangebote bzw. Digitalisierungskurse aus dem Studium Generale.

3.1.1 Digitalisierungsprozesse nutzerorientiert gestalten

Dieser Kurs stellt ein Grundlagenmodul dar, welches Studierende in die Lage versetzt, erfolgreich an der Implementierung von nutzerorientierten Digitalisierungs- und IT-Projekten und deren Überführung in den Dauerbetrieb mitzuwirken. Hierbei werden sie auf ihre zukünftigen Rollen in Digitalisierungsprojekten im Austausch mit diversen Stakeholdern (mit besonderem Fokus auf Endnutzer*innen) sowie Software-Entwicklungs-Teams vorbereitet.

Dieser Kurs adressiert primär die Kernherausforderung 3 (*Nutzerorientierung*) und sekundär Kernherausforderung 2 (*“Professional” Digital Literacy*) (siehe Abschnitt 2).

Lernziele Zielstellung der Veranstaltung ist es, die Studierenden zu befähigen, nutzerorientierte Digitalisierungsprojekte zu entwickeln und den nutzerzentrierten Standpunkt gegenüber IT-Teams und Vorgesetzten zu vertreten. Hierbei lernen sie typische Werkzeuge der Software-Entwicklung bzw. des Projektmanagements kennen, die sich auch größtenteils für das Fachstudium nachnutzen lassen. Studierende sind nach dem Absolvieren dieses Kurses in der Lage, an der Digitalisierung einer Verwaltungsdienstleistung entsprechend der Vorgaben des IT-Planungsrats [IT18] mitzuarbeiten und die gesetzlichen Vorgaben in diesem Bereich anzuwenden bzw. IT-Dienstleistern zu vermitteln.

Fach- und Methodenkompetenzen Neben der historischen Einordnung des User-Centered Designs im Rahmen der Software-Entwicklung, werden den Studierenden Methoden zur nutzerzentrierten Entwicklung in Abgrenzung zur System-zentrierten Entwicklung vorgestellt. Sie erwerben die nötigen Kompetenzen, um die Usability, Accessibility und User Experience eines digitalen Angebots zu bewerten und Verbesserungsvorschläge zu formulieren. Begleitend lernen sie die vom Gesetzgeber definierten Anforderungen an IT-Dienste der ÖV kennen. Außerdem werden sie befähigt, selbständig einfache Usability-Evaluierungen, wie Think-Aloud-Protokolle oder Cognitive Walkthroughs, auszuwählen und durchzuführen.

Kursinhalte In Ergänzung zu den Prinzipien des User-Centered Designs und des Usability Engineerings lernen die Studierenden den klassischen Software Development Life Cycle und ihre wechselnden Rollen mit Bezug auf die Nutzerorientierung darin kennen. Als konkrete Methoden im Rahmen der nutzerorientierten Anforderungsanalyse und des Designs werden Techniken wie Personas, Scenarios, Storyboards und der allgemeine Design-Thinking-Prozess eingeführt. Ein besonderer Fokus liegt dabei auf der Barrierefreiheit, welche durch die BITV 2.0 für weite Teile der ÖV gesetzlich verpflichtend ist. Um die zukünftige Zusammenarbeit mit Softwareentwickelnden zu vereinfachen werden Werkzeuge wie z.B. Git/GitLab⁵, Auszeichnungssprachen, Wikis oder Kanban-Boards eingeführt.

3.1.2 Projekt- und Servicemanagement von Digitalisierungsprojekten

In diesem Kurs wird ein Service-orientierter Blick auf digitalisierte Verwaltungsprozesse und das kontinuierliche Verbesserungsmanagement geworfen. Hierbei wird betrachtet, wie innovative Digitalisierungsprojekte durchgeführt und dann in den Dauerbetrieb überführt bzw. abgelöst werden können.

Dieser Kurs adressiert primär die Kernherausforderung 4 (*Strukturelle Herausforderungen*) und sekundär Kernherausforderung 2 (*“Professional” Digital Literacy*) (siehe Abschnitt 2).

