Commentarii
informaticae didacticae | 12

Nadine Bergner | René Röpke | Ulrik Schroeder |
Detlef Krömker (Hrsg.)

# Hochschuldidaktik der Informatik HDI 2018

8. Fachtagung des GI-Fachbereichs Informatik und Ausbildung/Didaktik der Informatik

12.–13. September 2018 an der Goethe-Universität Frankfurt am Main

Universitätsverlag Potsdam

Commentarii informaticae didacticae (CID)

Commentarii informaticae didacticae (CID) | 12

Nadine Bergner | René Röpke | Ulrik Schroeder |
Detlef Krömker (Hrsg.)

# Hochschuldidaktik der Informatik
# HDI 2018

8. Fachtagung des GI-Fachbereichs Informatik
und Ausbildung/Didaktik der Informatik
12.–13. September 2018 an der Goethe-Universität
Frankfurt am Main

Universitätsverlag Potsdam

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

**Universitätsverlag Potsdam 2018**
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe Commentarii informaticae didacticae (CID)
wird herausgegeben von:
Johannes Magenheim, Universität Paderborn
Sigrid Schubert, Universität Siegen
Andreas Schwill, Universität Potsdam

ISSN (print) 1868-0844
ISSN (online) 2191-1940


Druck: docupoint GmbH Magdeburg
Satz: text plus form, Dresden

ISBN 978-3-86956-435-7

Zugleich online veröffentlicht auf dem Publikationsserver der
Universität Potsdam:
URN urn:nbn:de:kobv:517-opus4-413542
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-413542

# Vorwort

Die 8. Fachtagung für Hochschuldidaktik der Informatik (HDI) fand im September 2018 zusammen mit der Deutschen E-Learning Fachtagung Informatik (DeLFI) unter dem gemeinsamen Motto **„Digitalisierungs**wahn**sinn? – Wege der Bildungstransformationen“** in Frankfurt statt. Dabei widmet sich die HDI allen Fragen der informatischen Bildung im Hochschulbereich. Schwerpunkte bildeten in diesem Jahr u. a.:

- Analyse der Inhalte und anzustrebenden Kompetenzen in Informatikveranstaltungen
- Programmieren lernen und Einstieg in Softwareentwicklung
- Spezialthemen: Data Science, Theoretische Informatik und Wissenschaftliches Arbeiten

Der erste Themenblock beschäftigt sich mit der Analyse der grundlegenden Inhalte und Kompetenzen der Informatik, auch für Studierende anderer Fachrichtungen. So stellen Seegerer und Romeike vor, was jeder ausgehend der Inhalte von Hochschulkursen für andere Fachrichtungen über Informatik lernen sollte. Tillmann, Krömker, Horn und Gattinger präsentieren in ihrem englischsprachigen Beitrag, wie die Kompetenzen von Studienanfängerinnen und -anfängern analysiert und hervorgesagt werden können. Weicker beschäftigt sich mit dem Thema der, aufgrund stetiger Entwicklungen im Bereich der Informatik, sehr wichtigen Vorlesungs-Pflege.

Im Fokus des zweiten Themenblocks stehen die Kernkompetenzen der Softwareentwicklung sowie die Besonderheiten des Programmierenlernens. Schmitz und Moldt präsentieren in ihrem Beitrag ein Lehr- und Lernkonzept für die Softwareentwicklung im Team. Auch Röpke, Larisch und Schroeder widmen sich der Kollaboration in der Softwareentwicklung und thematisieren die Förderung überfachlicher Kompetenzen. Keverpütz und Küppers betrachten in ihrem Beitrag die Möglichkeiten des E-Assessments in einer dualen IT-Ausbildung, um die Lehr-Lern-Prozesse zu verbessern. Der Beitrag von Striewe und Kramer konzentriert sich auf den formaleren Aspekt der Beherrschung der Syntax einer Programmiersprache.

Im dritten Themenblock werden drei Themengebiete der Informatik spezifisch beleuchtet. Im Beitrag von Grillenberger und Romeike wird das aktuelle Trendthema Data Science durch eine qualitative Analyse von Modulhandbüchern etablierter Studiengänge charakterisiert und ein Data-Lite-

racy-Kompetenzmodell vorgeschlagen. Frede und Knobelsdorf präsentieren in ihrem Beitrag eine Datenanalyse zur Erhebung der Studierendenperformance in einem Einführungskurs der theoretischen Informatik. Greven und Schroeder analysieren anhand empirischer Erhebungen Probleme beim Erlernen des wissenschaftlichen Arbeitens.

Unser besonderer Dank gilt dem Programmkomitee und den hier nicht genannten Helferinnen und Helfern für ihren Einsatz bei der Vorbereitung und Durchführung der Tagung sowie allen Autorinnen und Autoren und den aktiv Beitragenden.

Aachen (und Frankfurt), im Juli 2018
Nadine Bergner, René Röpke, Ulrik Schroeder sowie Detlef Krömker

# Programmkommitee

- Nadine Bergner, RWTH Aachen
- Torsten Brinda, Universität Duisburg Essen
- Jörg Desel, FernUniversität Hagen
- Ira Diethelm, Universität Oldenburg
- Peter Forbrig, Universität Rostock
- Jörg Haake, FernUniversität Hagen
- Reinhard Keil, Universität Paderborn
- Andrea Kienle, FH Dortmund
- Maria Knobelsdorf, Uni Hamburg
- Detlef Krömker, Universität Frankfurt (2. Vorsitz)
- Johannes Magenheim, Universität Paderborn
- Ralf Romeike, Universität Erlangen
- Alexander Schmolitzky, HAW Hamburg
- Ulrik Schroeder, RWTH Aachen (Vorsitz)
- Carsten Schulte, Universität Paderborn
- Andreas Schwill, Universität Potsdam
- Karsten Weicker, HTWK Leipzig
- Olaf Zukunft, HAW Hamburg
- Albert Zündorf, Universität Kassel

# Inhaltsverzeichnis

## Inhalte und Kompetenzen der Informatik

Was jeder über Informatik lernen sollte – Eine Analyse von Hochschulkursen für Studierende anderer Fachrichtungen ..... 13
*Stefan Seegerer und Ralf Romeike*

Analysing & Predicting Students Performance in an Introductory Computer Science Course ..... 29
*Alexander Tillmann, Detlef Krömker, Florian Horn und Thorsten Gattinger*

Vorlesungs-Pflege: Maßnahmen zur Runderneuerung degenerierender Informatikvorlesungen ..... 47
*Karsten Weicker*

## Programmieren lernen & Softwareentwicklung

Ein Lehr- und Lernkonzept für die Softwareentwicklung im Team ..... 63
*Dennis Schmitz und Daniel Moldt*

Förderung überfachlicher Kompetenzen in praktischen Software-Engineering-Veranstaltungen der RWTH Aachen ..... 79
*René Röpke, Kathrin Larisch und Ulrik Schroeder*

Konsistente Lehr-Lern-Prozesse in der dualen IT-Ausbildung ..... 91
*Claudia Keverpütz und Bastian Küppers*

Empirische Untersuchungen von Lückentext-Items zur Beherrschung der Syntax einer Programmiersprache ..... 101
*Michael Striewe und Matthias Kramer*

## Spezielle Themen des Informatikstudiums Data Science, Theoretische Informatik und Wissenschaftliches Arbeiten

Was ist Data Science? Ermittlung der informatischen Inhalte durch Analyse von Studienangeboten ..... 119
*Andreas Grillenberger und Ralf Romeike*

Explorative Datenanalyse der Studierendenperformance in der Theoretischen Informatik ..... 135
*Christiane Frede, Maria Knobelsdorf*

Wissenschaftliches Arbeiten lernen – Eine Problemanalyse ..... 151
*Christoph Greven und Ulrik Schroeder*

# Inhalte und Kompetenzen der Informatik

# Was jeder über Informatik lernen sollte – Eine Analyse von Hochschulkursen für Studierende anderer Fachrichtungen

**Stefan Seegerer und Ralf Romeike**
Friedrich-Alexander-Universität
Didaktik der Informatik
Martensstr. 3
91058 Erlangen
stefan.seegerer@fau.de
ralf.romeike@fau.de

**Abstract:** Um für ein Leben in der digitalen Gesellschaft vorbereitet zu sein, braucht jeder heute in verschiedenen Situationen umfangreiche informatische Grundlagen. Die Bedeutung von Informatik nimmt nicht nur in immer mehr Bereichen unseres täglichen Lebens zu, sondern auch in immer mehr Ausbildungsrichtungen. Um junge Menschen auf ihr zukünftiges Leben und/oder ihre zukünftige berufliche Tätigkeit vorzubereiten, bieten verschiedene Hochschulen Informatikmodule für Studierende anderer Fachrichtungen an. Die Materialien jener Kurse bilden einen umfangreichen Datenpool, um die für Studierende anderer Fächer bedeutenden Aspekte der Informatik mithilfe eines empirischen Ansatzes zu identifizieren. Im Folgenden werden 70 Module zu informatischer Bildung für Studierende anderer Fachrichtungen analysiert. Die Materialien – Publikationen, Syllabi und Stundentafeln – werden zunächst mit einer qualitativen Inhaltsanalyse nach Mayring untersucht und anschließend quantitativ ausgewertet. Basierend auf der Analyse werden Ziele, zentrale Themen und Typen eingesetzter Werkzeuge identifiziert.

**Keywords:** Informatik für alle, Hochschulkurse, Andere Fachrichtungen, Inhaltsanalyse.

# 1 Einleitung

Um für ein Leben in der digitalen Gesellschaft vorbereitet zu sein, braucht *jeder* heute in verschiedenen Situationen umfangreiche informatische Grundlagen. In dieser Hinsicht wird informatische Bildung als wichtiger Bestandteil der Allgemeinbildung betrachtet, der bisher vor allem im Schulkontext diskutiert und ausdifferenziert (z. B. [Ge16]) wurde. Auch an Universitäten steigt das Interesse an informatischer Bildung für Studierende aller Fachrichtungen. Mit steigender Nachfrage kommen unweigerlich Fragen über die Gestaltung entsprechender Kursangebote auf (vgl. [Ba10]). Ein „Informatikcurriculum für alle" müsste eine Basis enthalten, mit der jeder Studierende vertraut sein sollte. Bei der Gestaltung eines solchen Curriculums stellt sich u. a. die Frage, welche Ziele mit einem solchen Bildungsangebot verfolgt werden sollen und welche Themen dabei zentral sind. Diese Frage ist eng verknüpft damit, was *jeder* gebildete Mensch über Informatik wissen sollte. Dies kann zum einen normativ durch Experten festgelegt und argumentativ untermauert werden, andererseits aber auch durch eine empirische Erhebung der Aspekte, die bereits existierende Angebote als wichtig erachten, ermittelt werden.

Viele Hochschulen bieten Informatikveranstaltungen für Studierende anderer Fachrichtungen an. Die Materialien jener Kurse bilden einen umfangreichen Datenpool für einen entsprechenden empirischen Ansatz, der die für Studierende anderer Fächer bedeutenden Aspekte der Informatik identifizieren soll. Aus diesem Grund analysieren wir eine Auswahl solcher Kurse, die zwischen 2001 und 2018 angeboten wurden. Diese Datenbasis erlaubt es, die Gemeinsamkeiten verschiedener Ansätze informatischer Bildung zu erfassen. Im Mittelpunkt der Untersuchung stehen formulierte Ziele, thematisierte Inhalte und die Typen verwendeter Programmierwerkzeuge. Abschnitt 2 beschreibt zunächst Hintergründe und Argumentationslinien im Kontext „Informatik für alle". Daran anknüpfend werden in den Kapiteln 3 und 4 Forschungsfragen, Methodik und die zugrundeliegende Datenbasis dargestellt. In den Kapiteln 5 und 6 werden die Ergebnisse der Untersuchung präsentiert und diskutiert.

# 2 Hintergrund

Immer mehr Akteure betrachten Informatikkompetenz als zentralen Bestandteil der Allgemeinbildung. Es sind nicht mehr nur Vertreter des Fachs, die dessen Wichtigkeit betonen, sondern auch das Wirtschaftsministerium [Bu16] oder Fachverbände anderer Disziplinen wie der Chemie [BV18] sprechen sich für „Informatik für alle“ aus.

In öffentlichen Diskussionen über die Bedeutung informatischer Allgemeinbildung wird häufig auf die Anforderungen zukünftiger Arbeit verwiesen. Aber im Zusammenhang, warum Informatik für alle wichtig ist, werden noch weitere Argumente aufgeführt. Vogel et al. [VSC17] oder Döbeli Honegger [Dö16] stellen verschiedene Argumente für informatische Bildung heraus. Döbeli Honegger nennt dabei beispielsweise das Konzeptwissenargument, mit Informatik ließen sich digitale Werkzeuge besser nutzen, oder das Welterklärungs- bzw. Mündigkeitsargument, Informatik helfe die technische Welt zu verstehen bzw. mitzugestalten. Häufig wird auch problemlösendes Denken oder Computational Thinking als Argument für informatische Bildung genannt. Letzteres ist ein Begriff der, von Wing [Wi06] popularisiert, die Art und Weise charakterisiert, wie ein Informatiker Probleme löst. Ziel informatischer Bildung in diesem Kontext ist es also vor allem Denkweisen und Problemlösestrategien zu vermitteln. Dazu zählen das Zerlegen von Problemen in kleinere Teilprobleme (Dekomposition) oder das Weglassen von Details (Abstraktion). Solche informatischen Problemlösefähigkeiten bzw. Denkweisen werden an vielen Stellen als fundamental für die Allgemeinbildung betrachtet (z. B. [Di11]). Der Einfluss Wings auf Universitäts- und Collegekurse zeigt sich beispielsweise im Framework *Advanced Placement Computer Science Principles,* das, erstmals in 2010 veröffentlicht, die Inhalte und Ziele von Informatikkursen auf College Niveau an Highschools umreißt. Das Framework besteht aus sieben großen Ideen der Informatik und sechs Computational-Thinking-Praktiken.

Die den Argumenten zugrundeliegenden Sichtweisen können Zieldimensionen informatischer Allgemeinbildung an Hochschulen darstellen. Sie geben allerdings nur bedingt Aufschluss über mögliche Themen, die in solchen Kursen behandelt werden sollten. Mit den „Great Principles of Computing“ [De03], den „Fundamentalen Ideen der Informatik“ [Sc93] oder auch den „Big Ideas of K-12 Computing“ [BTY18] existieren Kataloge, die wichtige und zeitlose Aspekte der Informatik erfassen. Während Denning und Schwills Ansätze vor allem aus dem Fach heraus motiviert sind, bezieht sich der Ansatz von Bell konkret auf Schulbildung und nennt zehn wesentlichen Ideen,

die hinter Informatik in der Schule stehen und damit auch wichtige Themen für die Schulinformatik darstellen. Die Themen des Schulunterrichts wurden von Hubwieser et al. bereits mithilfe eines empirischen Ansatzes untersucht [Hu15]. Sie verglichen 2015 den Informatikunterricht in zwölf verschiedenen Ländern unter anderem hinsichtlich der Ziele, Inhalte sowie verwendeten Programmiersprachen und -werkzeuge. Ein zentrales Ergebnis der Studie waren 19 Kategorien, die die Inhalte der Schulcurricula auf Basis von acht der untersuchten Länder abbilden. Diese Studie wurde auch zur Orientierung für die hier erfolgende Analyse herangezogen.

Auch Kurse für Studierende anderer Fachrichtungen beschäftigen sich häufig mit Programmierung bzw. deren Grundlagen. Besonders in der Informatik spielen daher Werkzeuge eine bedeutende Rolle beim Lernen. Die in der Studie von Hubwieser et al. identifizierten Programmiersprachen und -werkzeuge wurden in die Kategorien didaktische Umgebung mit eigener Programmiersprache, didaktische Umgebung basierend auf einer anderen Programmiersprache und professionell genutzte Sprachen eingeordnet. Eine andere Herangehensweise Programmierwerkzeuge zu klassifizieren, stellt der Ansatz von Kelleher und Pausch dar [KP05]. Deren Kategorisierung von Programmierumgebungen für Einsteiger unterscheidet Teaching und Empowering Systems und kann im Rahmen von Bildungsangeboten dazu verwendet werden, mögliche Intentionen, die mit genutzten Werkzeugen einhergehen, zu erfassen.

## 3 Forschungsfragen

Die Konzeption von Informatikkursen für Studierende anderer Fachrichtungen erfordert eine bewusste Auswahl der zu erreichenden Ziele, zu behandelnden Themen oder zu verwendenden Werkzeugen. Bisher besteht keine Einigkeit über eine ausdifferenzierte Basis mit der jeder vertraut sein sollte. Daher soll in einem ersten Schritt ein empirischer Ansatz helfen, die Grundlagen der Informatik für alle Studierenden zu bestimmen. Diese sollten sich in entsprechenden Hochschulkursen für diese Zielgruppe widerspiegeln. Eine Analyse dieser Kurse kann demnach helfen, wesentliche Aspekte zu identifizieren. Dabei können im Kurs behandelte Themen nicht losgelöst betrachtet werden, da diese möglicherweise direkt von den angestrebten Zielen beeinflusst werden. Gleichzeitig beziehen sich informatische Bildungsangebote oft stark auf Programmierwerkzeuge bzw. -sprachen. Im weiteren Verlauf werden daher folgende Fragestellungen untersucht:

- Welche Ziele setzen sich Informatikkurse für Studierende anderer Fachrichtungen?
- Welche Themen der Informatik sind in Informatikkursen für Studierende anderer Fachrichtungen zentral?
- Welche Typen von Programmiersystemen werden in Informatikkursen für Studierende anderer Fachrichtungen eingesetzt?

## 4 Vorgehen

Das Vorgehen orientiert sich zunächst an den Schritten der qualitativen Inhaltsanalyse nach Mayring [Ma00]. Ziel dieser Analyse ist es, die zentralen Aspekte des untersuchten Materials kontextabhängig vollständig abzubilden. Zentral ist dabei die Auswahl einer geeigneten Datenbasis, die möglichst repräsentativ sein sollte. In dieser Untersuchung wurden daher 70 nationale wie internationale Kurse mit verschiedenen Ansätzen und unterschiedlichen Zielgruppen berücksichtigt. Grundlegend für die Ansätze Schwills und Dennings ist die Annahme, dass informatische Bildung sich auf zeitbeständige Aspekte fokussieren sollte. Daher wurden Dokumente von 2001 bis 2018 berücksichtigt. Kurse, die ausschließlich das Erlernen einer bestimmten Programmiersprache oder den Umgang mit bestimmten Anwendungen zum Ziel hatten, ohne sich auf die zugrundeliegenden Konzepte zu beziehen, wurden nicht in die Datenbasis übernommen. Kriterien waren außerdem eine Berücksichtigung in Studienplänen für Studierende anderer Fachrichtungen und ausreichend verfügbare Informationen in Form konkret benannter Ziele und Themen. Die Datenbasis umfasst sowohl Kurse, die rein für bestimmte Studierendengruppen, wie angehende Lehrerinnen und Lehrer, Historikerinnen und Historiker oder Biologinnen und Biologen konzipiert sind, als auch solche für alle Studierenden der jeweiligen Bildungseinrichtung.

Eine anfängliche Auswahl erfolgte mithilfe verfügbarer Publikationen in der ACM Digital Library. Hier wurde eine Schlagwortsuche mit „computing education“, „computer science education“, „computer science for non-majors“, „computing curriculum“ bzw. „CS0“ durchgeführt und jeweils die ersten 200 Treffer begutachtet. Innerhalb Deutschlands wurden die Vorlesungsverzeichnisse aller staatlichen Universitäten hinsichtlich der Begriffe „Informatik“, „Computer“ und „digital“ durchsucht. Für die endgültige Auswahl wurden die entsprechenden Kurse anschließend auf Relevanz nach o. g. Kriterien hin untersucht. Ergänzt wurde dies durch eine ausführliche Internetrecherche. Diese lieferte neben zusätzlichen Informationen zu den Kursen,

wie Wochenplänen und Skripten, auch Material von weiteren Veranstaltungen. Standen zu einem Kurs mehrere Quellen zur Verfügung, so wurde jeweils eine Auswahl getroffen, die insbesondere auf Detailgrad und Aktualität der Dokumente basierte. Eine Übersicht über Universitätsstandorte der untersuchten Kurse findet sich in Abbildung 1.

Der nächste Schritt ist die Entwicklung eines Kategoriensystems. Ein solches kann entweder deduktiv aus bestehender Theorie oder induktiv aus der Datengrundlage abgeleitet werden. Bei den Untersuchungsaspekten *Ziele und Inhalte* wurde eine induktive Kategorienbildung vorgenommen. Damit reduziert sich das Risiko, wichtige Aspekte aufgrund vorher festgelegter Kategorien nicht zu berücksichtigen. Wie die Klassifizierung von didaktischen Programmiersystemen von Kelleher und Pausch [KP05] zeigt, können Programmierwerkzeuge auch bestimmte Ziele unterschiedlich gut unterstützen. Daher wurde für den dritten Aspekt (Programmiersprachen und -werkzeuge) auf eine deduktive Entwicklung der Kategorien auf Basis dieses Ansatzes zurückgegriffen, es wurden jedoch induktive Ergänzungen zugelassen.

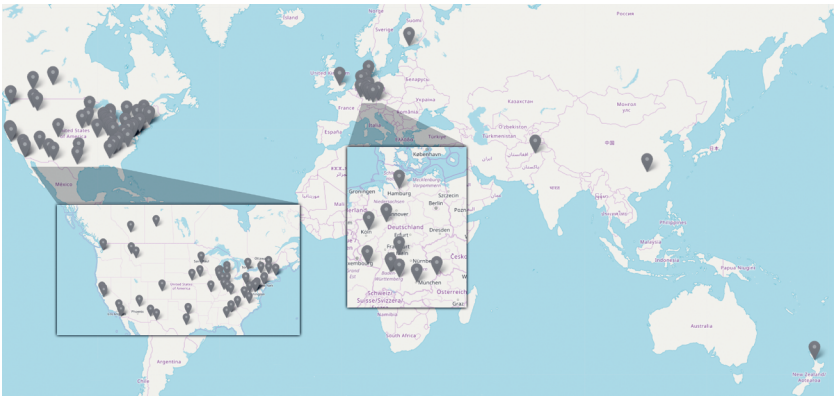

Abb. 1: Übersicht über die untersuchten Kurse für Studierende anderer Fachrichtungen

Die Darstellung der Ziele, Inhalte und Themen innerhalb der Kurse variierte teils deutlich. Während einige Veranstaltungskalender oder Syllabi sich auf eine Stichwortliste ihrer Inhalte beschränkten, enthielten andere weitaus detailliertere Informationen. Daher konnte eine zu kodierende Textpassage (Coding) sowohl aus einem Einzelwort, wie dem Titel einer Vorlesung, als auch aus einem ganzen Satz, beispielsweise einem Lernziel eines Kurses, be-

stehen. Insbesondere aus einem einzelnen Wort bestehende Codings wurden stets im Kontext interpretiert: Bei einer Verwendung des Wortes in anderem Sinnzusammenhang oder zur Abgrenzung von anderen Begriffen erfolgte keine Kodierung. Soweit verfügbar wurde zusätzliches Material zur Explikation herangezogen.

Nach Auswahl des Materials sowie Festlegung der Kategoriensysteme und Kodierungskriterien wurde die eigentliche Analyse softwaregestützt mit der Analysesoftware MaxQDA durchgeführt. Ausgangspunkt war ein Set bestehend aus zehn Kursen, sukzessive erweitert um je drei bis fünf weitere Kurse. Dabei wurde für jedes neu auftretende Coding überprüft, ob dieses in eine bereits bestehende Kategorie eingeordnet werden kann oder, wenn nicht, eine neue Kategorie angelegt. Sollte eine neue Kodierung hingegen in mehr als eine Kategorie passen, so waren die betroffenen Kategorien zu speziell und wurden zu einer neuen Kategorie mit entsprechender Benennung zusammengefasst. Nach jeder dieser Erweiterungen wurde auch das zuvor bereits analysierte Material neu bewertet und gegebenenfalls bereits existierende Codings neu eingeordnet oder Kategorien vereinigt.

Wegen des unterschiedlichen Detailgrads in der Darstellung durch die verschiedenen Autoren ist eine Aussage aufgrund der Anzahl an Kodierungen pro Dokument und deren jeweiliger Länge nicht valide möglich. Eine Vielzahl an Kodierungen eines Aspekts lässt nicht zwangsläufig auf eine höhere Bedeutung dieses Aspekts schließen. Zur Darstellung und Interpretation der Ergebnisse wurden daher die relativen Häufigkeiten ermittelt. Dabei wurde das Auftreten einer Kategorie pro Dokumentgruppe als Wahrheitswert kodiert. Da vor allem die Bedeutung einzelner Themen relativ zu anderen von Interesse war, ist eine solche Quantifizierung der Beantwortung der Fragestellungen zuträglich. Ähnliche Ansätze wurden bereits mehrfach zur Analyse von Curricula angewendet (bspw. bei [BKM15], [GR14]).

# 5 Ergebnisse

## 5.1 Verfolgte Ziele

Die Kursverantwortlichen verfolgen mit ihren Angeboten bestimmte Ziele. Während einige Kurse vor allem ein einzelnes Ziel in den Vordergrund stellten, verfolgten andere eine Kombination mehrerer Ansätze. Die inkrementelle Kategorisierung der intendierten Ziele führte zu vier verschiedenen Typen. Ein Kurs verfolgte dabei im Schnitt Ziele aus 1,87 dieser Kategorien (Modalwert 2).

**(G1) Denkweisen.** In diese Kategorie fallen Kurse, die das Vermitteln bestimmter Denkweisen als ein primäres Ziel herausstellen. Dazu gehören das Vermitteln von Computational Thinking, algorithmischem, kreativem oder auch problemlösendem Denken.

**(G2) Fluency.** Durch informatische Bildung wird ein tieferes Verständnis der verwendeten Technologien vermittelt. Studierende sollen befähigt werden, Informatiksysteme effizient und gewinnbringend zur Lösung von Problemen einzusetzen.

**(G3) Wissenschaft.** Kurse sollen einen Überblick über das Fachgebiet Informatik geben. Den Studierenden werden zentrale Ideen und Schlüsselkonzepte der Wissenschaft Informatik aufgezeigt. Es geht häufig auch darum, ein (breites) Bild der Disziplin zu vermitteln und gleichzeitig das Verständnis für dessen grundlegende Konzepte zu schaffen.

**(G4) Gesellschaft.** Studierende sollen den Einfluss und die Auswirkungen von Informatik und Informatiksystemen auf die Gesellschaft und auf ihr persönliches zukünftiges Leben verstehen. Es gilt ausreichend Informatikwissen zu erwerben, um beispielsweise Auswirkungen von Informatiksystemen diskutieren zu können.

Bezogen auf die Datenbasis zeichnet sich ein Trend zu den Zielkategorien (G3) Wissenschaft und (G1) Denkweisen ab, die insgesamt in 41 (59 %) bzw. 38 Kursen (54 %) vorkamen. (G2) Fluency wurde in 29 (41 %), (G4) Gesellschaft zumindest in 23 Kursen (33 %) kodiert. Unter den Publikationsquellen dominiert die Kategorie (G1) Denkweisen mit 75 %. Obwohl die Kategorie bereits zu Beginn der betrachteten Zeitspanne kodiert werden konnte, ist insbesondere nach Popularisierung des Begriffs „Computational Thinking“ ein deutlicher Anstieg zu verzeichnen.

Zusätzlich zu den Intentionen der Kurse fanden sich in den Dokumenten auch Begründungen, warum jeder Studierende informatische Bildung erwerben sollte. Diese Begründungen überschneiden sich mit den von Döbeli Honegger genannten Argumenten [Dö16], erweitern diese jedoch auch teilweise. In Bezug auf die ersten beiden Kategorien werden insbesondere die Notwendigkeit der Vorbereitung auf zukünftige berufliche Tätigkeiten, die Aussicht auf höher bezahlte Jobs und eine steigende Problemlösekompetenz (*Problemlöseargument*) angeführt. Die Denkprozesse, die beim Lösen informatischer Probleme involviert sind, helfen auch, Probleme in anderen Fach-

bereichen zu bewältigen oder dort neue Impulse zu liefern. In Zusammenhang mit der Kategorie (G1) Denkweisen wird oft auch das *Wissenschaftsargument* – Informatikkompetenz hilft neue wissenschaftliche Erkenntnisse zu gewinnen – betont. Dass informatische Bildung eine effiziente Nutzung von Informatiksystemen ermöglicht, basiert auf dem *Konzeptwissenargument*. Um die Möglichkeiten von Informatiksystemen voll auszuschöpfen, reichen Anwenderkompetenzen nicht aus, sondern es werden auch Informatikkompetenzen benötigt. Vorwiegend zeigt es sich bei (G2) Fluency. Kernprinzipien oder ein breites Bild der Wissenschaft zu vermitteln (G3), geht gerade im amerikanischen Raum teilweise mit dem Wunsch einher, Studierende für ein Nebenfach oder für eine Vertiefung in Informatik zu gewinnen (*Berufswahlargument*). Außerdem thematisieren diese Kurse die Grundlagen digitaler Technologien und entmystifizieren damit die digitale Welt (*Welterklärungsargument*). Kurse der Kategorie (G4) rücken sichtbare Phänomene der Informatik, ein Verständnis der technischen Welt und vor allem ihre Wechselwirkungen mit der Gesellschaft in den Vordergrund. Informatik wird in diesem Zusammenhang als Teil mündiger Staatsbürgerschaft gesehen. Es stehen vor allem das *Mündigkeitsargument* und *Denkobjektargument* – Vorstellungen über wichtige Konzepte unseres Lebens schärfen – dahinter. Hinzu kommt hier ein *Ethikargument* – die Notwendigkeit unterscheiden zu können zwischen dem, was technisch möglich und dem, was ethisch vertretbar ist.

In der „Dagstuhl-Erklärung zur Bildung in einer digitalisierten Welt" wird deutlich gemacht, dass eine gesellschaftlich-kulturelle, eine anwendungsorientierte und eine technologische Perspektive auf Phänomene, Artefakte, Systeme oder Situationen der digitalen vernetzten Welt eingenommen werden sollte [BD17]. Die hier identifizierten Zielkategorien lassen sich auf diese Perspektiven abbilden. So rücken sie bestimmte Perspektiven unterschiedlich stark in den Vordergrund: Gesellschaft („Wie wirkt das?"), Fluency („Wie nutze ich das?") oder Wissenschaft und Denkweisen („Wie funktioniert das?"). Informatikbildung stellt eine Basis für alle diese Perspektiven dar und unterstützt alle Zieldimensionen, beispielsweise um Informatiksysteme gewinnbringend zu nutzen oder gesellschaftliche Auswirkungen der Digitalisierung erst zu reflektieren.

## 5.2 Betrachtete Themen

Die analysierten Dokumente zeigen eine Vielfalt an Zugängen zu informatischer Bildung für Studierende anderer Fachrichtungen, bspw. über Geoinformationssysteme oder eine Datenzentrierung. Die Analyse im letzten Abschnitt offenbarte die verschiedenen vorhandenen Ziele, deren Fokus sich teils deutlich unterscheidet. Aus diesem Grund sollten die Themen auch im Kontext der jeweiligen Intentionen betrachtet werden. Die aggregierten Themennennungen wurden hierzu mit den identifizierten Zielen der Kurse ins Verhältnis gesetzt. Die relativen Häufigkeiten einzelner Themen bezüglich der Zielkategorien ist in Abbildung 2 dargestellt.

Trotz der unterschiedlichen Zielsetzungen lassen sich gemeinsame Themen feststellen, die in jeder Zielkategorie in über 50 % der Kurse kodiert wurden. Diese gemeinsame Basis wird aus den sieben Themenbereichen *Algorithmik, Programmierung, Repräsentation von Daten, Computerorganisation, soziale Implikationen, Datennutzung,* sowie *Netzwerke* gebildet. Damit wurde beispielsweise der Umgang – z. B. Analyse oder Visualisierung – mit (mehr oder weniger großen) Datenmengen relativ betrachtet deutlich häufiger thematisiert als traditionelle Themen wie *Datenbanken* oder *formale Sprachen.*

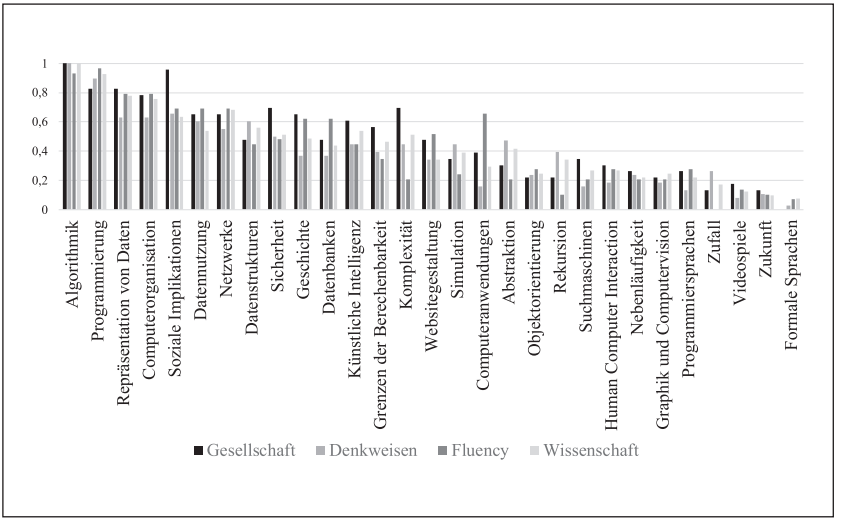


Abb. 2: Betrachtete Themenbereiche dargestellt in relativer Häufigkeit

Es zeigt sich, dass neben absolut grundlegenden Themen wie *Algorithmik, Programmierung* oder *Repräsentation von Daten,* auch eher technische Themen, wie *Computerorganisation* (Komponenten eines Rechners, die Von-Neumann-Architektur und logische Schaltungen) oder *Netzwerke* (insbesondere Aufbau und die Funktionsweise des Internets) im Rahmen eines einsemestrigen Angebots von einer Mehrheit der untersuchten Kurse als relevant eingestuft wurden.

Eine Analyse der Häufigkeiten bezogen auf einzelne Zielkategorien erlaubt weitere Einsichten und lässt bestimmte Zusammenhänge erkennen. Tabelle 1 zeigt die Themen, die neben o. g. Themen in über 50 % der Kurse einer Kategorie kodiert wurden. So enthalten Kurse aus der Kategorie (G2) Fluency eher die Themen *Anwendungssoftware, Geschichte, Datenbanken* oder *Webentwicklung.* Kurse mit gesellschaftsbezogenen Zielen betonen *Sicherheit, Komplexität, Grenzen der Berechenbarkeit* und geschichtliche Aspekte stärker, während sich bei der Kategorie (G3) Wissenschaft *Datenstrukturen, künstliche Intelligenz, Komplexität* oder *Sicherheit* hervortun. Sofern primäre Kursziele vor allem das Vermitteln von Denkweisen und Problemlösestrategien sind, ist die Anzahl weiterer (häufiger) Themen im Vergleich geringer.

Tab. 1: Zielkategoriespezifische Themen

| **(G1) Denkweisen** | **(G2) Fluency** | **(G3) Wissenschaft** | **(G4) Gesellschaft** |
|---|---|---|---|
| Datenstrukturen | Anwendungs-software | Datenstrukturen | Sicherheit |
| Sicherheit | Geschichte | Künstliche Intel-ligenz | Komplexität |
| | Datenbanken | Komplexität | Geschichte |
| | Webentwicklung | Sicherheit | Künstliche Intel-ligenz |
| | | | Grenzen der Berechenbarkeit |

Vergleicht man die Liste an Themen aus der induktiven Analyse mit bekannten Katalogen, zeigt sich, dass alle zehn „Big Ideas“ nach Bell et al. [BTY18] in den Kategorien enthalten sind. Auch die sieben Ideen der AP CS Principles

Standards finden sich bis auf Kreativität, welche nicht als eigenständiges Thema auftritt, wieder. Schulcurricula scheinen teils andere Schwerpunkte zu setzen. Im Vergleich zur Untersuchung der Schulcurricula durch Hubwieser et al. [Hu15] weisen die Ergebnisse dieser Analyse ein abweichendes Kategoriensystem auf. Nichtsdestoweniger verfügen einige Kategorien über ein Äquivalent in der jeweils anderen Untersuchung. Die Gegenüberstellung zeigt, dass *Anwendungssoftware, Datenstrukturen* oder *Betriebssysteme* und *Geräte* in den acht untersuchten Schulcurricula anteilig deutlich häufiger kodiert wurden als in Hochschulkursen. Auf der anderen Seite war der Stellenwert von Themen wie *künstliche Intelligenz* oder *soziale Implikationen* im hier untersuchten Material höher. Allerdings war die Stichprobe bei der Untersuchung der Schulcurricula auch deutlich geringer.

Unterschiede zeigen sich aber auch zu traditionellen Hochschulkursen. Viele der bei Bröker et al. [BKM15] identifizierten Themenbereiche für Informatikstudierende sind in Veranstaltungen für Studierende anderer Fachrichtungen nicht relevant. Auswirkungen auf die Gesellschaft werden hingegen in Modulen für Informatikerinnen und Informatiker deutlich seltener thematisiert. Indes hat auch das Thema *Simulationen* relativ betrachtet bei Kursen für Studierende anderer Fachrichtungen eine höhere Bedeutung.

## 5.3 Typen eingesetzter Programmiersprachen und -werkzeuge

Die Analyse zeigte auch, dass Kurse häufig eng mit ihren verwendeten Werkzeugen verzahnt sind. Während einige Kurse aber das Programmieren stark in den Fokus rückten und eine oder mehrere Programmiersprachen bzw. -werkzeuge einsetzten, verzichteten 10 % der Kurse explizit auf die Verwendung eines Programmierwerkzeugs.

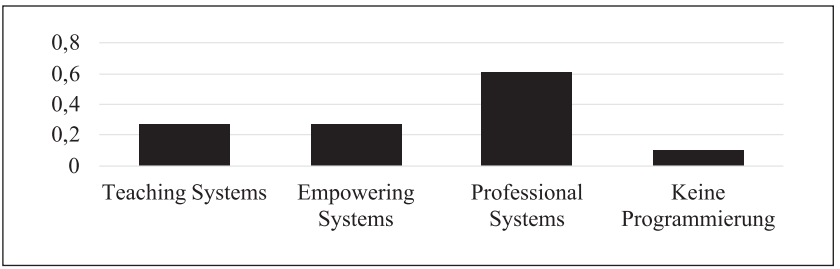


Abb. 3: Relativer Anteil verwendete Werkzeuge gruppiert nach Typ

Wie Abbildung 3 zeigt, finden professionell genutzte Werkzeuge die größte Verbreitung. In mehr als der Hälfte der Kurse werden auch – oder sogar ausschließlich – professionell genutzte Systeme eingesetzt, vor allem basierend auf Python. In den Kategorien (G1), (G2) und (G3) zeigt sich ein ähnliches Bild: Mit 65–72 % liegt ein hoher Anteil professionell genutzter Systeme bzw. Sprachen vor, während Empowering Systems in rund einem Viertel der Kurse eingesetzt werden und nur 7–8 % auf Programmiersysteme verzichten. In der Kategorie (G4) Gesellschaft hingegen setzen anteilig mehr Kurse auf Empowering Systems (35 %) bzw. verzichten auf Programmierung (17 %). Nur 43 % verwendeten hier professionell genutzte Systeme. Bei Empowering Systems ist nicht entscheidend wie gut das erlernte Wissen auf in der Praxis übliche Programmierwerkzeuge übertragen werden kann, sondern dass die Nutzer mit diesen so viel wie möglich umsetzen können [KP05]. Obwohl Empowering Systems auch im universitären Kontext eingesetzt werden, zeichnet sich dennoch eine Tendenz hin zu transferierbarem Wissen oder professionellen Systemen ab. Gründe dafür liegen möglicherweise in der Verbreitung von Sprachen wie Python beispielsweise bei wissenschaftlichen Berechnungen.