Lernziele Die Studierenden erlangen die Fähigkeit zur kooperativen Einführung und zum dauerhaften Betrieb und der geregelten Ablösung von Digitalisierungsprojekten. Ferner lernen sie den Software Development Life Cycle kennen und können ihre Rollen in den verschiedenen Phasen des Lebenszyklus eines IT-Diensts einordnen. Sie lernen zu bewerten, welche Projektmanagement-Methoden sich für verschiedene Aufgaben und Organisationen eignen und lernen dabei einige agile Werkzeuge kennen, die sich für das eigene Fachstudium und zukünftige Aufgaben nutzen lassen.

Fach- und Methodenkompetenzen Die Studierenden erlernen, das bestehende Projektmanagement- und Servicemanagement einer Organisation zu analysieren und Verbesserungsvorschläge abzuleiten sowie menschliche Einflussfaktoren zu moderieren und zu bewerten. Hierbei wird der Schwerpunkt auf die agile Transformation von hierarchischen Organisationen gelegt. Ferner lernen sie IT-bezogene Projektmanagement- und Servicemanagement-Methoden und Werkzeuge zu verstehen und erwerben erste Anwendungskompetenzen in diesen Bereichen.

Kursinhalte Abgeleitet aus dem Software Development Life Cycle entwickeln die Studierenden ein Verständnis für die typischen Herausforderungen in Digitalisierungsprojekten

⁵ Hierbei liegt der Lehrfokus auf den angebotenen Dokumentations- und Issue-Management-Werkzeugen.

wie dynamische Ziele und Anforderungen. Im Vergleich miteinander lernen sie die Vor- und Nachteile klassischer und agiler Methoden mit Hinblick auf die Digitale Transformation der ÖV kennen. Die Studierenden lernen Aspekte des Anforderungs-, Multiprojekt- und Risikomanagements kennen, wobei sowohl die Projektphasen als auch der Dauerbetrieb von Digitalisierungsprojekten thematisiert werden. Hierbei werden gängige Frameworks wie Scrum, Kanban, ITIL, Cobit und PRINCE2 vorgestellt, die sich prinzipiell für die Verwendung in der ÖV eignen.

3.2 Zukünftiges Lehrangebot

Ab dem Sommersemester 2022 wird der ÖV-Studiengang aufgrund des bereits dargelegten Bedarfs an Digitalisierungskompetenzen um den wählbaren Studienschwerpunkt “Digitalisierung und nutzerorientierte Verwaltungsinnovation” mit insgesamt 10 ECTS-Credits ergänzt, welcher im Folgenden kurz präsentiert wird. Voraussetzung für die Wahl dieses Studienschwerpunkts ist der Besuch des in Abschnitt 3.1.1 vorgestellten Kurses “Digitalisierungsprozesse nutzerorientiert gestalten”, um dem Fokus des Gesetzgebers auf die Nutzerorientierung gerecht zu werden.

3.2.1 Aktuelle Themen der Digitalen Transformation

In diesem Kurs wird eine Abgrenzung der Begriffe “Digitalisierung” und “Digitale Transformation” vorgenommen sowie Einflussfaktoren auf den Transformationsprozess, wie Digital Divide, Digital Literacy, Digital Sovereignty und ethische Fragestellungen thematisiert. Grundlage des Kurses bilden aktuelle Fallbeispiele und typische algorithmische Anwendungen der Digitalen Transformation.

3.2.2 Daten- und Prozessmanagement der Digitalen Transformation

Das aus §12a EGovG abgeleitete “Open Data-Gesetz” rückt die Bereitstellung von maschinell lesbaren Daten und (Linked) Open Data in den Fokus des Verwaltungshandelns. Der Erwerb dafür nötiger Kenntnisse, z.B. über den Data Life Cycle, die Erstellung von Data-Governance-Policies und typische Rollen, wie die des Data Stewards, stehen im Zentrum dieses Kurses. Technische Hintergründe des Big-Data-Managements (Datenbanken, Data Warehouses, Repository- und Information-Retrieval-System sowie Data Lakes) und die Anwendungsgebiete für maschinelles Lernen in diesem Bereich ergänzen die Kursinhalte.

3.2.3 Kollaboratives User Experience-Design

Dieser Kurs vertieft die in Abschnitt 3.1.1 erworbenen Kenntnisse anhand von Praxisprojekten und erweitert das Methodenwissen der Studierenden um weitere Ideation- und Prototyping-Techniken. Ein besonderes Augenmerk wird dabei auf die inklusive Gestaltung von Dienstleistungen inkl. der aktuellen gesetzlichen Grundlagen (z.B. OZG, EU-Richtlinie 2016/2102, BITV 2.0 etc.) und deren Evaluierung gelegt.

4 Ablauf und Evaluierungsauswertung der durchgeführten Kurse

Zum Zeitpunkt der Publikation wurden alle drei Kurse “Digitalisierungsprozesse nutzerorientiert gestalten”, “Projekt- und Servicemanagement von Digitalisierungsprojekten” und “Digitalisierungsprozesse und -technologien”, welcher hier keine Betrachtung findet (s.o.), in jedem Semester ausschließlich unter Corona-Bedingungen angeboten; das heißt: als vorrangig synchron angebotene Online-Veranstaltungen.

Im Falle des Kurses “Digitalisierungsprozesse nutzerorientiert gestalten” (DigiNutzer) werden je nach Semesterverlauf 1-2 Lerneinheiten asynchron angeboten, wobei Videoaufzeichnungen von kurzen Arbeitsaufgaben, die das Lernverständnis erhöhen sollen bzw. die Nutzung eines bereitgestellten Werkzeugs voraussetzen, unterbrochen werden. Die Bearbeitungsreihenfolge ist dabei linear vorgegeben und wird durch die Ablaufkontrolle der Moodle-Lernplattform durchgesetzt.

Der Kurs “Projekt- und Servicemanagement von Digitalisierungsprojekten” (ProjSer) ist wesentlich stärker auf den Dialog zwischen den Studierenden ausgelegt, die Praxiserfahrungen mit einbringen sollen⁶. Zwei Veranstaltungen werden im Format des Inverted Classrooms durchgeführt, während drei weitere Termine auf Arbeitsaufgaben zurückgreifen, welche die Studierenden in freier Zeitaufteilung bis zur jeweiligen Unterrichtswoche bearbeiten sollen.

Die Lehrveranstaltungen (LV) werden zusätzlich mit vertiefenden Lernmaterialien für das Selbststudium in Form von wissenschaftlichen Publikationen, Podcasts, Videos und relevanten Publikationen der Verwaltung (Whitepaper, Gesetze, Verordnungen etc.) und interaktiven Übungselementen wie Quizzes auf Moodle ergänzt. Alle Veranstaltungen werden live aufgezeichnet und mitsamt der Folien nach der jeweiligen LV online zur Verfügung gestellt.

Zu Semesterbeginn werden die Studierenden auf den aktuellen Beta-Status der Kursinhalte hingewiesen, um ihnen einerseits einen Einstieg in die Software-Entwicklungsterminologie zu bieten und andererseits die Möglichkeit zur nutzerorientierten Mitwirkung mit Hinblick auf die Gestaltung der Kursinhalte zu ermöglichen. Die letztgenannte Mitwirkungsmöglichkeit ist insbesondere deshalb wichtig, da es sich um eine sehr heterogene Studierendengruppe

⁶ Hierbei kann der Umstand, dass mindestens ein Drittel der Studierenden bereits in der ÖV tätig ist, genutzt werden. Ein weiterer Anteil an Studierenden verfügt bereits über Projekterfahrungen aus dem Studium.