## 6 Fazit

Die Ergebnisse lassen Aussagen über informatische Hochschulveranstaltungen für Studierende anderer Fachrichtungen hinsichtlich verfolgter Ziele, betrachteter Themen und verwendeter Werkzeugtypen zu. Zentral waren dabei weitestgehend unstrittige Themen wie Algorithmik, Programmierung oder Repräsentation von Daten, aber auch eher technische Grundlagen von Computern und dem Internet. Gegenüber traditionellen Angeboten zeigten sich vor allem Unterschiede hinsichtlich der Berücksichtigung reflexiver Komponenten zu sozialen Implikationen, insbesondere zu Privatsphäre, der Rolle der Informatik in der Gesellschaft oder Urheberrecht und der Berücksichtigung von Datenanalysen und -visualisierungen. Im Hinblick auf Programmierumgebungen lässt sich erkennen, dass Hochschulkurse für Studierende anderer Fachrichtungen teils didaktische Programmierumgebungen, insbesondere auch solche, die nicht für expliziten Transfer entwickelt wurden (Empowering Systems), einsetzen. Trotzdem wurden professionell genutzte Programmiersysteme in einer Mehrheit der Kurse verwendet.

Die Ergebnisse bieten einen Einblick in die verschiedenen Themen, die im Rahmen solcher Kurse betrachtet werden könnten. Damit können Ideen generiert, aber auch eine Einordnung bestehender Kurse getroffen werden.

Ein stärkerer Fokus auf Themen wie Datennutzung und Simulationen zeigt eben auch, dass ein Informatikkurs, der für Studierende anderer Fächer der einzige Kontakt zum Fach ist, neue Themen in Erwägung ziehen sollte, die traditionelle Veranstaltungen auslassen. Soll beispielsweise ein Kursangebot vor dem Hintergrund der Digitalisierung geplant werden, bieten sich neben der gemeinsamen Basis, insbesondere für die Förderung reflexiver Kompetenzen, auch Themen aus der Kategorie (G4) Gesellschaft an.

Außerdem liefern die Ergebnisse Anhaltspunkte dafür, welche Themen bei der Planung entsprechender Angebote direkt oder zumindest mit Anpassungen berücksichtigt werden sollten. Die am häufigsten betrachteten Themen bieten dabei einen Rahmen, der auch aus unterschiedlichen fachlichen Sichten betrachtet werden kann, um bspw. wichtige Algorithmen oder digital repräsentierte Informationen im jeweiligen Fachgebiet zu identifizieren. Repräsentation von Daten könnte sich für Geographinnen und Geographen beispielsweise auf Raster- und Vektordaten oder für Musikerinnen und Musiker auf MIDI und MP3 beziehen. Der Themenbereich künstliche Intelligenz könnte sich exemplarisch auf die Proteinstrukturvorhersage mit neuronalen Netzen für Biologinnen und Biologen oder auf computergenerierte Texte für geisteswissenschaftliche Studiengänge fokussieren.

Die Ergebnisse helfen darüber hinaus, die als für *jeden* als bedeutsam angenommenen Aspekte der Informatik empirisch zu belegen. Damit helfen sie nicht nur konkrete Kursinhalte zu rechtfertigen, sondern können auch zur Diskussion um die Bedeutung und Inhalte informatischer Bildung beitragen. Die quantitativen Betrachtungen erlauben auf Grundlage der Stichprobe Aussagen darüber, welche Themen anteilig als wichtiger erachtet wurden als andere und geben damit Hinweise darauf, welche Inhalte aus dem Themenfeld Informatik für jeden Studierenden relevant sind.

## Literaturverzeichnis


[Ba10] Barr, J. et al.: What everyone needs to know about computation. In: Proceedings of the 41st ACM technical symposium on Computer Science Education. ACM, New York, 2010, S. 127–128.

[BTY18] Bell, T.; Tymann, P.; Yehudai, A.: The Big Ideas of K-12 Computer Science Education. In: Bulletin of EATCS Bd. 124 (2018).

[BD17] Brinda, T.; Diethelm, I.: Education in the Digital Networked World. In: International Federation for Information Processing (IFIP): Proceedings of the World Conference on Computers in Education (WCCE 2017): Springer, 2017, S. 653–657.

[BKM15] Bröker, K.; Kastens, U.; Magenheim, J.: Competences of Undergraduate Computer Science Students. In: KEYCIT 2014 – Key Competencies in Informatics and ICT. Universitätsverlag Potsdam, Potsdam, 2015, S. 77–96.

[Bu16] Bundesministerium für Wirtschaft und Energie: Digitale Strategie 2025, 2016. – www.bmwi.de/Redaktion/DE/Publikationen/Digitale-Welt/digitale-strategie-2025. pdf, Stand: 27. 06. 2018

[BV18] Bundesarbeitgeberverband Chemie e. V.; Verband der chemischen Industrie e. V.: Digitale Bildung – Positionen und Forderungen der chemischen Industrie, 2018. – www.vci.de/langfassungen/langfassungen-pdf/2018-02-12-vci-bavc-digitale-bildung-positionen-forderungen-chemische-industrie.docx, Stand: 27. 06. 2018

[De03] Denning, P.: Great Principles of Computing. In: Communications of the ACM Bd. 46 (2003), Nr. 11, S. 15–20.

[Di11] Dierbach, C. et al.: A Model for Piloting Pathways for Computational Thinking in a General Education Curriculum. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. ACM, New York, 2011, S. 257–262.

[Dö16] Döbeli Honegger, B.: Mehr als 0 und 1: Schule in einer digitalisierten Welt. Bern, Schweiz: hep verlag, 2016.

[Ge16] Gesellschaft für Informatik (GI): Bildungsstandards Informatik SI und SII, 2016. – www.informatikstandards.de, Stand: 27. 06. 2018

[GR14] Grillenberger, A.; Romeike, R.: A Comparison of the Field Data Management and its Representation in Secondary CS Curricula. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education. ACM, New York, 2014, S. 29–36.

[Hu15] Hubwieser, P. et al.: A Global Snapshot of Computer Science Education in K-12 Schools. In: Proceedings of the 2015 ITiCSE on Working Group Reports. ACM, New York, 2015, S. 65–83.

[KP05] Kelleher, C.; Pausch, R.: Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. In: ACM Computing Surveys (CSUR) Bd. 37 (2005), Nr. 2, S. 83–137.

[Ma00] Mayring, P.: Qualitative Content Analysis. In: Forum Qualitative Sozialforschung/Forum: Qualitative Social Research Bd. 1 (2000), Nr. 2.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik Bd. 25 (1993), Nr. 1, S. 20–31.

[VSC17] Vogel, S.; Santo, R.; Ching, D.: Visions of Computer Science Education: Unpacking Arguments for and Projected Impacts of CS4All Initiatives. In: Proceedings of the 2017 ACM Technical Symposium on Computer Science Education. ACM, New York, 2017, S. 609–614.

[Wi06] Wing, J.: Computational thinking. In: Communications of the ACM Bd. 49 (2006), Nr. 3, S. 33–35.

# Analysing & Predicting Students Performance in an Introductory Computer Science Course

**Alexander Tillmann, Detlef Krömker, Florian Horn und Thorsten Gattinger**
Goethe-Universität Frankfurt, studiumdigitale
Varrentrappstr. 40–42
60486 Frankfurt am Main
tillmann@sd.uni-frankfurt.de
kroemker@sd.uni-frankfurt.de
horn@sd.uni-frankfurt.de
gattinger@sd.uni-frankfurt.de

**Abstract:** Students of computer science studies enter university education with very different competencies, experience and knowledge. 145 datasets collected of freshmen computer science students by learning management systems in relation to exam outcomes and learning dispositions data (e. g. student dispositions, previous experiences and attitudes measured through self-reported surveys) has been exploited to identify indicators as predictors of academic success and hence make effective interventions to deal with an extremely heterogeneous group of students.

**Keywords:** Learning analytics, Learning dispositions, Dispositional learning analytics, Formative assessment, Blended learning, heterogeneity.

## 1 Introduction

Due to the dramatically increasing student numbers in computer science studies, the heterogeneity of university entrants is growing continuously. Approaches like learning analytics (LA) try to address this diversity by identifying, collecting and analyzing data features from different sources like student information systems (SIS) or learning management systems (LMS). Through systematically analyzing learning-related data it should be possible to support the learning processes of heterogeneous students by recognizing and consid-

ering individual requirements. By regularly collecting and analyzing data it could be possible to directly react and offer support to prevent students from dropping out of university. The aim is to provide informative feedback on an individual and organizational level and offer opportunities for institutions to support student progress [BFM12; WI12]. Pattern recognition and predictive analytics have not yet been widely used in educational settings [WI12] and much of the data on which LA applications depend comes from learning management systems where grades are only included in some cases. While such learning analytics approaches demonstrated the potential of LA for feedback on the individual level, the findings based on demographics, grades and data tracked from the LMS were rather limited to the descriptive functions of LA. The limits of SIS and LMS data and the lack of important markers of students' heterogeneity thus make it difficult to design pedagogically informed interventions [CH15]. To overcome this shortcoming recent studies have proposed a Dispositional Learning Analytics (DLA) approach. A DLA infrastructure combines data of learning dispositions that impact learning processes (e. g., student dispositions, values, and attitudes measured using self-reported surveys) with data extracted from LMSs and SISs to optimize the connection with learning interventions [RCZ17]. In our empirical research focusing on a introductory module in computer science and programming we provide an application of the theoretical framework of DLA [BC12]. Furthermore the DLA perspective of individual difference characteristics that impact learning processes is expanded by an approach that focuses on the requirements of the currently heterogeneous student body. In order to ensure the implementation of an adequate learner-centered course design particular attention has to be paid to the increasing heterogeneity of the students. Different dimensions of heterogeneity are of particular relevance for higher education [MSM12]: social heterogeneity (age, family status, migrant background, academic background of the family), cognitive heterogeneity (skills, competences), expectancy (vocational and practice-orientation), motivational heterogeneity (procrastination, pragmatism, self-organization) heterogeneous situations in life (professional activity, part-time study, commuter). The main aim of this predictive modeling endeavor is to identify a set of key variables from a rich set of data as a basis for evidence-based decisions concerning a range of interventions intended to deal with the heterogeneity of undergraduate computer scientists. On the one hand, major academic success factors could be identified from the derived datasets, curricula adjusted accordingly and targeted academic support offered for disadvantaged students (e. g. with little prior knowledge of computer science). On the other hand, the LA infrastructure could serve as a

learning coach to provide timely, formative feedback about learning activities and success and indicate a risk of underperformance.

## 2 Learning Analytics and learning dispositions data

### 2.1 Learning Analytics in Higher Education

Learning analytics tries to apply the outcomes of analyzing data collected by monitoring and measuring the learning process and its contexts [Te17]. Empirical studies furthermore show that academic success can be well predicted by a range of demographic, psycho-emotional, cognitive, methodological and social factors [CN12]. Models based on learning psychological and pedagogic theories are often only able to explain up to thirty percent of variance. Learning analytics research that predicts academic success from log data in virtual learning environments [Ag14] indicates that the combination of interaction data from the LMS and learning dispositions data can substantially improve the explained variance of learning success. Demographic characteristics, motivation, conscientiousness and commitment, prior knowledge, skills, competences, talent and personality have all proved to be features of learning dispositions with a significant impact on learning in higher education. By looking at the role of several alternative data sources this study extends the analysis of predictive modeling of academic performance and learning behavior.

### 2.2 Learning dispositions data

In contrast to previous DLA approaches which based their research on a single newly constructed instrument to collect dispositions data [BC12], our study employs well-established and validated instruments. Some learner characteristics and attitudes towards learning can be influenced by education (e. g. promotion of self-reliance, learning strategies). However, other personality qualities like psychometric properties also play a significant role in influencing academic achievement [ZZ16] and are quite stable over time [EKS17; HB17]. Consequently, we cannot try to influence the personality traits of the students so that they better fit to our organization of study programs and curriculum. Having accepting this, we then have to turn the tables: can the various requirements of the students be satisfied by the study program or are adjustments necessary? According to social-constructivist learning theories,

learning is an active process of learning construction in which prior knowledge plays an important role [BP12]. We assume that when learning computer programming the effect of prior knowledge is of particularly crucial importance. Unfortunately the application of an entry diagnostic test of programming skills is time-consuming and could have a dissuasive effect on freshmen students with less prior knowledge. Therefore we would like to find out the extent of the predictive power of self-reported prior knowledge in programming for overall course performance. Other DLA research covering various aspects of affective, behavioral and cognitive antecedents of learning processes has found quite weak relationships between the instruments applied and academic performance [Te17]. Our contribution to the DLA research is based on a case study of freshman students in computer science and focuses on the influence of personality traits on academic achievement and the role of prior knowledge.

## 3 Method

While an increasing body of research is becoming available about the relationship between LMS data and academic performance [e. g. AG14; MD10], about the effects of formative assessment and feedback on learning [BF06], and about the relationship between learning dispositions data covering aspects of affective, behavioral and cognitive antecedents of learning processes [BC12], our empirical research examines the relationship between personality traits and the role of prior knowledge on academic achievement and how all these elements (LMS, formative assessment, learning dispositions data) can be integrated into a research context to analyze the relative contributions of each of the elements to student achievement. With a focus on combining longitudinal learning data extracted from an institutional LMS (including track data extracted from formative tests/quizzes), results from weekly exercises and self-reported learning dispositions data, this study aims to answer the following research questions:

- To what extent does the data of LMSs, formative assessments, personality traits and self-reported learning dispositions predict academic performance over time and what is the most effective data for predictive modeling?
- What is the relationship between learner data sources (LMSs and self-reported data) and assessment data (formative and summative) and to what extent do these predictions overlap?

- Which data source (LMSs, formative assessments or self-reported learner data) has the most potential to generate timely, informative feedback for students?
- To what extent does the LA approach support decisions about pedagogical interventions (e. g. redesign of courses)?

## 3.1 Description of the course and participants

145 datasets of freshmen computer science students enrolled in 2016/2017 on a module in introductory computer science and programming at Frankfurt Goethe-University could be derived. Most of the students, 70 %, are male. The educational setting in which students learn can be described as a blended learning system. One component consists of lectures which are also recorded and available as e-lectures. Attendance of the lectures is optional. Another component is a face-to-face tutorial course with small groups (15 students) and a problem-based learning approach with weekly exercise sheets (11 sheets about basic knowledge and concepts in computer science and seven sheets introducing programming and code writing). Students have to prepare the exercises and subsequently discuss the solution approach with a tutor. Participation in these tutorial groups is also optional, as are the online components of the blended setting. The LMS system Moodle is used to share basic course information and learning resources like PDFs and quizzes. Overall 24 quizzes are available and consist of items which expand on the content of the lecture and tutorial courses. The use of quizzes and participation in tutorial courses is stimulated by making bonus points available for good performance. Overall the bonus is maximized to 20 % of what can be scored in the exam. Due to the very diverse levels of prior knowledge in computer programming we chose this pedagogical setting as it stimulates students with less prior knowledge to make intensive use of learning resources. Learners with less prior knowledge may realize from the continuous feedback that they are falling behind other students, and therefore need to achieve a good bonus score for both, so as to compensate and improve their learning. A good way to do this is to attend the lecture and tutorial courses regularly and use the learning resources on the LMS with e-lectures, quizzes and literature.

## 3.2 Data sources and procedure

**Learning Management System:** Every user action is stored as a database record by Moodle data extraction tool. Following a system-independent classification of interactions in LMSs [Ag14] three different types of interactions associated with virtual learning are possible: student-student interactions (online communication between students, using chats and messages in forums), student-teacher interactions (e. g. interactions involving synchronous and asynchronous tutoring) and student-content interactions (interactions that happen when students make use of the content resources).

In our educational setting students could access e-lectures, various course materials (documents, videos, textbooks) and quizzes and automated feedback on their quiz attempts. These student-content interactions are usually associated with browsing and accessing the different resources, tasks, etc. Communication tools like the E-Mail-Function or course discussion boards were not used systematically, because within our educational setting the students see each other at least twice a week, so there is limited need for online-communication via the LMS. According to the concepts of previous research [Ag14; MD10] and our educational setting, we pre-selected items from the 140 possible interactions tracked in Moodle that describe student-content interactions: accessing the different resources (LMS_content) and the quizzes (LMS_quiz).

**Learning dispositions data:** Three different types of learning dispositions were included in this study: prior knowledge in computer programming, marks in math, English, German and informatics at school, and personality traits. Prior knowledge in computer programming was measured within the first week of the module. The data is based on an instrument adapted to the German language from the computer programming self-efficacy scale (PSES) [AD09; RW98]. The 7-point scale consists of 28 items and the reliability of the scores was $\alpha = .98$; example item: *I could write a computer program that computes the average of three numbers.* In addition, items for collecting information related to gender and marks in high school in math, English, German and informatics were prepared and delivered to the students via the LMS, together with the third component, the measurement of personality traits. This was undertaken using a short version (21 items) of the Big Five Inventory (BFI-K) for assessment of the five factors of personality with very well-tested psychometric properties [RJ05]. The Big Five Factors are: extraversion, agreeableness, conscientiousness, neuroticism and openness to experience [JSS99].

The inventory was administered with a self-report survey scored on a 5-point Likert scale.

**Students' academic performance:** For the predictive modeling three measures of academic performance were included: score in written exam (only exam results, without bonus points), aggregated scores for the exercise sheets about basic knowledge and concepts in computer science (ProgrammingBasics [PBasics]) and the sheets for introduction to programming and code writing (ProgrammingPractice [PPractice]).

**Data analysis:** For further analysis the LMS data was exported into an Excel file and merged with final course exam data, learning dispositions data and the grade data of the exercise sheets. 145 datasets could be derived after merging all data files. Then the dataset was imported into SPSS for statistical analysis. The first step of the analysis consisted of an observation of bivariate correlations between the various datasets and the results of the course exam. Since simply relying on correlations for predictive power may lead to Type II errors, we carried on to confirm the results with multiple regression. For the analysis a data transformation was used to standardize variable data as Z-scores, because the dataset contains variables measured in different scales. The multiple regression was calculated with the final course score achieved by each student as the dependent variable and the LMS datasets, learning dispositions data and results of the exercise sheets as the independent variables. Multiple regressions calculate the variance of the dependent variable as linear combinations of the independent variables. This method makes it possible to generate predictive models for the dependent variable based on data from the independent variables and assigns regression coefficients to each independent variable, allowing us to assess their relative importance in the predictive model. In the study a backwards multiple regression was calculated as this has the advantage that there is no suppression effect such as occurs when independent variables interact with opposite effects, possibly leading to Type II errors [Br13].

## 4 Results and Discussion

Table 1 shows the predictive power, as multiple correlation r, of alternative longitudinal datasets. All correlation coefficients above 0.2 are significant at $p < 0.01$ (two-tailed) and those of 0.2 or smaller at $p < 0.05$ (two-tailed).