aus zwei Studiengängen in den Semestern 1-7 handelt, welche kaum über eine professionelle IT-Vorbildung verfügt. Außerdem werden den Studierenden die Lernziele, die zu erwerbenden Fach- und Methodenkompetenzen sowie die groben Kursinhalte erläutert (siehe Abschnitt 3.1). Die Studierenden werden informiert, dass die LV jedes Semester evaluiert wird, um den Ansprüchen der Studierenden gerecht und auf dem Stand der Technik und der Gesetzgebung zu bleiben. Anhand dieser Mitwirkungsmöglichkeit und der regelmäßigen Evaluierung wird außerdem ein thematischer Bogen dazu geschlagen, was es heißt, nutzerorientierte Dienste zu gestalten. Insofern wird das Lehrangebot als “digitalisierte Verwaltungsdienstleistung” mit den Studierenden als Nutzerkreis eingeordnet. Im Semesterverlauf werden die Studierenden regelmäßig aufgefordert, im Forum oder per E-Mail Feedback zur LV zu geben. Individuelles Feedback wird während des gesamten Semesters entweder in der Folgeveranstaltung oder durch die Bereitstellung weiterer Informationsmaterialien (z.B. in Form selbstgezeichneter Animationsfilme) in Moodle aufgegriffen. Die eigentliche LV-Evaluierung seitens der Hochschule erfolgt zur Semesterhalbezeit, um auf Anregungen eingehen zu können und die Ergebnisse zu diskutieren. Tabelle 1 illustriert die Teilnahme- und Rücklaufquoten der durch die Hochschule durchgeführten Evaluierung.

Tab. 1: Anzahl an Teilnehmer*innen und Rücklaufquoten der jeweiligen LV

DigiNutzer: “Digitalisierungsprozesse nutzerorientiert gestalten”

ProjSer: “Projekt- und Servicemanagement von Digitalisierungsprojekten”

<i>Titel der Lehrveranstaltung</i>	<i>Semester</i>	<i>Anzahl</i>	<i>Rücklaufquote</i>
DigiNutzer	WiSe 20/21	40	65,0%
DigiNutzer	SoSe 21	58	50,0%
ProjSer	WiSe 20/21	20	61,5%
ProjSer	SoSe 21	29	55,2%

Auch wenn sich aus den zwei Iterationen der Kursdurchläufe noch keine allgemein gültigen Aussagen ableiten können, so lohnt doch ein Blick auf die in Tabelle 2 dargestellten Evaluationsergebnisse der Kurse. Aus der gemittelten Gesamtbeurteilung der Kurse auf einer Skala von 1 bis 6 (mit 6 als Bestwert) wird deutlich, dass die Kurse sehr positiv durch die evaluierenden Student*innen aufgenommen werden. Ein ähnliches Bild mit zeigt sich bei den beiden Bewertungsaspekten “Struktur und Didaktik” und “Lernmethoden und Lernmaterial”. Allerdings ist erkennbar, dass die Kursbewertung generell zum Sommersemester 2021 hin abnimmt.

Dieser Effekt wird besonders deutlich bei der “Interessantheit des Kurses” und der durch die Studierenden eingeschätzten Relevanz des Kurses bezüglich ihrer zukünftigen beruflichen Handlungsfähigkeit. Hier ergibt sich eine deutliche Verschlechterung der Bewertung von bis zu 1,7 Punkten für den Kurs “Projekt- und Servicemanagement von Digitalisierungsprojekten”, während der andere Kurs sich nur mäßig verschlechtert.

Inwiefern die teilweise Verschlechterung der Bewertungen mit Ermüdungserscheinungen

seitens der Studierenden während Corona zusammenhängt bleibt offen. Zwar artikulieren einige der Evaluierenden, dass die Beteiligung in der Online-Lehre bzw. die Studienmotivation zunehmend schwerfällt, jedoch lässt sich dies anhand der vorliegenden Datenbasis nicht zweifelsfrei klären.

Auch wenn die konzipierten Kurse seitens der Studierenden generell gut angenommen werden, zeigen die Bereiche “Interessantheit des Kurses” und Relevanz ein deutliches Verbesserungspotential. Das relativ schlechte Abschneiden der Kurse in diesen Bereichen lässt aufhorchen, da diese in direkter Verbindung mit den in Abschnitt 2 geschilderten Problemen während der Digitalen Transformation der ÖV stehen: Nutzerzentriertheit und die Zusammenarbeit mit der IT wird noch nicht als originäre Aufgabe der angehenden Verwaltungsmitarbeiter*innen gesehen.