Table 1: Multiple bivariate correlations R between various data sources and three academic performance measures. (n. s. = not significant)

| Data source | Exam | PBasics | PPractice |
|---|---|---|---|
| Gender | n. s. | n. s. | .23 |
| Marks in math | .52 | .32 | .42 |
| in English | .24 | n. s. | .26 |
| in German | .44 | .27 | .35 |
| in informatics | .51 | n. s. | .42 |
| PSES | .31 | .19 | .35 |
| Big five factors: | | | |
| extraversion | n. s. | n. s. | n. s. |
| agreeableness | n. s. | n. s. | n. s. |
| conscientious | n. s. | n. s. | n. s. |
| neuroticism | −.18 | n. s. | n. s. |
| openness | n. s. | n. s. | n. s. |

## 4.1 Predictive power of demographic data

Apart from gender (code: female = 1, male = 2) the other tested demographic variables (age and number of semesters) did not show significant correlations with exam results or results from the exercise sheets (PBasic and PPractice). A statistically significant relation was measured between gender and academic performance in solving the exercise sheets with computer programming tasks (PPractice). These exercises were performed better by males. This could be explained by the fact that the group of women had less prior knowledge in computer programming (variable PSES) ($M = 2.7$; $SD = 1.4$) than men ($M = 3.6$; $SD = 1.8$). This difference is significant with $t(137) = -4.2$, $p < 0.01$. Furthermore, there are no gender differences in high school marks, which are also significant predictor variables. This shows the impact of prior knowledge in programming. Even without differences in math, informatics, English or German, women have difficulties catching up with men in programming, because of a lack of prior knowledge.

## 4.2 Predictive power of learning dispositions

Marks in high school all show significant correlations with performance measures. The impact of marks in math is substantial: its beta weight in predicting the course exam is 0.52, explaining in itself 27 % of variation. Also the language marks (German, R = .44 and English, R = .24) and marks in informatics (R = .51, n = 97, not all participants took informatics in school) are significant predictor variables. The data of the computer programming self-efficacy scale (PSES) is also a powerful predictor for exam performance (R = .31) and performance in solving exercise sheets (PBasic, R = .19; PPractice, R = .35). 12 % of variation of the practical computer programming performance can be explained by the self-reported programming skills. In line with social-constructivist learning theories where prior knowledge plays an important role, prior education (marks in school and domain-specific competences) seem to be useful factors to include in learning analytics modeling.

The Big Five Factors of personality do not show substantial predictive power for academic success. In contrast to previous research [GW01] a systematic relation between the factor conscientiousness and learning performance could not be found. There is only one exception: a significant negative correlation between the factor neuroticism and the course exam (R = −.18). This means that students who are uncertain, frightened and nervous perform worse in the course exam, but performed as well as the others in the continuous exercises (PBasic and PPractice). The result shows that the educational setting discriminates against students with a high level of neuroticism. Providing students with past exam papers for practicing purposes or writing a trial exam or completely different forms of examination could be measures to support such students.

## 4.3 Predictive power of LMS data

Given the wealth of LMS student-content interaction data, preliminary analyses were applied to find out which indicators of learning intensity performed well in most of the consecutive weeks. The total number of clicks per week, merging the overall activities, showed the most consistent role in all of the weekly models to predict the impact of student-content interactions (LMS_content) on academic performance. Other variables like *total time online*, *#files viewed, #uploaded,* etc. did not show a consistent role for predictive modeling. The results show little progress in predictive power over time. The

earliest predictions have a beta weight in predicting course exam results of 0.23, indicating that only about 5 % of performance variation can be explained by LMS student-content interactions data. After some weeks without significant correlations between overall LMS user activity and performance at the end of the semester coefficients increase to a medium-large effect size (R = .30–.50) with values up to 0.37 in week 15. This is one week before the exam. This late time close to the exam is not very useful as a prediction model for providing early feedback to students.

A remarkable and consistent feature of prediction is the LMS quizzes dataset. The variable "first score" showed the highest consistency in all of the weekly models and was selected for prediction. Figure 1 demonstrates the predictive power in terms of the multiple correlation coefficients of longitudinal models developed on the Moodle quizzes data for three performance measures. Multiple correlations values demonstrate a medium–large effect size (R = .30–.50) and develop from around R = 0.30 to R = 0.48 at the end of the semester.

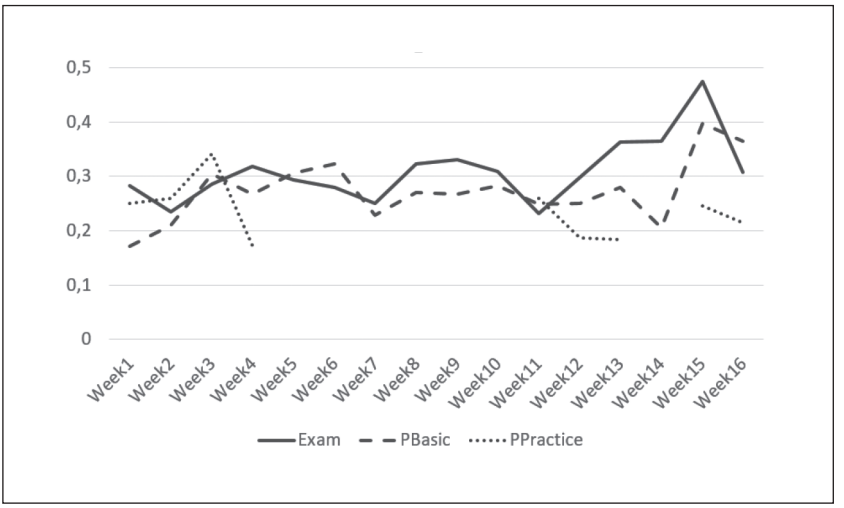


Figure 1: Predictive power of the Moodle quizzes dataset for three performance measures (only significant p < .05 correlations are shown)

In line with previous research [Wo13], the results of quizzes seem to be a reasonable indicator for learning in prediction modeling.

## 4.4 Predicting performance by exercise sheet results

A good predictor for exam performance are the weekly results of the exercise sheets. Exercise sheets about basic knowledge and concepts in computer science (PBasics) had to be prepared from the first week until week 11, and the sheets for the introduction to programming and code writing (PPractice) had to be prepared in weeks one and two and then every second week until week 12. Students who learn continuously throughout the semester and perform the exercises well, also perform better in the final exam. Therefore, performance measured by exercise sheets constitutes a reliable predictor for the exam. Multiple correlation values range from R = 0.20 to R = 0.59, which explains 35 % of variation.

## 4.5 Predictive modeling by multiple regression

Although different variable sets appear to show a significant correlation with final exam results, it would be erroneous to rely too heavily on the predictive power of simple correlations. A multiple backwards stepwise regression was conducted in order to develop a predictive model in which the variable "final course exam" was the continuous dependent variable. The final results of the backwards stepwise multiple regression are shown in Table 2.

Table 2: Coefficient data after multiple backwards regression

| R2 | Corrected R2 | Durbin-Watson |
|---|---|---|
| .727 | .712 | 2.04 |

Model overview after multiple backwards regression

| Variable | Standardized coefficient beta | t | Sig. | VIF |
|---|---|---|---|---|
| LMS_Quiz | .313 | 5.30 | .000 | 1.398 |
| PBasic | .301 | 4.53 | .000 | 1.765 |
| PPractice | .258 | 3.74 | .000 | 1.899 |
| Marks German* | .131 | 2.31 | .023 | 1.277 |
| Marks math* | .119 | 2.00 | .048 | 1.409 |
| PSES* | .114 | 2.09 | .039 | 1.186 |

* Learning Dispositions Dataset

The analysis generated a predictive model of students' exam results as a combination of LMS quiz tracking data, learning dispositions data and formative assessment data. The Durbin-Watson coefficient value indicates that there are no auto-correlation problems in the model. All six variables are statistically significant contributors ($p < 0.05$) and the values of the variance inflation factor (VIF) suggest that multicollinearity effects may be ruled out in this analysis. The multiple squared correlation coefficient for the model is .727, indicating that some 73 % of the variability in students' final performance in this course can be explained by this combination of LMS data and learner data. The single most predictive variable reported here, with a regression coefficient (Beta) of .313 is the LMS tracking variable LMS_Quiz, measuring the total achievement of the first scores of online self-test quizzes. The variable represents student engagement with the learning content, rather than their effort in completing graded course assessments.

The student-content interaction data, measured by the number of total clicks per week, is dominated by the predictive power of the other data components. The LMS student-content interactions data has no added value in predicting performance and was excluded from the predictive model by the backwards stepwise regression analysis. An important finding is that knowledge of actual course design (e. g. what online resources are available, online communication between students and students, and students and teachers) is critical in determining which students are not engaging with course materials in a manner that is indicative of an effective learning strategy. In line with previous findings [MD10; Wo13; Xi15] quizzes, integrated in the course design, seem to be a good indicator of students' engagement in learning and therefore also for predictive modeling. The second and third significantly predictive variables PBasic and PPractice represent student engagement in solving exercise sheets, also used as a formative assessment to stimulate continuous learning throughout the semester with the possibility of gaining nine bonus points for the final exam. The final three significantly predictive variables observed in this study come from the student learning dispositions dataset. They are related to prior knowledge. Students with better marks in German and math and prior knowledge of computer programming, measured by the computer programming self-efficacy scale (PSES), achieve higher overall final exam results. The Big Five Factors of personality do not show substantial predictive power for academic success. The variables were excluded from the model. Learning dispositions data is not as easily collected as tracking data from the LMS [BC12]. Whether the effort involved in collecting learning dispositions data, like prior knowledge in computer programming or high school marks, is

worthwhile or not also depends on whether this improves the ability to predict the passing rate.

To test the predictive power of the model in terms of whether or not an individual student is considered "at risk of failure", binary logistic regressions were calculated. Students with course exam scores < 50 % were coded as "at risk" (0), while students with exam scores ≥ 50 % were coded as "performing adequately or better" (1). In our course scoring scheme < 42 % was considered a fail. The division point was selected to include students whose final exam scores indicate that they barely passed the course and may have benefited from earlier feedback, support and intervention. The logistic regression model (Tab. 3) accurately placed individual students in either the "at risk" or "performing adequately or better" category 91.9 % of the time. The model resulted in errors, classifying an "at risk" student as "performing adequately or better" at a rate of only 5.2 %: only seven students out of 135 were predicted to be performing adequately or better when their actual final exam score placed them in the "at risk" category. Out of these seven students, only three actually failed the course. They did not achieve scores over 42 %. Three out of 135 represents a predictive failure rate of only 2.2 %.

Table 3: Logistic regression "Risk of failure" classification results (N = 135)

| **Observed** | **Predicted** | | |
|---|---|---|---|
| | **At risk** | **Not at risk** | **Percentage correct** |
| At risk | 37 | 7 | 84.1 |
| Not at risk | 4 | 87 | 95.6 |
| Overall percentage | | | 91.9 |

Note: "At risk" = exam score < 50 %; "Not at risk" = exam score ≥ 50 %

By placing four students in the "at risk" category even though these students eventually passed the course (achieving scores of > 50 %) the model predicted wrongly 3 % of the time. This error seems to be of less concern because it seems better to mistakenly identify a student as being at risk of failure than to neglect a student who requires additional learning support. If these results had been accessible earlier in the semester, the majority of students who failed or almost failed and those who would have been recognized "at risk of failure" could have been identified by teachers.

The value of the learning dispositions data for prior knowledge (expressed as self-reported programming skills and high school marks in German and math) can be demonstrated by a binary logistic regression analysis where only these three variables are included in the model (Tab. 4).

Table 4: "Risk of failure" classification results from three dispositions data variables (N = 141)

| Observed | Predicted | | |
|---|---|---|---|
| | At risk | Not at risk | Percentage correct |
| At risk | 19 | 27 | 41.3 |
| Not at risk | 12 | 83 | 87.4 |
| Overall percentage | | | 72.3 |

Overall, the model accurately placed the students 72.3 % of the time (Tab. 5), which shows that even with only these three variables collected in the first week of the semester the model is still comparatively precise. In the model 27 students out of 141 were predicted to be performing adequately, while their final course score placed them in the "At risk" category. This is a failure rate of (only) 20 %. However, as soon as quiz and formative assessment data become available their predictive power is dominant.

## 5 Conclusions

In this exploratory study we integrated data from several different sources and found evidence for the strong predictive power of data from LMS tracked quiz data and formative assessment data. We also found evidence for the predictive power of some learning dispositions data. Predictions of academic performance using data of domain-specific skills like prior knowledge in computer programming or high school marks in – for the field of knowledge – relevant subjects were much more accurate than when using personality trait data. The Big Five Factors of personality could not be used for predictive purposes. The role of student-content interactions data from the LMS (total number of weekly clicks) also appeared to be minimal and was thus also excluded from predictive modeling. The combination of LMS quiz tracking data, the results

of exercise sheets, data of prior education (high school marks), and self-reported prior knowledge led to the predictive model with the greatest significance. Therefore learning analytics should combine LMS data with learner data. As soon as this data is available the generation of timely feedback based on performance predictions and early signaling of underperformance is possible. Differences in prior knowledge and education of the students seem to have a high impact on learning success, which could be quite accurately predicted with our dataset. The high cognitive heterogeneity of the students becomes a key issue for course planning and learning process planning. One idea for dealing with this challenge is to select students with less prior knowledge in computer programming and invite them to a special tutorial so that they can catch up with the others. Future studies should be directed towards the investigation and analysis of potential indicators in relation to other dimensions of heterogeneity such as social heterogeneity, heterogeneous situations in life, or motivational heterogeneity. Even if previous findings show weaker relationships between motivational heterogeneity data and academic performance [Te17], these kinds of learning dispositions could be used in combination with LMS and assessment data to provide better predictions, e. g. in cases where the quality of quizzes and tests are weak (e. g. when they are focusing on factual knowledge only) and do not fit well to the requirements of the final course exam.

## References


[Ag14] Agudo-Peregrina, Á. F. et al.: Can we predit success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning. Computers in Human Behavior, 31, S. 542–550, 2014.

[AD09] Askar, P.; Davenport, D.: An investigation of Factors related to self-efficacy for java programming among engineering students. The Turkish online Journal of Educational Technology, 8(1), S. 26–32, 2009.

[BFM12] Bienkowski, M.; Feng, M.; Means, B.: Enhancing teaching and learning through educational data mining and learning analytics: An issue brief. US Department of Education, Office of Educational Technology, S. 1–57, 2012.

[BF06] Boud, D.; Falchikov, N.: Aligning assessment with long-term learning. Assessment & Evaluation in Higher Education, 31, 4, S. 399–413, 2006.

[Br13] Brosius, F.: SPSS 21. mitp, Heidelberg, 2013.

[BC12] Buckingham Shum, S.; Crick, R. D.: Learning dispositions and transferable competencies: Pedagogy, modelling and learning analytics. In (Buckingham Shum, S.; Gasevic, D.; Ferguson, R. Eds.): Proceedings of the 2nd international conference on learning analytics and knowledge. New York, S. 92–101, 2012.

[BP12] Bulu, S. T.; Pedersen, S.: Supporting problem-solving performance in a hypermedia learning environment: The role of students' prior knowledge and metacognitive skills. Computers in Human Behavior. 28, 4, S. 1162–1169, 2012.

[CH15] Conde, M. A.; Hernandez-García, Á.: Learning analytics for educational decision making. In: Computers in Human Behavior, 47, S. 1–3, 2015. DOI: http://dx.doi.org/10.1016/j.chb.2014.12.034

[CN12] Credé, M.; Niehorster, S.: Adjustment to college as measured by the student adaptation to college questionnaire: A quantitative review of its structure and relationships with correlates and consequences. Educational Psychology Review, 24(1), S. 133–165, 2012.

[EKS17] Elkins, R. K.; Kassenboehmer, S. C.; Schurer, S.: The stability of personality traits in adolescence and young adulthood. Journal of Economic Psychology. 60, 37–52, 2017.

[GW01] Gibbons, D. E.; Weingart, L. R.: Can I do it? Will I try? Personal efficacy, goals, and performance norms as motivators of individual performance. Journal of Applied Social Psychology. 31, S. 624–648, 2001.

[HB17] Hopwood, C. J.; Bleidorn, W.: Stability and change in personality and personality disorders. Current Opinion in Psychology, In press, accepted manuscript, 2017.

[JSS99] John, O. P.; Svrivastava, J.; Svrivastava, S.: The Big-Five trait taxonomy: History, measurement, and theoretical perspectives. In (Pervin, L. A.; John, O. P. Eds.): Handbook of personality: Theory and research, 2, New York, S. 102–138, 1999.

[MD10] MacFadyen, L. P.; Dawson, S.: Mining LMS data to develop an "early warning system" for educators: A proof of concept. Computers & Education. 54(2), S. 588–599, 2010.

[MSM12] Metzger, C.; Schulmeister, R.; Martens, T.: Motivation und Lehrorganisation als Elemente von Lernkultur. Zeitschrift für Hochschulentwicklung, 2012. DOI: http://zfhe.at/index.php/zfhe/article/view/433

[RW98] Ramalingam, V.; Wiedenbeck, S.: Development and Validation of Scores on a Computer Programming Self-efficacy Scale and Group Analysis of Novice Programmer Self-efficacy. Journal of Educational Computing Easearch, 9(4), S. 367–381, 1998.

[RJ05] Rammstedt, B.; John, O. P.: Kurzversion des Big Five Inventory (BFI-K): Entwicklung und Validierung eines ökonomischen Inventars zur Erfassung des fünf Faktoren der Persönlichkeit. Diagnostica, 51,(4), S. 195–206, 2005.

[RCZ17] Rienties, B.; Cross, S.; Zdrahal, Z.: Implementing a learning analytics intervention and evaluation Framework: What works? In (Daniel, B. K. Ed.): Big data and learning analytics: Current theory and practice in higher education, Cham: Springer International Publishing, S. 147–166, 2017. DOI: http://dx.doi.org/10.1007/978-3-319-06520-5_10

[SDL13] Siemens, G.; Dawson, S.; Lynch, G.: Improving the quality of productivity of the higher education sector: Policy and strategy for systems-level deployment of learning analytics. Society for Learning Analytics Research, 2013.

[Te17] Tempelaar, D. et al.: Student profiling in a dispositional learning analytics application using formative assessment. Computers in Human Behavior. S. 1–13, 2017.

[WI12] Wagner, E.; Ice, P.: Data Changes Everything: Delivering on the Promise of Learning Analytics in Higher Education. In: EDUCAUSE Review Online. 2012. http://educause.edu/ero/article/data-changes-everything-delivering-promise-learning-analytics-higher-education.

[Wo13] Wolff, A. et al.: Improving retention: Predicting at-risk students by analysing clicking behaviour in a virtual learning environment. In: Paper presented at the proceedings of the third international conference on learning analytics and knowledge, 2013.

[Xi15] Xing, W. et al.: Participation-based student final performance prediction model through interpretable Genetic Programming: Integrating learning analytics, educational data mining and theory. Computers in Human Behavior, 47, S. 168–181, 2015.

[ZZ16] Zhang, J.; Ziegler, M.: How do the big five influence scholastic performance? A big five-narrow traits model or a double mediation model. Learning and Individual Differences. 50, S. 93–102, 2016.

# Vorlesungs-Pflege: Maßnahmen zur Runderneuerung degenerierender Informatikvorlesungen

**Karsten Weicker**
HTWK Leipzig, Fakultät IMN
Gustav-Freytag-Straße 42A
04277 Leipzig
karsten.weicker@htwk-leipzig.de

**Zusammenfassung:** Ähnlich zu Alterungsprozessen bei Software degenerieren auch Vorlesungen, wenn sie nicht hinreichend gepflegt werden. Die Gründe hierfür werden ebenso beleuchtet wie mögliche Indikatoren und Maßnahmen – der Blick ist dabei immer der eines Informatikers. An drei Vorlesungen wird erläutert, wie der Degeneration von Lehrveranstaltungen gegengewirkt werden kann. Mangels hinreichend großer empirischer Daten liefert das Paper keine unumstößlichen Wahrheiten. Ein Ziel ist es vielmehr Kollegen, die ähnliche Phänomene beobachten, einen ersten Anker für einen inneren Diskurs zu bieten. Ein langfristiges Ziel ist die Sammlung eines Katalogs an Maßnahmen zur Pflege von Informatikvorlesungen.

**Schlüsselworte:** Wartung von Lehrveranstaltungen, Re-Engineering, Strukturverbesserung, Lehrevaluation, Degenerationsprozesse.

## 1 Motivation

Regelmäßig gehaltene Vorlesungen scheinen trotz der zunehmenden Reife, jahrelanger Verbesserung der Beispiele und damit stetig zunehmendem Perfektionsgrad, trotz einer immer ausgefeilteren Vorbereitung und Durchführung der Vorlesung immer weniger erfolgreich zu sein – Studenten[1] zeigen weniger Begeisterung als die Kommilitonen vor zehn Jahren und

1 Alle geschlechterbezogenen Bezeichnungen stehen für Personen jeglichen Geschlechts; also „Studenten“ für „Studentinnen und Studenten“ – analog: Informatiker, Dozent, Kommilitone, Kollege etc.

die Prüfungsergebnisse lassen ebenfalls nach. Ich bezeichne den Effekt als Tao-of-Pooh-Paradoxon nach dem Buchzitat „And when you try too hard, it doesn't work" [Ho82] – trotz gleichbleibender oder gar zunehmender eigener Anstrengungen ist man als Dozent festgefahren in dem Gefüge der eigenen Lehrveranstaltung und kann eher nachteilige Auswirkungen der eigenen Bemühungen beobachten. Wenn sich bei einer wiederholt zu haltenden Vorlesung beim Dozenten das Gefühl einstellt, dass die Dinge eher schlechter als besser werden, bedeutet dies, dass es notwendig sein kann, sich der Vorbereitung anders zu nähern, als man dies bisher gemacht hat. Auch wenn dies kein gesichertes Naturgesetz ist, lässt es sich immer wieder beobachten.

Vielfältige Erklärungsversuche können bemüht werden, um die Ursachen des Tao-of-Pooh-Paradoxon zu beleuchten. Da es in unserem Lehr- und Forschungsgebiet allerdings um Informatikvorlesungen geht, bietet es sich an, das Problem mit den Augen des Informatikers zu untersuchen und entsprechende Lösungsvorschläge aufzuzeigen.

Dieses Paper hat einen Fokus auf der Lehrveranstaltungsform der klassischen Vorlesung – genauer: auf Vorlesungen, die von demselben Dozenten über einen längeren Zeitraum immer wieder gehalten werden. Dabei kann es sich ebenso um Grundlagenveranstaltungen wie um Spezialvorlesungen handeln. Andere Veranstaltungsformen wie Seminare, Praktika, Übungen und Selbstarbeit werden in diesem Paper nicht berücksichtigt, womit die Vorlesung nicht über die anderen Lehrformen gestellt werden soll. Allerdings ist die Vorlesung gerade im technisch-naturwissenschaftlichen Kontext kaum verzichtbar und bedarf deshalb einer besonderen Berücksichtigung.

Abschnitt 2 und Abschnitt 3 stellen frische und degenerierte Vorlesung einander gegenüber. Nach möglichen Indikatoren für eine beginnende Degeneration in Abschnitt 4 zeigt Abschnitt 5 auf, welche Maßnahmen zur Vorlesungspflege zur Verfügung stehen. Einige persönliche Beispiele aus den eigenen Lehrveranstaltungen werden in Abschnitt 6 präsentiert. Die Ausführungen erheben dabei nicht den Anspruch auf eine unumstößliche wissenschaftliche Wahrheit, sondern sie sind streitbar und sollen Kollegen, die ähnliche Phänomene beobachten, einen ersten Anker für einen inneren Diskurs bieten.

## 2 Frische Vorlesungen

Ein wesentliches Ziel einer Vorlesung ist die Wissensvermittlung. Wie heute hinlänglich bekannt ist, funktioniert dies nicht gemäß des Nürnberger Trichters, sondern es ist notwendig, dass die Studierenden mitdenken und ihr Vorwissen mit den neuen Impulsen verknüpft wird. Allein durch eine aktive Beschäftigung mit den Inhalten können diese Verknüpfungen entstehen.

Um dieses Ziel der aktiven Auseinandersetzung auch in Vorlesungen erreichen zu können, gibt es eine Reihe von didaktischen Maßnahmen wie z. B. aktivierende Methoden. Doch auch darüber hinaus fällt dem Dozenten eine entscheidende Rolle zu, inwieweit es ihm gelingt, den sprichwörtlichen Funken auf die Studenten überspringen zu lassen und ihr Interesse zu wecken. Diese Fähigkeit kann man an den folgenden zwei Eigenschaften festmachen.

- Die *Begeisterung* des Dozenten für seine Inhalte und seine Vorlesung beeinflusst, wie stark sich die Studentinnen und Studenten auf das Fach einlassen – Begeisterung ist ansteckend.
- Die *Vorbildfunktion* des Dozenten als Bild für die Denk- und Herangehensweise von Informatikern. Wird Informatik als reine Routine oder als spannendes, sich ständig weiter entwickelndes Gebiet gesehen und gelebt? Grundsätzlich vermittelt ein Studium in erster Linie Haltungen, Denkweisen und Problemlösekompetenzen. Dies gilt insbesondere und in verstärktem Maß in der Informatik, wo sich Themen, Programmiersprachen und Technologien schnell ändern – laut [HSZ13] bestehen Forschungs-Communities etwa 4,5 Jahre lang, Lehrinhalte wurden 1995 mit einer Halbwertszeit von 5 Jahren beziffert [Sc95]. Vor diesem Hintergrund fällt dem Dozenten eine Vorbildfunktion zu, wie Informatiker in ihrem weiteren Leben mit Neuerungen umgehen sollten: Flexibilität und Aktualität in der Lehre statt eines antiquierten Anstrichs.

Gerade diese beiden Eigenschaften leiden allerdings häufig darunter, wenn dieselben Vorlesungen immer wieder gehalten werden und sich eine Routine breit macht. Im weiteren Text bezeichnen wir eine wiederholt gehaltene Vorlesung als *frisch,* wenn die Begeisterung und die Vorbildfunktion gleichermaßen erhalten bleiben.

# 3 Degenerierende Vorlesungen

Um die Güte einer Vorlesung zu messen, ist es wichtig, zunächst einmal grundlegende Eigenschaften von Vorlesungen allgemein zu betrachten. Hier drängt sich dem Informatiker ein Vergleich der Eigenschaften von Vorlesungen mit denen von Software auf. Dabei ergeben sich viele z. T. unerwartete Parallelen, aber auch einige Unterschiede. Nach [Ba09] ist Software

- immateriell, was auch für Vorlesungen und vor allem für das zu vermittelnde Wissen selbst gilt (durch ihre inhärente Struktur und den Vortrag selbst von Vorlesungen; einzig die Vorlesungsfolien und ein eventuelles Skript sind in gewissem Sinn „anfassbar"),
- verschleißfrei, was oberflächlich betrachtet auch für Vorlesungen gilt, im Weiteren aber noch etwas näher betrachtet wird, und
- einem Alterungsprozess ausgesetzt, der genauso auch für die Vorlesung gilt (siehe Abbildung 1): Die Umgebung einer Lehrveranstaltung entwickelt sich weiter und verändert sich. Die mathematische Vorbildung von Studienanfängern nimmt ab [Ba13], durch Prägung mittels Messenger-Systemen beträgt deren Aufmerksamkeitsspanne nur noch etwa 280 Zeichen, die Erwartungen, die an Absolventen gestellt werden, ändern sich, wie sich auch die vermittelten Inhalte in Lehrveranstaltungen von Kollegen verschieben und nicht zuletzt entwickelt sich die Informatik weiter und insbesondere (aber nicht ausschließlich) technologielastige Module laufen ständig Gefahr inhaltlich veraltet zu sein. Lehrveranstaltungen, die diesen Änderungen nicht folgen, veralten zwangsläufig.

Zum zweiten Punkt, der Verschleißfreiheit: Eine einmal konzipierte Lehrveranstaltung zu gut gesetzten und gereiften Grundlagen, z. B. Logik, Automaten und formale Sprachen oder Algorithmen und Datenstrukturen, sollte doch problemlos nach 10 oder 15 Jahren noch unverändert funktionieren – wenn man vom oben beschriebenen Alterungsprozess absieht, der sich bei Grundlagenveranstaltungen nur auf veränderte Rahmenbedingungen seitens der Studienanfänger beschränken sollte. Das gilt allerdings nur mit gewissen Einschränkungen, denn das Problem sind wir als Dozenten:

- Erläuterungen zu den Vorlesungsfolien werden über die Jahre hinweg beständig optimiert und leicht verändert – und das trägt nicht zwingend zur Verbesserung der Lehrveranstaltung bei: zunehmende Perfektion verringert die Mitdenklücken für Studierende und manchmal verschieben

sich auch leicht die eigenen Erwartungen, was die Studierenden eigentlich schon wissen sollten.

- Auch die anfängliche Begeisterung des Dozenten für die Vermittlung seiner Inhalte weicht einer Routine, was im Extremfall sogar bis zum Disengagement, also der inneren Kündigung, wie im Schulkontext [SV11] führen könnte. In jedem Fall hat schwindende Begeisterung auf Seiten des Dozenten einen großen Einfluss auf die Wahrnehmung seitens der Studierenden und damit auch auf deren Bereitschaft, sich mit der Materie auseinanderzusetzen.

Zusammenfassend halten wir fest, dass verschiedene Faktoren (vgl. Abbildung 1) dafür sorgen, dass Vorlesungen degenerieren können und damit Maßnahmen der Vorlesungs-Pflege erforderlich sind.

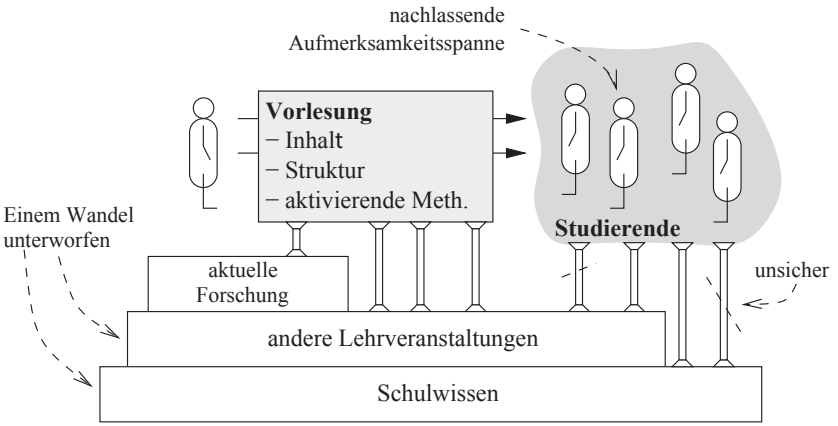


Abb. 1: Überblick über Faktoren, die eine langsame Degeneration einer Vorlesung begünstigen

## 4 Kriterien für frische Vorlesungen

Wie kann man nun einschätzen, wie frisch eine Vorlesung noch ist? Ideal wäre natürlich eine Metrik als Qualitätskriterium, ähnlich zu Metriken wie wir sie in der Software-Entwicklung kennen. Kandidaten sind

- die Durchschnittsnoten in Klausuren – weniger ein Frischekriterium, sondern von vielen anderen Faktoren überdeckt,
- die Ergebnisse der Lehrevaluation – eher ein Wohlfühlindikator – oder
- das Maß der aktiven Beteiligung in der Vorlesung oder in Übungen – kann sowohl Verständnisschwierigkeiten oder begeistertes Mitdenken bedeuten.

Alle angeführten Qualitätsmerkmale lassen sich zwar messen, ergeben jedoch höchstens in Kombination und unter starker Einbeziehung des Kontexts eine wirkliche Aussage. Alternativ könnte man durch Fragebögen oder andere empirische Methoden über einen längeren Zeitraum hinweg untersuchen, ob sich die studierendenseitige Wahrnehmung der Vorlesung ändert.

Da dieses Paper nicht auf eine empirische Untersuchung abzielt, sondern in erster Linie ein Problembewusstsein wecken soll, bleibt als gut gemeinter Hinweis an Dozenten nur das eigene Gefühl als Indikator und Alarmsignal. Zunehmende Unzufriedenheit mit der studentischen Mitarbeit oder den Ergebnissen in Klausuren und Belegen, zunehmendes Abspulen der Vorlesung oder auch vermehrte kritische Kommentare bei Lehrevaluationen könnten ein erstes Anzeichen für eine degenerierende Vorlesung sein.

## 5 Maßnahmen zur Vorlesungspflege

Wenn die Analogie zwischen Software und Vorlesungen halbwegs schlüssig ist, bietet sich auch für die Kategorisierung von Maßnahmen der Vorlesungspflege ein Blick auf die Software-Pflege an. Opferkuch und Ludewig [OL04] unterscheiden die folgenden Kategorien:

- Software-Wartung als jede Arbeit an einem Software-System, die unmittelbare Auswirkungen auf den Benutzer hat und nicht von Anfang an geplant war oder hätte geplant werden können – dies möchten wir in unserem Kontext uminterpretieren als jegliche Änderung an einer Vorlesung, die eine inhaltliche Veränderung nach sich zieht, und

- Re-Engineering als absehbare Strukturverbesserung, die der ausschließlichen Verbesserung von Wartungsqualitäten dient – in unserem Kontext jegliche Veränderung an einer Vorlesung bei gleichbleibendem Inhalt (einschließlich Umstrukturierungen, die auch das leichte Verankern von neuen Inhalten ermöglichen).

Beispielhafte Maßnahmen zur Vorlesungswartung sind

- das Bereinigen von Fehlern,
- die Überarbeitung durch eine frische, eigene, in die Tiefe gehende Auseinandersetzung mit dem Inhalt, z. B. durch neue anwendungsnahe oder aktuelle Beispiele, die selbst im Detail aufbereitet werden, bzw. die beispielhafte Umsetzung von Algorithmen in einer aktuellen Programmiersprache,
- die Berücksichtigung von aktuellen Forschungsergebnissen und Veröffentlichungen – dies kann eigene wie auch fremde Forschung sein, aber auch der Bericht zu aktuellen Themen vom Besuch einer Fachtagung – und
- das Infragestellen von Standardlehrstoff innerhalb der Lehrveranstaltung, wie z. B. der Frage warum statt Quicksort der Timsort-Algorithmus in den Java-Collections implementiert ist.

Beim Re-Engineering einer Vorlesung handelt es sich um strukturverändernde Maßnahmen wie z. B.

- jegliches Umstellen der Reihenfolge von Inhalten – dadurch können unterschiedliche Ziele verfolgt werden, wie z. B. die bessere Anknüpfung an Vorwissen, eine bessere Verknüpfung von Inhalten oder durch eine anschließende Vorlesungswartung die Verschiebung des Fokus der Vorlesungseinheit auf ein übergeordnetes Konzept – und
- als Extrembeispiel Konzepte wie Flipped Classroom, die bei gleichbleibenden Inhalten den Ablauf des Vorlesungsalltags maßgeblich verändern und die Rollen zwischen Präsenzveranstaltung und Selbstarbeit tauschen.

# 6 Beispiele

Ich möchte die Überlegungen an einigen Beispielen aus meinen eigenen Vorlesungen illustrieren.

## 6.1 Spezialvorlesung „Evolutionäre Algorithmen“

Die Lehrveranstaltung „Evolutionäre Algorithmen“ wird seit 1999 gehalten und hat im Rahmen der Weiterentwicklung des Forschungsgebiets aber auch der Überarbeitung des Lehrbuchs [We15] für Neuauflagen verschiedene Änderungen erfahren. Der grundsätzliche Aufbau der Vorlesung wurde seit 2007 allerdings nur marginal modifiziert.

Daher machten sich Degenerationserscheinungen bemerkbar. Ein anfangs sehr erfolgreicher großer Theorie-/Konzeptblock am Beginn des Moduls funktionierte mit den Jahren immer weniger gut und hat am Ende sogar das Verständnis der Standardalgorithmen erschwert. Dies machte sich auch in den Ergebnissen der Lehrevaluation bemerkbar. So wurde die Frage „Diese Lehrveranstaltung fördert mein Interesse am Thema“ im Sommersemester 2009 mit 1,36 (auf einer Skala von 1 = „sehr gut“ bis 5 = „mangelhaft“) und im Wintersemester 2012/13 mit 1,9 bewertet. Auch bei der Frage „Ich habe in dieser Lehrveranstaltung tiefes Verständnis für den Stoff gewonnen“ verschlechterte sich der Wert von 2,0 auf 2,3.

Eine strukturelle Überarbeitung der Lehrveranstaltung war unumgänglich, die ich in erster Linie dazu genutzt habe, Konzepte und Standardalgorithmen parallel einzuführen und so klarer die Bezüge dazwischen zu verdeutlichen. Abbildung 2 stellt bildlich dar, wie die Inhalte aus den bisherigen zwei großen Abschnitten zu Konzepten und Standards umverteilt und wie ein Sandwichbelag in kleinere Einheiten gebettet werden. Dasselbe Prinzip wurde auch mit den Fallstudien am Ende der Vorlesung angewandt, wo jeweils eine Fallstudie mit Techniken zum Umgang mit speziellen Anforderungen verknüpft wird.

Das reine Re-Engineering der Vorlesung genügte mir an dieser Stelle allerdings nicht, sondern ich entschloss mich zusätzlich zu den bewährten Algorithmenvisualisierungen kleine Programmbeispiele mit einzuflechten. Hierbei war das Ziel, einerseits die einfache Umsetzung der vorgestellten Ideen zu betonen und andererseits mehr Spaß und Begeisterung in die Vorlesung zu bringen. Daher habe ich die Beispiele in der noch relativ jungen Sprache Go implementiert. In jeder der ersten sieben Vorlesungseinheiten wird ein

kleines Beispielprogramm präsentiert, dessen Quelltext genauer analysiert und damit kleine Optimierungsexperimente durchgeführt.