Der extreme Einbruch bezüglich der Relevanzeinschätzung und der Interessantheit des Kurses “Projekt- und Servicemanagement von Digitalisierungsprojekten” vom Winter- hin zum Sommersemester lässt sich so jedoch nicht vollständig begründen. Vielmehr steht diese Bewertung im Widerspruch zur Gesamtbewertung des Kurses und den steigenden Teilnehmer*innenzahlen. Auch eine Analyse der Freitextfragen bezüglich eventueller Verbesserungsmöglichkeiten ergibt keine Erklärung, zumal die verbalen Rückmeldungen nur den konkreten Termin der LV bemängeln und die geringe Beteiligung der Studierenden in der virtuellen Lehre beklagen. Der Einbruch bei den Bewertungen spiegelt sich auch nicht in den durch die Studierenden erstellten Kurzreflexionen (siehe Abschnitt 3.1) wider, in welchen das Kursangebot vorwiegend – bis auf eine explizit Ausnahme – als bereichernd und gut in das sonstige Curriculum integriert dargestellt wird.

Ob die Bewertung mit dem Semester zusammenhängt, in welchem die Evaluierenden studieren, lässt sich aufgrund der mangelnden Datenerhebung seitens der Hochschulevaluation nicht ergründen.

Tab. 2: Gemittelte Bewertung der Kurse im WiSe 20/21 und SoSe 21

DigiNutzer: “Digitalisierungsprozesse nutzerorientiert gestalten”

ProjSer: “Projekt- und Servicemanagement von Digitalisierungsprojekten”

<i>Bewertungsaspekt</i>	DigiNutzer		ProjSer	
	<i>WiSe 20/21</i>	<i>SoSe 21</i>	<i>WiSe 20/21</i>	<i>SoSe 21</i>
Gesamtbeurteilung	5,6	5,5	5,7	5,4
Struktur und Didaktik	5,5	5,5	5,6	5,4
Lernmethoden und Lernmaterial	5,5	5,4	5,6	5,3
Interessantheit des Kurses	4,9	4,2	5,1	3,6
Relevanz bzgl. d. berufl. Handlungsfähigkeit	4,2	3,9	5,4	3,7

5 Fazit

Der vorliegende Beitrag benennt Anknüpfungspunkte für die Methodenlehre des Software und Usability Engineerings in der nicht-informatischen Lehre am Beispiel der Ausbildung für die ÖV, welche seit drei Semestern angeboten wird. Die in jedem Semester begleitend stattfindende Evaluierung zeigt eine hohe Lernbereitschaft der Studierenden in diesem Bereich, da ihr Alltag bereits vollständig von Digitalisierungslösungen durchdrungen ist und ein entsprechend hoher Leidensdruck bei bereits in der ÖV tätigen Studierenden vorliegt. Außerdem zeigt sich, dass sich auch der ÖV fachfremde Methoden und Prinzipien der Informatik Studierenden ohne explizit technische Vorbildung zielgruppengerecht, strukturiert und verständlich vermitteln lassen.

Die Evaluierung zeigt jedoch auch, dass manche Studierende es als Herausforderung empfinden, sich mit in der Informatik üblichen Methoden und Denkmustern zu konfrontieren bzw. deren Relevanz für ihr künftiges Berufsfeld noch nicht vollständig einschätzen können.

Ob der Studienfortschritt der Studierenden einen Einfluss auf diese Relevanzbewertung hat, muss in zukünftigen Befragungen in Ergänzung zur Evaluierung seitens der Hochschule untersucht werden.

Die aktuell laufende, dritte Iteration der Kurse adressiert die genannten Defizite indem mehr Bezüge zu konkreten Verwaltungstätigkeiten aufgegriffen werden. Exemplarisch zu nennen sind hier die Integration weiterer Beispiele digitalisierter Verwaltungsdienstleistungen, die Diskussion aktueller Stellenprofile sowie die Präsentation einschlägiger Konferenzbeiträge in Videoform, um den Studierenden ein möglichst ganzheitliches Bild auf die Digitale Transformation der Verwaltung bieten zu können.