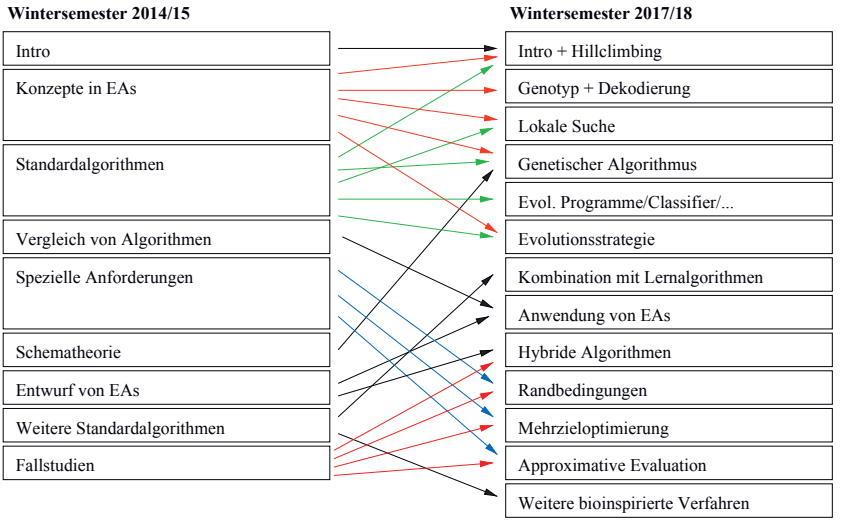


Abb. 2: Vergleich der Struktur des Moduls „Evolutionäre Algorithmen“ vor und nach der Restrukturierung

Die Lehrveranstaltung wurde erstmalig in dieser Form im Wintersemester 2017/18 gehalten und auch wieder durch die Lehrevaluation begleitet. Die Frage „Diese Lehrveranstaltung fördert mein Interesse am Thema“ wurde jetzt mit 1,3 und die Frage „Ich habe in dieser Lehrveranstaltung tiefes Verständnis für den Stoff gewonnen“ mit 1,8 bewertet. In den Freitextkommentaren wurde insbesondere die Verwendung von Go bzw. von vielen praktischen Demonstrationen und Beispielen positiv hervorgehoben, wobei die typischen Gefahren von viel Quelltext in einer Vorlesung auch hier negativ vermerkt wurden und das Vorstellen von Code von einem Zuhörer als „oft verwirrend und nicht hilfreich beim Verstehen“ bezeichnet wurde. Insgesamt hat die Lehrveranstaltung durch die Veränderungen neuen Schwung bekommen, was meines Erachtens nach maßgeblich an der neu entfachten Begeisterung des Dozenten für die frische (und manchmal auch noch nicht ganz perfekt ausgearbeitete) Vermittlung des Lehrinhalts liegt.

## 6.2 Grundlagenveranstaltung „Algorithmen und Datenstrukturen“

An der HTWK Leipzig findet die Lehrveranstaltung „Algorithmen und Datenstrukturen“ im zweiten Fachsemester des Bachelorstudiums statt. Auch diese Lehrveranstaltung wurde mehreren Strukturveränderungen unterworfen. Die größte Modifikation war im Jahr 2007 die Neustrukturierung gemäß eines Spiralkonzepts, das bereits an anderer Stelle [We07] beschrieben wurde: fünf algorithmische Probleme dienen als roter Faden, der sich durch die Lehrveranstaltung zieht und an dem verschiedene algorithmische Ideen verankert werden (vgl. linker Teil von Abbildung 3). Der Aufbau führte zu einer wesentlich verbesserten Vermittlung der involvierten Konzepte.

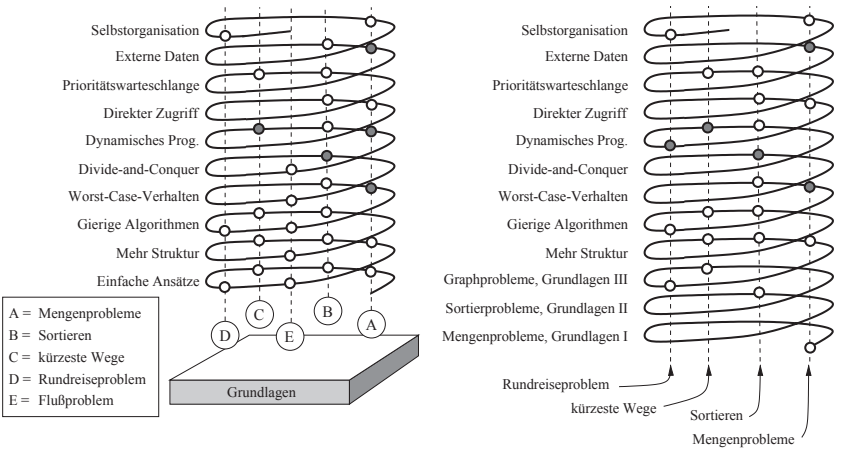


Abb. 3: Vergleich der Struktur der Vorlesung „Algorithmen und Datenstrukturen“. Links das ursprüngliche Spiralkonzept, rechts die Überarbeitung mit dem veränderten Einstieg in die Lehrveranstaltung

Allerdings hinterließen neun Jahre bis zum Jahr 2016 auch bei dieser Lehrveranstaltung Spuren der Degeneration. Maßgeblich haben sich in dieser Zeit die Vorkenntnisse der Studierenden verändert, da sich Ausrichtung und Inhalt von Lehrveranstaltungen im ersten Fachsemester leicht verschoben haben.

Die konkreten Maßnahmen waren hier weniger einschneidend. Ähnlich zum Re-Engineering der Spezialvorlesung in Abschnitt 6.1 wurden auch hier die Grundlagen mit den einfachen Lösungsansätzen verschränkt, sodass eine

stärker im Beispielproblem verhaftete Einführung der Grundlagen entstand (vgl. Abbildung 3). Unabhängig von der Re-Strukturierung wurde durch Verkürzung der Vorlesungszeit an der Hochschule die Menge der betracheten algorithmischen Probleme um ein Problem verkleinert.

## 6.3 Praxisorientierte Lehrveranstaltung „Softwaretechnik“

Das Modul „Softwaretechnik“ wird von mir seit 2004 als Lehrveranstaltung des dritten Bachelorfachsemesters gehalten und zeigt insgesamt wesentlich weniger Symptome der Degeneration. Das liegt daran, dass die Lehrveranstaltung durch Überarbeitungen mindestens alle zwei Jahre einem großen empirischen Dauerexperiment gleicht, bei dem ich regelmäßig Inhalte erneuere oder austausche. Die Auslöser hierfür sind vielfacher Natur: der Wegfall einer Folgeveranstaltung „Softwaretechnik II“, aktuelle Entwicklungen in der Branche oder die eigene Verschiebung von Schwerpunkten. Zwei wesentliche Einflussfaktoren waren die Einführung von Programmierübungen im Computerpool sowie die parallel startende Lehrveranstaltung des Softwareprojekts.

Die Übungen im Poolraum finden 14-tägig statt und es müssen im Rahmen des 90-minütigen Termins verschiedene Aufgaben individuell gelöst werden. Die Themenblöcke umfassen Source-Code-Management, Testen mit JUnit, Entwurfsmuster, Refactoring, Modellierung mit UML und werkzeuggestützte Code-Erzeugung. Der Zwang, über den gesamten Verlauf der Lehrveranstaltung beständig auch praktisch ein- und umsetzbare Fachinhalte zu präsentieren, hatte einen positiven Effekt auf den Erhalt der Frische der Lehrveranstaltung.

Ähnlich positiv hat sich das Softwareprojekt ausgewirkt, das in größeren Teams mit Masterstudenten als Projektleiter parallel im dritten Semester startet und über den Zeitraum von zwei Semestern läuft. Als Konsequenz muss die Lehrveranstaltung „Softwaretechnik“ beständig und rechtzeitig Prozesswissen für die Durchführung von Softwareprojekten liefern, welches durch die Bachelorstudenten auch fortwährend mit ihren praktischen Erfahrungen im größeren Projektkontext abgeglichen wird.

## 7 Diskussion und Ausblick

Erfolgreiche Lehrveranstaltungen sind kein Selbstläufer sondern müssen gezielt gepflegt werden. Hierbei hilft zumindest dem Informatikdozenten die Analogie zur Software-Entwicklung. Als wesentliches Ziel sollte dabei die Begeisterung für den Stoff und die Veranstaltung sowohl beim Lehrenden als auch bei den Studierenden erhalten werden. Die Beispiele in diesem Paper stützen sich stark auf den Re-Engineering-Aspekt, also der strukturellen Überarbeitung eines Moduls. Hierfür können didaktische Konzepte oder auch einfache pragmatische Gedanken als Leitidee der Überarbeitung wirksam werden. Drei typische Muster kamen in den präsentierten Beispielen zur Anwendung: das Spiralkonzept als iterierendes, Wissen in Schichten aufbauendes, didaktisches Prinzip, die sandwichartige Verschränkung von Grundlagen bzw. Konzepten mit Standardalgorithmen und die Einbettung von Fallbeispielen in den fortgeschrittenen Teil der Lehrveranstaltung.

Für die Zukunft wäre es wünschenswert, wenn ein konkreter Katalog von möglichen Maßnahmen für die Pflege von Vorlesungen gesammelt werden könnte – ähnlich zu den Entwurfsmustern bei der Software-Architektur, die auch für typische Probleme durch ein Schema eine Lösung anbieten. Auch hierfür bietet es sich an, den Kontext noch weiter zu öffnen – wie das Beispiel des Moduls „Softwaretechnik“ zeigt, ist das Potential der möglichen Ansätze zur Pflege einer Lehrveranstaltung wesentlich größer, wenn man betreute Übungen im Computerpool, Projektarbeiten oder die Schnittstellen zu parallel gehaltenen Lehrveranstaltung berücksichtigt. Auch aus der Hochschuldidaktik können noch wesentlich mehr Impulse berücksichtigt werden, man denke nur an problembasiertes Lernen oder Learning-by-Teaching.

## Literaturverzeichnis


[Ba09] Balzert, H.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Akademischer Verlag, 2009, S. 9.

[Ba13] Baumann, A.: Mathe-Lücken und Mathe Legenden: Einige Bemerkungen zu den mathematischen Fähigkeiten von Studienanfängern. Die Neue Hochschule, 2013(5), S. 150–153, 2013.

[Ho82] Hoff, B.: The Tao of Pooh, E. P. Dutton, 1982.

[HSZ13] Hoonlor, A.; Szymanski, B. K.; Zaki, M. J.: Trends in Computer Science Research. Communications of the ACM, 56(10), S. 74–83, 2013.

[OL04] Opferkuch, S.; Ludewig, J.: Software-Wartung – Eine Taxonomie. Softwaretechnik-Trends, 24(2), S. 35–36, 2004.

[Sc95] Schinzel, B.: „Welchen Wert haben theoretische Grundlagen in der Berufspraxis? Was Theorie leisten kann und soll.“ Universalismus, Abstraktion und Modellbildung in der Informatik. In (Huber-Wäschle, F.; Schauer, H.; Widmayer, P. Hrsg.): GISI 95: Herausforderungen eines globalen Informationsverbundes für die Informatik, Springer, S. 366–373, 1995.

[SV11] Schmitz, E.; Voreck, P.: Einsatz und Rückzug an Schulen: Engagement und Disengagement bei Lehrern, Schulleitern und Schülern, VS Verlag, 2011.

[We07] Weicker, N.: Zielorientierte Didaktik der Informatik – Kompetenzvermittlung bei engen Zeitvorgaben. In (Schubert, S. E. Hrsg.): Didaktik der Informatik in Theorie und Praxis. INFOS2007, Lecture Notes in Informatik 207, Gesellschaft für Informatik, S. 337–348, 2007.

[We15] Weicker, K.: Evolutionäre Algorithmen, 3. Auflage, Springer Vieweg, 2015.

# Programmieren lernen & Softwareentwicklung

# Ein Lehr- und Lernkonzept für die Softwareentwicklung im Team

**Dennis Schmitz und Daniel Moldt**
Universität Hamburg, Fachbereich Informatik
Vogt-Kölln-Str. 30
22527 Hamburg
schmitz@informatik.uni-hamburg.de
moldt@informatik.uni-hamburg.de

**Abstract:** Um beim Berufseinstieg erfolgreich als Informatiker wirken zu können, reicht es oft nicht aus nur separierte Kenntnisse über technische und theoretische Grundlagen, Programmiersprachen, Werkzeuge und Selbst- und Zeitmanagement zu besitzen. Vielmehr sollten Absolventen diese Kenntnisse praktisch miteinander verzahnt einsetzen können. An der Universität wird Studierenden leider selten die Möglichkeit geboten, diese verschiedenen Bereiche der Informatik miteinander integriert auszuüben. Dafür entwickeln wir seit über zwei Dekaden ein Lehr- und Lernkonzept zur Unterstützung praktischer Softwareentwicklungsveranstaltungen und setzen dieses um. Dadurch bieten wir angehenden SoftwareentwicklerInnen und ProjektmanagerInnen eine Umgebung, in der sie neues, praktisch relevantes Wissen erwerben können, sich selbst praktisch erproben und ihr Wissen konkret einsetzen können. Hier legen wir einen Schwerpunkt auf das Arbeiten im Team. Das hier vorgestellte Konzept kann auf ähnliche Lehrveranstaltungen übertragen und aufgrund seiner Modularisierung verändert und erweitert werden.

**Keywords:** Softwareentwicklung, Lehre, Teamarbeit, Projekte.

## 1 Einleitung

Eine Herausforderung an die universitäre Ausbildung besteht in der Einsetzbarkeit der Absolventen in Projekten in der Praxis ohne lange Einarbeitungsphasen. Vor dem Hintergrund der theoretischen, konzeptionellen Ausrichtung des Studiums erfordern Projekte praktisches Wissen und Erfahrungen in der

Anwendung von Konzepten, Techniken, Methoden und Werkzeugen. Somit sind organisatorische und persönliche Fähigkeiten bei der Interaktion mit anderen ein wichtiger Bestandteil der Ausbildung.

Bereits seit mehr als zwei Dekaden wird von unserem Arbeitsbereich systematisch an der Konzeption und Umsetzung eines Lehr- und Lernkonzepts in Praktika und Projekten gearbeitet. Neben den fachlichen Inhalten liegt der Schwerpunkt auf der Vermittlung einer ganzheitlichen Sicht auf ein Projekt. Zentraler Bestandteil ist, dass die Studierenden erlernen, wie sie gemeinsam an einer Aufgabenstellung arbeiten. Werkzeuge (wie Versionsverwaltung, integrierte Entwicklungsumgebungen, Continuous Integration, Testumgebungen, Kommunikations- und Projektmanagementwerkzeuge etc.) werden systematisch eingesetzt, um das Team bei der kollaborativen Arbeit zu unterstützen.

Als Lehrveranstaltungsformen werden Praktika und Projekte adressiert (4.–6. B. Sc. und 1.–3. M. Sc. Semester). Im Folgenden fassen wir die beiden Veranstaltungsformen unter dem Begriff *Lehrprojekt* zusammen. Dabei haben wir Erfahrungen mit Lehrprojektgrößen von sechs bis über 30 Teilnehmenden gesammelt.

Der in unserem Arbeitsbereich entwickelte Softwareentwicklungsansatz Paose (Petrinetz-, Agenten- und Organisationsorientierte Softwareentwicklung) basiert auf unseren Erfahrungen. In diesem Beitrag präsentieren wir didaktische Methoden und wie wir diese mittels Werkzeugeinsatz unterstützen. Nach der Benennung unserer erfahrenen Herausforderungen gehen wir detailliert auf die Durchführung des Lehrprojekts ein, diskutieren eingesetzte Lehr- und Lernkonzepte und erläutern deren Effekte.

## 2 Anforderungen, Inhalte und Herausforderungen

Um selbst eine geeignete Evaluation unserer Lehrprojekte vornehmen zu können, definieren wir zu Beginn Anforderungen, die wir im Laufe des Projekts erfüllen wollen. Daraus ergeben sich die Anforderungen, die von unserem hier präsentierten Lehr- und Lernkonzept berücksichtigt werden müssen. Die Anforderungen lassen sich grob in die folgenden Kategorien einteilen: (1) Anforderungen, die die Betreuenden selbst erfüllen müssen/wollen; (2) Anforderungen, die die Studierenden während und am Ende des Projekts erfüllen sollen; (3) Anforderungen, die die entwickelte Software erfüllen muss/soll. In diesem Beitrag konzentrieren wir uns auf die zweite Kategorie. Die erste und dritte werden hier lediglich exemplarisch umrissen.

Zu den Anforderungen der ersten Kategorie zählen z. B. ein neues Lehrkonzept, neue Konzepte, Techniken, Methoden oder Werkzeuge zu erproben sowie seine eigenen Projektmanagementfähigkeiten zu verbessern. Die Anforderungen der dritten Kategorie setzen sich aus den drei Gruppen konzeptioneller, anwendungsbezogener und technischer Anforderungen zusammen [Ba10]. Die ersten beiden Gruppen sind abhängig von dem Anwendungsgebiet der Software. Die Anforderungen der dritten Gruppe ergeben sich sowohl aus dem Anwendungsgebiet als auch aus der technischen Umgebung, die eingesetzt werden soll.

In unseren Lehrprojekten lehren wir u. a. die Paose. Die Paose ist ein umfassender Softwareentwicklungsansatz für die verteilte agentenorientierte Softwareentwicklung. Dieser beinhaltet Vorgaben für die Softwaretechnik, das Vorgehen und die Kollaboration der Softwareentwickler. Die grundlegende Programmiersprache in der Paose basiert auf *Java Referenznetzen* [Ku02], die mit der integrierten Entwicklungsumgebung (IDE) Renew [CHM16] modelliert werden. Außerdem ermöglicht Renew die Modellierung mit weiteren Techniken. Die erstellten Modelle werden zur Generierung von Programmcode und zur Konfiguration eingesetzt. Eine detaillierte Beschreibung der Paose lässt sich in [Ca10] finden.

Die zweite Kategorie der Anforderungen fokussiert auf die Studierenden. Die Studierenden sollen möglichst viele Facetten der Softwareentwicklung im Team kennenlernen. In unseren Projekten gehen wir auf die technischen, die sozialen und die Managementbereiche ein, welche wir im Folgenden beschreiben. Außerdem heben wir die – basierend auf unserer Erfahrung – prägnantesten Herausforderungen hervor.

## 2.1 Techniken und Werkzeuge

Die Studierenden lernen in unseren Projekten eine neue Programmiersprache (*Java Referenznetze*). Sonstige bei uns auftretende Auszeichnungs-, Programmier-, Skript- und Spezifikationssprachen sind *Java, JavaScript, Java Server Pages, HTML, CSS, Bash, XML* und *LaTeX*. Die am meisten eingesetzte IDE ist Renew, hinzukommt – je nach eigener Präferenz – Eclipse[1] oder Intellij IDEA[2].

1 https://www.eclipse.org/, zugegriffen am 17. 04. 2018
2 https://www.jetbrains.com/idea/, zugegriffen am 17. 04. 2018

Zusätzlich lernen die Studierenden ein – für die meisten – neues Softwareentwicklungsparadigma, die agentenorientierte Softwareentwicklung sowie unseren Softwareentwicklungsansatz Paose. In diesem werden Techniken eingesetzt, die vielen Studierenden unbekannt sind bzw. speziell für die Paose entwickelt wurden. Zu diesen zählen z. B. vier spezielle agentenorientierte, aber UML-ähnliche Diagramm- und Modelltypen, auf die hier jedoch nicht weiter eingegangen wird. Deren Beschreibung ist in [Ca10] zu finden.

Darüber hinaus setzen wir auch weit verbreitete Techniken und Werkzeuge ein. So verwenden wir die Linux-Distribution Ubuntu und die Kommandozeile (Bash). Letzteres stellt für viele Studierende eine besondere Herausforderung dar. Außerdem nutzen wir *Apache Ant*[3], *Git*[4], *Jenkins*[5] und *Redmine*[6]. Diese Bandbreite an Techniken und Werkzeugen zu erlernen und effektiv einzusetzen kann die Studierenden bereits zu Beginn der Projekte überfordern und abschrecken.

## 2.2 Soziales

Häufig kennen die Studierenden sich untereinander zuvor nicht und können somit weder die Fähigkeiten und den Kenntnisstand der anderen einschätzen noch wie diese sich in einem Team verhalten. Diese Aspekte sind für ein erfolgreiches Softwareentwicklungsprojekt jedoch entscheidend.

Wir betonen bereits zu Beginn eines jeden Projekts wie wichtig es ist, dass die lose Gruppe der Studierenden zu einem Team zusammenwächst. Um dies zu fördern, verdeutlichen wir den Studierenden, welche sozialen Ängste und Vorurteile in einer solchen Situation die Teambildung negativ beeinflussen können. Wir weisen darauf hin, dass die Studierenden zum Lernen an dem Projekt teilnehmen und nicht, um sich gegen andere zu behaupten. Die Zusammenarbeit mit erfahreneren Softwareentwicklern stellt somit eine Chance für sie dar und keine Herausforderung. Außerdem heben wir hervor, dass, wenn jemand Schwächen in der Programmierung aufzeigt, er durchaus Stärken in der Entwicklung kreativer Lösungen auf konzeptioneller Ebene haben kann. Wir warnen also davor, dass die Studierenden sich gegenüber zu voreingenommen sind.

---

3 https://ant.apache.org/, zugegriffen am 17. 04. 2018
4 https://git-scm.com/, zugegriffen am 17. 04. 2018
5 https://jenkins.io/, zugegriffen am 17. 04. 2018
6 https://www.redmine.org/, zugegriffen am 17. 04. 2018

Eine weitere Herausforderung stellen Studierende dar, die partiell wenig Vorwissen besitzen (Studierenden belegen in der Regel verschiedene Studiengänge) oder weniger begabt sind und somit bereits früh im Projekt drohen von den anderen Studierenden abgehängt zu werden. Meistens ist es möglich, dass diese Personen durch andere Studierende unterstützt werden, so dass die Unterstützenden sich selbst in die Rolle der Betreuenden begeben. Wir erwarten von unseren Studierenden nämlich nicht nur, dass sie sich selbst in das Team integrieren, sondern auch dabei mitwirken andere in das Team zu integrieren und diese bei aufkommenden Schwierigkeiten zu unterstützen.

## 2.3 Management und Kommunikation

In unseren Softwareentwicklungsprojekten setzen wir auf eine flache Hierarchie. Somit fordern wir von den Studierenden ein adäquates Aufgaben-, Selbst- und Zeitmanagement. Außerdem beinhalten unsere Projekte Phasen in denen räumlich, zeitlich und organisatorisch voneinander getrennt gearbeitet wird (verteilte Phasen).

Da die Studierenden meistens keine Erfahrung in der Softwareentwicklung im Team besitzen, fehlt ihnen häufig die Fähigkeit, die anstehenden Aufgaben sinnvoll zu priorisieren und zu parallelisieren. Besonders in den verteilten Phasen ist das Verständnis über die Abhängigkeiten und die Zuständigkeiten der Aufgaben wichtig. Andernfalls können durch eine unzureichende Priorisierung Situationen entstehen, in denen andere nicht mit ihrer Arbeit fortfahren können, weil ihnen z. B. notwendige Informationen fehlen. Sind die Zuständigkeiten von Aufgaben nicht ersichtlich, kann es zu der Mehrfachbearbeitung einer Aufgabe kommen oder die Studierenden wissen nicht, an wen sie sich mit einer konkreten Frage wenden sollen. Unsere Herausforderungen bestehen darin, ihnen diese Sachverhalte zu vermitteln, sie rechtzeitig auf entsprechende Situationen hinzuweisen und ihnen zu zeigen, wie die Verwendung von Projektmanagementwerkzeugen Abhilfe schaffen kann. Die organisatorisch-fachliche Aufteilung erfolgt anhand der Matrixorganisation von Paose [Ca07].

Des Weiteren sollen die Studierenden lernen ihre verrichtete Arbeit zu präsentieren, um die restlichen Teammitglieder über ihren aktuellen Fortschritt, aufkommende Probleme und kreative Ideen zu unterrichten. Wie wir die in diesem Abschnitt hervorgehobenen Herausforderungen angehen, beschreiben wir in den beiden folgenden Abschnitten.

# 3 Durchführung der Lehrprojekte

Der Arbeitsaufwand für die Studierenden in den Lehrveranstaltungen beläuft sich auf sechs (Praktikum) bzw. neun (Projekt) ECTS Punkte. Die verpflichtende Anwesenheitszeit für die Studierenden beträgt wöchentlich vier (ein Termin) bzw. sechs Stunden (zwei Termine). M. Sc. Projekte haben zusätzlich ein Seminar mit drei ECTS Punkten.

Die Struktur unserer Lehrprojekte besteht aus vier Phasen, wobei die Studierenden i. d. R. nur an den zwei mittleren Phasen mitwirken. Vor Semesterbeginn befinden sich die Betreuenden in der *Vorbereitungsphase*. Mit Semesterbeginn startet das Projekt in die *Einarbeitungsphase,* um im späteren Verlauf des Semesters in die *Softwareentwicklungsphase* überzugehen. Nach dessen Abschluss beginnen die Betreuenden mit der *Nachbereitungsphase*. Wir beschränken uns in diesem Beitrag auf die Einarbeitungsphase und die Softwareentwicklungsphase. Anschließend wird außerdem auf die Beurteilung der Leistungen der Studierenden eingegangen.

## 3.1 Die Einarbeitungsphase

In der Einarbeitungsphase bekommen die Studierenden die Möglichkeit sich mit den Techniken, den Werkzeugen, der agentenorientierten Softwareentwicklung und dem Softwareentwicklungsansatz Paose vertraut zu machen. Die Studierenden lernen sich in dieser Phase kennen, indem sie in unterschiedlichen Konstellationen miteinander Aufgaben lösen und sich gemeinsam neues Wissen erarbeiten. Dadurch wird eine gemeinsame Wissensbasis etabliert, auf der wir anschließend ein Softwareentwicklungsprojekt aufsetzen. Aus diesem Grund nutzen wir bereits in der Einarbeitungsphase Werkzeuge, die in der Softwareentwicklungsphase verwendet werden.

Da die Betreuenden in dieser Phase die zentralen Ansprechpersonen darstellen, bildet sich das Team i. d. R. um sie herum. Die Projektkonzeption erfordert die durchgehende Anwesenheit der Betreuenden. Bei großen Gruppen ist eine Unterstützung durch weitere Personen, die mit der Umgebung vertraut sind, hilfreich.

**Aufgabenblätter**

Die Basis dieser Phase bilden Aufgabenblätter, anhand derer die Studierenden sukzessive die Entwicklung von Multiagentensystemen (MAS) mit einem Agentenrahmenwerk erlernen. Die Aufgabenblätter bauen jeweils auf einander auf und führen Schritt für Schritt zu der Entwicklung einer verteilten webbasierten Chat-Applikation.

Den Zeitraum für die Bearbeitung eines Aufgabenblatts passen wir individuell auf die jeweilige Form der Lehrveranstaltung und auf die Fähigkeiten der Studierenden an. Somit wird ein Aufgabenblatt im Praktikum in ein bis zwei Wochen bearbeitet und im Projekt i. d. R. in einer Woche.

Damit die Studierenden seltener die Hilfe von Betreuenden benötigen und somit schneller und intensiver Unterstützung erhalten, haben wir die Aufgabenblätter überarbeitet. Wir haben die Prozesse, die zur Lösung der einzelnen Aufgaben führen, extrahiert und die Struktur der Aufgabenblätter nach dem Vorbild der Prozesse entworfen. Das Ergebnis sind Aufgabenblätter, die explizit das Vorgehen zur Lösung einer Aufgabe zeigen, jedoch nicht die Lösung selbst vorwegnehmen. Die Studierenden müssen die Aufgaben somit weiterhin selbst lösen, werden dabei jedoch in ihrem strukturierten Vorgehen unterstützt. Durch den prozessorientierten Charakter bieten die Aufgabenblätter die benötigten Informationen zu den richtigen Zeitpunkten an. Außerdem verweisen die Aufgabenblätter auf Informationen, die als Hilfestellungen zur Lösung der Aufgaben genutzt werden können und einzelne Themengebiete vertiefen. Diese weiterführenden Informationen stellen wir in der Form eines Wikis bereit. Einen tieferen Einblick in die Gestaltung der Aufgabenblätter bieten wir in [Sc16].

**Bearbeitung der Aufgabenblätter**

Während der Anwesenheitszeit setzen wir bei der Bearbeitung der Aufgabenblätter auf ein erweitertes Konzept des *Pair-Programmings*. Um zu vermeiden, dass einzelne Paare abgehängt werden und um das Kennenlernen zwischen den Studierenden zu unterstützen, mischen wir die Paare etwa alle 30 Minuten neu. Ein neu geformtes Paar setzt seine Arbeit bei dem jeweils geringeren Bearbeitungsstand fort. Somit ist gewährleistet, dass die Studierenden sich bereits während der Bearbeitung intensiv über ihre Lösungen austauschen und miteinander auseinandersetzen.

Für die Heimarbeitszeit lassen wir die Studierenden kleine Gruppen bilden, in denen sie den Rest der Aufgabenblätter lösen und ihre gemeinsamen Lösungen anschließend in der Form einer kurzen Präsentation aufbereiten.

Wir wählen für jede Aufgabe ein bis zwei Lösungen aus, die dann im Plenum besprochen werden. Dabei heben wir die Inhalte hervor, die durch die Bearbeitung des Aufgabenblatts vermittelt werden sollen und greifen Themen auf, die Schwierigkeiten bereitet haben.

**Präsentationen**

Die Aufgabenblätter werden von Präsentationen unterstützt, die die jeweilige Thematik zu Beginn eines Aufgabenblatts aufbereitet darstellen. Dadurch wissen die Studierenden, worauf das kommende Aufgabenblatt abzielt und sie können sich leichter Zusammenhänge erschließen. Während einige Präsentationen von den Betreuenden durchgeführt werden, ist jede(r) Studierende dazu verpflichtet selbst mind. eine Präsentation zu halten. Die Studierenden fokussieren in diesen Präsentationen gebräuchliche Werkzeuge und Vorgehensweisen in der Softwareentwicklung. Jede(r) Studierende arbeitet sich also in eine Thematik ein, wächst dadurch im Projekt für diese zu einem Experten heran und steht somit den Kommilitonen für Nachfragen zur Verfügung. Dadurch nehmen die Studierenden selbst in einem Teilbereich des Projekts die Rolle einer/eines Betreuenden ein.

**Wiki**

Als zentrales Wiki für unsere Lehrprojekte nutzen wir das integrierte Wiki von Redmine. In diesem beschreiben wir die einzelnen Aspekte, die für die Entwicklung von MAS mit dem verwendeten Agentenrahmenwerk essentiell sind und wie diese miteinander integriert werden. Außerdem sind alle Werkzeuge, die wir einsetzen, ihre Konfiguration und wie sie ineinandergreifen beschrieben. Darüber hinaus gibt es zu jedem Aufgabenblatt eine Wiki-Seite, die die relevanten Themen verlinkt.

Die Studierenden werden von uns dazu angehalten, ihnen fehlende Informationen selbstständig zu ergänzen. Dadurch entsteht eine gemeinsame Wissenssammlung, die Studierenden können die Lernumgebung selbst prägen und es entsteht ein Verantwortungsbewusstsein für den Lernerfolg aller derzeitigen und zukünftigen Projektteilnehmer.

**Projektmanagement**
Um die Studierenden bereits in dieser Phase an ein *Issue-System* zu gewöhnen, bilden wir für jede(n) Studierende(n) jedes Aufgabenblatt als eine hierarchische Sammlung von Tickets ab. Wir beschränken uns dabei auf folgende Granularität: Jede (Teil-)Aufgabe eines Aufgabenblatts ist ein Ticket. Die Tickets werden alle von uns inklusive einer Aufwandsschätzung erstellt. Die Studierenden müssen lediglich den Bearbeitungsstatus anpassen und ihre aufgewendete Zeit eintragen. Durch dieses Vorgehen erlernen die Studierenden den Umgang mit Tickets und können ihren Fortschritt im Vergleich mit ihren Kommilitonen selbst einschätzen.

Die Betreuenden erhalten so die Möglichkeit den Fortschritt der Studierenden auch außerhalb der Anwesenheitszeit einzusehen. Zusätzlich lassen sich die von den Studierenden gelieferten Ergebnisse in Relation zu der aufgewendeten Zeit setzen und so potentielle Schwierigkeiten identifizieren.

## 3.2 Die Softwareentwicklungsphase

Nachdem sich die Betreuenden und die Studierenden in der Einarbeitungsphase eine gemeinsame Wissensbasis geschaffen, sich auf Konventionen geeinigt sowie sich untereinander kennen gelernt haben, beginnt die Softwareentwicklungsphase. Damit die Studierenden sich im Laufe dieser Phase von ihrer eigenen Unsicherheit weniger gehemmt fühlen, machen wir ihnen bewusst, dass sie sich nicht in einer Prüfungssituation bzgl. ihrer softwaretechnischen Fähigkeiten befinden. Vielmehr bewerten wir in dieser Phase ihr Engagement, um die sozialen Kompetenzen der Studierenden zu stärken.

Die Betreuenden nehmen – neben ihren Softwareentwicklungsaufgaben, die jeweils im Paar mit einem Studierenden bearbeitet werden – in dieser Phase eine beratende und dirigierende Rolle ein. Sie stellen sicher, dass die Richtlinien des Softwareentwicklungsansatzes sowie Konventionen eingehalten werden. Zwar sollen die Studierenden eigenverantwortliches, selbstorganisiertes Arbeiten lernen, jedoch ist nicht zu unterschätzen, dass die meisten Studierenden unerfahren in der Softwareentwicklung im Team sind. Ohne ein zentrales Projektmanagement ist diese Phase nur selten von Erfolg gekrönt. Je nach Konstellation des Teams, können die Betreuenden ihre dirigierenden Aufgaben im Laufe dieser Phase auf die Studierenden verteilen. Die Studierenden nehmen somit außer der Rolle des Entwicklers auch organisatorische, konzeptionelle, beratende und lehrende Rollen ein.

**Struktur**
Wir folgen in der Softwareentwicklung zwar den mittlerweile gängigen Konzepten von Meilensteinen und Sprints, haben jedoch kein vollständiges Scrum implementiert. Zum Beispiel greifen wir nicht explizit auf alle Scrum-Rollen zurück. Zu Beginn eines Sprints definieren wir gemeinsam mit den Studierenden den neuen Meilenstein. Am Ende eines Sprints analysieren wir in der Retrospektive, welche Anforderungen erfüllt wurden und was die Gründe für das Nichterfüllen von Anforderungen waren. Wir verfolgen die gleiche Anwesenheitspflicht wie in der Einarbeitungsphase, um wenigstens ein bzw. zwei Mal in der Woche zentralisiert arbeiten zu können. Die restliche Zeit in der Woche wird meist räumlich und zeitlich verteilt gearbeitet. Wir haben die Erfahrung gemacht, dass längere verteilte Phasen besonders bei unerfahrenen Softwareentwicklern zu Problemen führen und das Projekt negativ beeinflussen.

**Zu entwickelnde Softwareapplikation**
In der Regel definieren die Betreuenden grobe Anforderungen an die zu entwickelnde Applikation. In den letzten Jahren haben wir dabei vermehrt auf die Unterstützung von Softwareentwicklern abgezielt, die in einem verteilten Team arbeiten. Dies bringt zwei Vorteile mit sich: die Studierenden kennen in etwa den Anwendungskontext und sie denken darüber nach, wie sie selbst von einer technischen Umgebung unterstützt werden wollen. Somit reflektieren sie ihre eigene Arbeitsweise im Team. Aus den Erfahrungen, die sie während der Entwicklung der Applikation machen, können sie außerdem weitere Anforderungen ableiten.

**Kommunikation und Koordination**
Unser grundlegendes Werkzeug für das Management des Softwareentwicklungsprojekts ist der Issue-Tracker von Redmine. Die Studierenden haben bereits in der Einarbeitungsphase gelernt, wie sie diesen dazu nutzen können, die aktuellen Tätigkeiten und den Fortschritt einzusehen. Außerdem wird die Entwicklung der einzelnen Komponenten und die Definition der Schnittstellen in dem in Redmine integrierten Wiki dokumentiert. Dadurch wird eine indirekte Kommunikation etabliert, die den Studierenden zeigen soll, wie wichtig diese Dokumentation ist. Studierende, die die Dokumentation vernachlässigen, erfahren, dass sie häufig auf gleiche Nachfragen reagieren müssen und somit in ihrer Arbeit unterbrochen werden.

Des Weiteren setzen wir auf eine klare Strukturierung der Applikation und leiten von dieser die Struktur des Softwareentwicklungsteams ab [Ca07]. Die Beteiligten werden jeweils einer oder mehreren Softwarekomponenten zugeordnet. Somit bilden sich kleine Gruppen, die jeweils für eine Softwarekomponente zuständig sind. Auf diese Weise bietet die strukturelle Übersicht der Applikation gleichzeitig die strukturelle Übersicht des Softwareentwicklungsteams. Die Interaktion zwischen Softwarekomponenten impliziert somit die Interaktion zwischen den einzelnen Gruppen, da diese gemeinsame Schnittstellen und auszutauschende Daten definieren müssen. Dies stärkt das Bewusstsein der Studierenden darüber, mit wem sie sich koordinieren müssen. Außerdem weisen wir die Studierenden darauf hin, zu Beginn eines Sprints die potentiellen Treffen mit anderen Gruppen zu koordinieren und auf diese Aufgaben ein besonderes Augenmerk zu legen.

Wir haben festgestellt, dass wir durch die Visualisierung dieser Beziehungen das Bewusstsein der Studierenden für die nötige Interaktion mit ihren Kommilitonen auf einfache Weise verstärken können. Dazu nutzen wir ein sog. *Coarse Design Diagram* (CDD, s. Abbildung 1). Da es die einzelnen Komponenten der zu entwickelnden Applikation darstellt (Agenten-Rollen werden als Akteure und Agenten-Interaktionen als Anwendungsfälle dargestellt), können die Studierenden die Schnittstellen ihrer Komponente sehen. Außerdem annotieren wir die einzelnen Komponenten mit den Namen der verantwortlichen Studierenden bzw. Betreuenden. Weitere Unterstützung für die Koordination von Teammitgliedern in verteilten Softwareentwicklungsprojekten ist in [Sc18] zu finden.

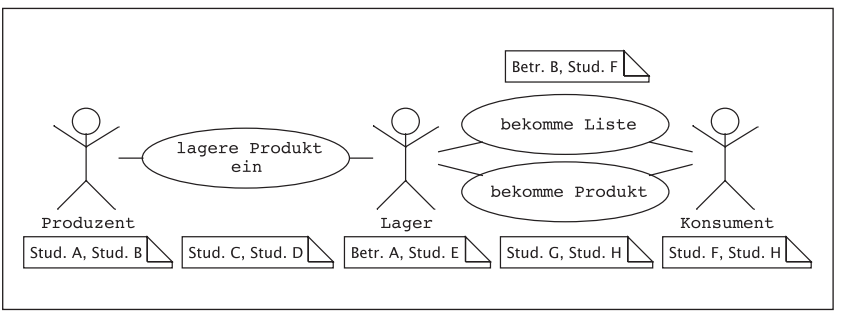


Abb. 1: Coarse Design Diagram (CDD) eines Produzent-Lager-Konsument-Beispiels

**Retrospektive**
Zusätzlich zu den applikationsbezogenen und softwaretechnischen Themen, nehmen wir uns entweder mit dem gesamten Team oder auch mit ausgewählten Teammitgliedern die Zeit, um Gründe zu erörtern, die zu einem reibungslosen Ablauf oder zu Schwierigkeiten geführt haben. Gemeinsam erarbeiten wir Lösungen, wie Schwierigkeiten in Zukunft vorgebeugt und reibungslose Abläufe hergestellt werden können. Diese Komponente bildet einen wichtigen Aspekt in dem iterativen Lernprozess der Studierenden während der Softwareentwicklungsphase und sollte nicht vernachlässigt werden.

## 3.3 Leistungsbeurteilung

Am Ende unserer Lehrprojekte wird keine einheitliche Prüfung durchgeführt. Hier gehen wir kurz darauf ein, wie wir die individuellen Leistungen der Studierenden beurteilen.