Offen bleibt jedoch die Frage, inwiefern sich die graduelle Verschlechterung der Lehrbeurteilung auf Corona zurückführen lässt. Die Evaluierungsdaten der dritten Iteration liegen zum Zeitpunkt der Erstellung dieses Artikels noch nicht vor. Eine vollständige Beantwortung dieser Frage erscheint jedoch in jedem Fall erst nach einigen Iterationen im Präsenzunterricht nach Corona möglich, wenn Vergleichsdaten vorliegen.

Inwiefern die Hypothese, die Digitalisierung der ÖV durch den gezielten Ausbau von Schnittstellenkompetenzen hin zur Software-Entwicklung zu beschleunigen, zutrifft, kann zum aktuellen Zeitpunkt nicht gesagt werden. Hierzu sind weitere Studien notwendig, nachdem eine signifikante Anzahl an Studierenden ihr Studium absolviert hat. In jedem Fall wird deutlich, dass die Studierenden noch weiter für die Schnittstellenthematiken hin zur IT sensibilisiert werden müssen, um deren Relevanz für ihre zukünftigen Arbeitsfelder besser einschätzen zu lernen.

In jedem Fall lassen die bisherigen Ergebnisse darauf hoffen, dass es möglich ist, die Digitalisierungsbemühungen in der ÖV “bottom-up” bereits im Rahmen der Ausbildung durch gezielten Kompetenzaufbau zu unterstützen. Dieser Ansatz ist wichtig, da Indikatoren dafür vorliegen, dass sich öffentliche Digitalisierungsprojekte – gerade im Innovationsbereich –

kaum “top-down” [Er20] oder einem Mindestmaß an “*professional*” *digital literacy* des gesamten Projektteams erfolgreich durchführen lassen.

Literatur

- [BG01] Beck, K.; Grenning, J.; et al., 2001, URL: <https://agilemanifesto.org>, Stand: 11. 10. 2021.
- [Bu21] 2021, URL: <https://bund.dev>, Stand: 14. 10. 2021.
- [Er20] Erdmann, P.: Digitale Lotsen in der öffentlichen Verwaltung. OPAS-Plattform 002012/1, S. 161–165, 2020, URL: <https://www.polver.uni-konstanz.de/mergel/>.
- [GL87] Gould, D. J.; Lewis, C.: Designing for Usability: Key Principles and What Designers Think: Human-computer interaction. In (Baecker, M. R., Hrsg.). Morgan Kaufmann, San Francisco, CA, USA, S. 528–539, 1987.
- [Ha15] Hanson, C., 2015, URL: <https://www.codyh.com/writing/software.html>, Stand: 11. 10. 2021.
- [HU17] Fachspez. SPO Bachelorstudium Bibliotheks- u. Informationswissenschaft, Amtliches Mitteilungsblatt - Humboldt-Universität zu Berlin, Phil. Fak., Nr. 42/2017, 2017.
- [IT18] IT-Planungsrat: Beschluss 2018/22: Digitalisierung von Verwaltungsdienstleistungen, 2018, URL: <https://www.it-planungsrat.de/beschluss/beschluss-2018-22>, Stand: 28. 12. 2021.
- [MBH21] Mergel, I.; Brahimi, A.; Hecht, S.: Agile Kompetenzen für die Digitalisierung der Verwaltung. Innovative Verwaltung 2021/10, S. 28–31, 2021.
- [MGW21] Mergel, I.; Ganapati, S.; Whitford, A. B.: Agile: A New Way of Governing. Public Administration Review 81/1, S. 161–165, 2021.
- [Sch21] Schneider, K.: Interview: Was braucht innovative Verwaltung?, 2021, URL: <https://background.tagesspiegel.de/digitalisierung/interview-was-braucht-innovative-verwaltung>, Stand: 11. 10. 2021.
- [Su21] Surfshark, 2021, URL: <https://surfshark.com/dq12021?country=DE>, Stand: 14. 10. 2021.
- [Ze21] Zellhöfer, D.: Agilität und Weltkulturerbe: Erfahrungen mit sieben Jahren agilen Methoden an der Staatsbibliothek zu Berlin – Eine Fallstudie I/II. ABI Technik 41/3, S. 194–201, 2021.