In der Einarbeitungsphase beurteilen wir die Studierenden danach, ob sie die Aufgabenblätter fristgerecht bearbeitet und die Ergebnisse in der geforderten Form eingereicht haben, sowie nach ihren eigenen Präsentationen. Da die Studierenden die einzelnen Aufgaben nicht alleine, sondern gemeinsam mit unterschiedlichen Studierenden bearbeiten, eignen sich die Aufgabenergebnisse schlecht als Bewertungsgrundlage. Wir beurteilen jedoch die individuelle Entwicklung der Studierenden und sprechen sie darauf an, wenn wir der Meinung sind, dass sie sich z. B. mit einem bestimmten Werkzeug intensiver beschäftigen sollten. Dies kontrollieren wir – mit Ankündigung – beim nächsten Termin.

Die Beurteilung in der Softwareentwicklungsphase ist schwieriger. Zwar lassen sich aus Redmine und Git normative Werte – wie z. B. die aufgewendete Zeit oder die Anzahl eingepflegter Code-Zeilen – extrahieren, jedoch lässt sich z. B. kein Wert für das aufgebrachte Engagement in der Integration ins Team und in Diskussionen aus diesen Werkzeugen gewinnen. Die Betreuenden stützen sich dafür auf ihre Erfahrungen und Beobachtungen, die sie regelmäßig untereinander diskutieren. Dies wird dadurch unterstützt, dass die Betreuenden Teil des Softwareentwicklungsteams sind, eng mit den Studierenden zusammenarbeiten und mit ihnen in kontinuierlichem Austausch stehen.

# 4 Diskussion der eingesetzten Lehr- und Lernkonzepte

Im Folgenden benennen wir konkrete Lehr- und Lernkonzepte bzgl. des vorherigen Abschnitts und diskutieren, warum wir diese Konzepte einsetzen. Diese Betrachtung dient außerdem als Grundlage für Lehrende, um ähnliche praktische Lehrprojekte zu gestalten.

**Gemeinsame Einarbeitungsphase** In unserem Kontext ist die gemeinsame Einarbeitungsphase bereits aufgrund der für die Studierenden unbekannten Paose und den zugehörigen Werkzeugen notwendig. Jedoch plädieren wir darauf, mit Studierenden auch in bekannteren Kontexten eine gemeinsame Einarbeitungsphase vor einem Softwareentwicklungsprojekt durchzuführen. Diese Phase dient besonders dem Kennenlernen, dem Einüben von Arbeitsprozessen und der Einigung auf Konventionen.

**Kooperation beim Lernen** Durch das Mischen der Paare in der Einarbeitungsphase wird die Kooperation beim Lernen unter den Studierenden gestärkt. Dies stärkt wiederum das Bewusstsein dafür, dass sie sich alle zusammen mit den Herausforderungen auseinandersetzen. Da der Fortschritt in einem neuen Paar meistens nicht übereinstimmt, ergibt sich daraus eine Situation, in der ein(e) weiter fortgeschrittene(r) Studierende(r) als Lehrende(r) auftritt und so das zuvor Gelernte festigen kann. Dies gleicht den Fortschritt der Studierenden an und verhindert das Abhängen einzelner. Das eingesetzte und gemeinsam gepflegte Wiki verstärkt den kooperativen Faktor beim Lernen zusätzlich. In der Softwareentwicklungsphase bilden alle Studierenden ein Team, welches gemeinsam eine Applikation entwickelt. Da wir die Studierenden auf die einzelnen Softwarekomponenten aufteilen, die nur in Zusammenarbeit die Funktionen der Applikation erfüllen, müssen die Studierenden miteinander kooperieren, um das Gesamtziel des Projekts zu erreichen. Dadurch sind die Studierenden stets dazu bereit, sich auch Schwierigkeiten bei der Entwicklung anderer Komponenten anzuhören und Lösungen zu entwickeln.

**Expertenbildung** Die Studierenden müssen sich in der Einarbeitungsphase in eine Technik oder ein Werkzeug einarbeiten und diese/dieses ihren Kommilitonen präsentieren. Im weiteren Verlauf des Lehrprojekts fungieren sie als Experten dafür und stehen ihren Kommilitonen als Ansprechpartner zur Verfügung. Dies spiegelt nicht nur die Gegebenheiten eines Projekts in der Wirtschaft wieder, sondern rückt die Studierenden in die Rolle eines Lehren-

den und gibt ihnen somit ein stärkeres Verantwortungsbewusstsein für den Erfolg des gesamten Projekts.

**Arbeitsprozessorientiertes Lernen** Ein gemeinsames Verständnis über das Vorgehen in der Softwareentwicklungsphase ist wichtig für eine erfolgreiche Zusammenarbeit. Deshalb legen wir bereits in der Einarbeitungsphase großen Wert darauf, dass die Studierenden einheitliche Arbeitsprozesse verinnerlichen. Dazu tragen zum einen die prozessorientierten Aufgabenblätter und zum anderen die Verwendung des Issue-Systems bei. Das arbeitsprozessorientierte Lernen führt außerdem dazu, dass die Studierenden die Werkzeuge nicht getrennt voneinander erlernen, sondern wie diese ineinandergreifen und somit ihren Platz in der sog. Tool-Chain kennen lernen.

**Projektlernen** Das Lernen im Rahmen eines konkreten Auftrags, der in der Form eines Projekts umgesetzt wird, ist in unseren Lehrprojekten inhärent. Die Anforderungen an das Projekt kommen dabei jedoch nicht von einem reellen Kunden, sondern von den Betreuenden und Studierenden. Diese Anforderungen beschränken sich nicht nur auf Funktionalitäten der Applikation, sondern beinhalten auch softwaretechnische, organisatorische und soziale Anforderungen. Dadurch, dass wir die Studierenden laufend auf kritische Aufgaben und Zeitpunkte hinweisen, lernen sie ihre Aufgaben sinnvoll zu priorisieren und welche Aufgaben voneinander abhängen bzw. parallelisiert werden können. Durch den verteilten Projektcharakter lernen sie außerdem wie bedeutend es ist, dass Zuständigkeiten und Entscheidungen explizit dokumentiert werden.

**Lernen mit erfahrenen Softwareentwicklern** Wir (als erfahrene Paose-Softwareentwickler) haben sehr gute Erfahrungen damit gemacht, gemeinsam mit den Studierenden eine Applikation zu entwickeln. Die Studierenden können Arbeitsweisen übernehmen, lernen Tricks, die man nur lernen kann, wenn man eng (z. B. im Paar) zusammenarbeitet, und haben jederzeit Ansprechpartner, die sich mit der in der Entwicklung befindlichen Applikation auskennen. Betreuende, die nicht an der Entwicklung teilnehmen, können zwar bei konzeptionellen und konkreten technischen Herausforderungen unterstützen, jedoch kaum Fragen beantworten, die auf Details der Applikation abzielen.

**Retrospektive** Die individuellen Gespräche zwischen Studierenden und Betreuenden, in denen Zeit für die Retrospektive eingeräumt wird, tragen

erheblich zum Lernerfolg der Studierenden bei. Es ist wichtig, dass in diesen Gesprächen keine Vorwürfe oder Schuldzuweisungen gemacht werden, sondern an das Verständnis für Entscheidungen appelliert wird und lediglich Vorschläge gemacht werden, wie bestimmte Situationen hätten besser gelöst werden können.

Diese Lehr- und Lernkonzepte können gemeinsam oder auch einzeln auf andere Lehrprojekte übertragen und in diesen eingesetzt werden. Von zentraler Bedeutung sind die Verankerung des Teamgedankens und das tiefgehende Verständnis der Betreuenden für nebenläufige, verteilte Abläufe in sozialen und organisatorischen Kontexten, die sich wechselseitig stark beeinflussen.

## 5 Schlussbetrachtung

Anhand der von uns veranstalteten praktischen Lehrprojekte in der Universität präsentieren wir in diesem Beitrag Herausforderungen und Lösungsvorschläge in der Form eines Lehr- und Lernkonzepts. Wir schildern, wie wir unsere Lehrprojekte strukturieren und die auftretenden Herausforderungen mit gezieltem Vorgehen und Werkzeugeinsatz lösen. In diesem Zuge bieten wir eine Übersicht einiger eingesetzter Lehr- und Lernkonzepte und beschreiben deren Effekte.

Wir versuchen unsere Lehrprojekte kontinuierlich zu verbessern und fokussieren die Entwicklung und den Einsatz weiterer Werkzeuge, die den Projektablauf ganzheitlichen unterstützen. Dazu zählt z. B. eine prozessorientierte Managementumgebung, die die Studierenden in der Priorisierung ihrer Aufgaben unterstützt, indem sie die Abhängigkeiten zwischen Studierenden auf einer detaillierteren Ebene visuell darstellt.

## Literaturverzeichnis

[Ba10] Balzert, H.; Koschke, R.; Lämmel, U.; Liggesmeyer, P.; Quante, J.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Verlag, 2010.

[Ca07] Cabac, L.: Multi-Agent System: A Guiding Metaphor for the Organization of Software Development Projects. In (Petta, P. Ed.): MATES 2007. Band 4687 in LNCS, Springer-Verlag, Leipzig, Germany, S. 1–12, 2007.

[Ca10] Cabac, L.: Modeling Petri Net-Based Multi-Agent Applications. Logos Verlag, Berlin, 2010.

[CHM16] Cabac, L.; Haustermann, M.; Mosteller, D.: Renew 2.5 – Towards a Comprehensive Integrated Development Environment for Petri Net-Based Applications. In (Kordon, F.; Moldt, D. Eds.): PETRI NETS 2016. Proceedings. Band 9698 in LNCS. Springer-Verlag, S. 101–112, 2016.

[Ku02] Kummer, O.: Referenznetze. Logos Verlag, Berlin, 2002.

[Sc16] Schmitz, D.; Moldt, D.; Cabac, L.; Mosteller, D.; Haustermann, M.: Utilizing Petri Nets for Teaching in Practical Courses on Collaborative Software Engineering. In Desel, J.; Yakovlev, A. Ed.): ACSD 2016. IEEE Computer Society, S. 74–83, 2016.

[Sc18] Schmitz, D.; Moldt, D.; Haustermann, M.; Mosteller, D.; Röder, C.: Team Coordination Based on Causal Nets with Synchronous Channels. In (Chatain, T.; Grosu, R. Eds.): ACSD 2018, IEEE Computer Society, S. 60–69, 2018.

# Förderung überfachlicher Kompetenzen in praktischen Software-Engineering-Veranstaltungen der RWTH Aachen

**René Röpke, Kathrin Larisch und Ulrik Schroeder**
RWTH Aachen, Informatik 9 (Learning Technologies)
Ahornstraße 55
52074 Aachen,
roepke@informatik.rwth-aachen.de
larisch@informatik.rwth-aachen.de
schroeder@informatik.rwth-aachen.de

**Abstract:** Im Rahmen eines Informatikstudiums wird neben theoretischen Grundlagen und Programmierfähigkeiten auch gezielt vermittelt, wie moderne Software in der Praxis entwickelt wird. Dabei wird oftmals eine Form der Projektarbeit gewählt, um Studierenden möglichst realitätsnahe Erfahrungen zu ermöglichen. Die Studierenden entwickeln einzeln oder in kleineren Teams Softwareprodukte für ausgewählte Problemstellungen. Neben fachlichen Inhalte stehen durch gruppendynamische Prozesse auch überfachliche Kompetenzen im Fokus. Dieser Beitrag präsentiert eine Interviewstudie mit Dozierenden von Softwareprojektpraktika an der RWTH Aachen und konzentriert sich auf die Ausgestaltung der Veranstaltungen sowie Förderung von überfachlichen Kompetenzen nach einem Kompetenzprofil für Softwareingenieure.

**Keywords:** Kompetenzen, Fertigkeiten, Software Engineering, Teamarbeit, Kollaboration.

## 1 Einleitung

Zur Vermittlung realitätsnaher und praktischer Erfahrungen im Informatikstudium wird oftmals die Form der Projektarbeit gewählt, d. h. Studierende entwickeln einzeln oder in kleinen Teams Softwareprodukte zu ausgewählten Problemstellungen. Im Vergleich sind es in der Industrie meist größere Teams, die über einen längeren Zeitraum ein sehr komplexes Problem bearbeiten.

Diese Komplexität in zeitlichen und personellen Dimensionen lässt sich nur bedingt auf die Lehre übertragen. Veranstaltungskonzepte wie Projekte und Praktika beabsichtigen eine möglichst authentische Simulation, wenn auch mit starken Abweichungen zur realen Arbeitswelt.

Im Rahmen von Softwareprojektpraktika im Bachelorstudiengang Informatik an der RWTH Aachen werden neben Fachwissen auch überfachliche Fertigkeiten der Studierenden gefördert. In der Modulbeschreibung werden die mit der Arbeitsteiligkeit verbundenen gruppendynamischen Effekte als Lerninhalt formuliert. Diese seien insoweit garantiert, da sich Studierende selbstständig in Gruppenarbeit organisieren und miteinander kommunizieren.[1]

Um ein besseres Verständnis von der Förderung überfachlicher Kompetenzen in praktischen Lehrveranstaltungen des Informatikstudiums zu erlangen, wurde mit Dozierenden, die ein Softwareprojektpraktikum im Wintersemester 2017/2018 anbieten, eine Interviewstudie durchgeführt. Hierbei wurde der Fokus auf die Struktur der Veranstaltung und die Erwartungen an die Studierenden gelegt. Im Weiteren wurden Auszüge eines Kompetenzmodells für Softwareingenieure präsentiert und diskutiert.

Die Struktur des Beitrags ist wie folgt: Abschnitt 2 präsentiert inhaltlich verwandte Arbeiten. In Abschnitt 3 wird das Studiendesign inkl. Durchführung beschrieben. Abschnitt 4 führt die Ergebnisse auf, während in Abschnitt 5 ein Ausblick gegeben wird.

## 2 Verwandte Arbeiten

In diesem Abschnitt werden praktische Lehrveranstaltungen im Informatikstudium betrachtet und es wird Bezug auf Studienprogramme anderer Universitäten genommen. Außerdem wird ein Kompetenzprofil für Softwareingenieure angeführt, welches im weiteren Verlauf als Bezugsmodell verwendet wird.

---

1 https://www.campus.rwth-aachen.de/my/rwth/all/abstractModule.asp?gguid=0x05EFB7DE0B830542A83B0 0CBC3CCE84E, zugegriffen am 19.06.2018

## 2.1 Praktische Lehrveranstaltungen im Informatikstudium

An der RWTH Aachen gibt es neben dem SPP, welches meist im fünften Semester des Bachelorstudiums abgelegt wird und mit 6 ECTS bewertet wird, das Praktikum Systemprogrammierung mit gleichem Umfang. Dieses Praktikum wird in der Regel im dritten Semester belegt. Weitere rein praktische Lehrveranstaltungen sind im Bachelorstudium nicht vorhanden[2].

Die TH Köln bietet mit ebenfalls zwei praktischen Veranstaltungen einen ähnlichen Rahmen. Im Gegensatz zur RWTH Aachen haben das sogenannte Informatikprojekt und das Praxisprojekt mit 10 bzw. 15 ECTS einen deutlich größeren Umfang. Weiterhin bietet sie die Möglichkeit eines integrierten Praxissemesters im 4. Fachsemester[3].

Auch an der Heinrich-Heine-Universität Düsseldorf findet sich das gleiche Prinzip wieder. Dort gibt es zwei praktische Veranstaltungen im Umfang von je 8 ECTS, die bereits im zweiten und dritten Semester abgelegt werden können. Das erste Praktikum konzentriert sich auf die individuellen Programmierfähigkeiten, während das zweite explizit Teamarbeit fordert[4].

Der Studiengang Softwaretechnik der Universität Stuttgart legt seinen Schwerpunkt bewusst anders als die Informatikstudiengänge. Wieder werden zwei kleinere praktische Einheiten im Umfang von 6 ECTS abgelegt. Zusätzlich gibt es ein Studienprojekt (18 ECTS), in dem Studierendengruppen ein größeres Softwareprojekt entwickeln[5].

In allen betrachteten Universitäten enthält das Studienfach Informatik praktische Veranstaltungen. Der Umfang schwankt abhängig von der Hochschule leicht, doch meist ist explizit ein Softwareprojekt, das in einem Team bearbeitet wird, genannt.

---

2 http://www.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaazpbwy, zugegriffen am 19.06.2018

3 https://www.th-koeln.de/studium/informatik-bachelor--inhalte_3487.php, zugegriffen am 19.06.2018

4 http://www.cs.hhu.de/studium-lehre-informatik/studierende/ba-po2016.html, zugegriffen am 19.06.2018

5 https://www.informatik.uni-stuttgart.de/studium/interessierte/bsc-studiengaenge/softwaretechnik, zugegriffen am 13.11.2014

## 2.2 Softwareprojekte im Informatikstudium

Der Bereich der Softwareentwicklung beschäftigt sich mit der Entwicklung von komplexen Software-basierten Systemen für unterschiedliche Problemstellungen. Oft wird in einem großen, interdisziplinären Team und über längere Zeiträume gearbeitet. Durch die sehr unterschiedlichen Aufgaben im Rahmen eines Softwareprojektes gibt es viele verschiedene Rollen, welche professionell zusammenarbeiten müssen.

Sedelmaier und Landes führen an, dass die Komplexität von Softwareprojekten im Informatikstudium nur bedingt und in Teilen abbildbar ist [SL15]. Gruppengrößen sind oftmals kleiner und der Zeitraum deutlich kürzer, da die Projektarbeiten im Rahmen einer Lehrveranstaltung absolviert werden. Während Industrieprojekte mehrere Monate bis Jahre dauern, ist das Zeitfenster in universitären Veranstaltungen nur drei bis sechs Monate, i. d. R. nicht länger als ein Semester. Daraus ergeben sich andere Anforderungen und Herausforderungen als bei Projekten in der Industrie [SL15]. Sie zeigen außerdem auf, dass Studierende im Rahmen ihres Studiums erst eine fachliche Grundbildung erwerben müssen, bevor Softwareentwicklungsprojekte durchgeführt werden können.

Für Marques und Ochoa ist die Arbeit im Team eine der wichtigsten Kompetenzen für die erfolgreiche Softwareentwicklung. Die Teamarbeit selbst wird wiederum von der Kommunikation, Koordination und Motivation innerhalb der Gruppe beeinflusst. Studierende können diese Kompetenz nicht in einer Vorlesung erwerben, sondern nur in konkreten praktischen Aufgaben, in denen Gruppenarbeit gefordert wird. Marques und Ochoa führen weiterhin aus, dass die Softwareentwicklung in mindestens zwei bis drei Kursen, die sich beispielsweise auf verschiedene Phasen des Entwicklungsprozesses konzentrieren können, geübt werden muss [MO14].

## 2.3 Kompetenzprofil für Softwareingenieure

Während viele Publikationen versuchen einen „guten“ Softwareingenieur zu charakterisieren und mögliche Fähigkeiten zu ermitteln, die ebendiese beherrschen sollen, gibt es wenige auf theoretischen Modellen basierende Ansätze zur Beschreibung der Fähigkeiten und Kompetenzen eines Softwareingenieurs. Zwar gibt es fachliche Profile und Anforderungen, die oft auch aus der Industrie beeinflusst sind, doch mit Fokus auf überfachliche Kompetenzen fehlte lange Zeit ein Modell oder Ansatz. Aufbauend auf dem SWEBOK –

„Guide to the Software Engineering Book of Knowledge, eine Handreichung im Software Engineering“ [Bo14], das als Standard anerkannt ist, entwickelten Sedelmaier und Landes ein Kompetenzprofil [Se16, SL15]. Das Profil „Software Engineering Body of Skills“ (SWEBOS) erweitert den SWEBOK um die besondere Berücksichtigung kontextsensitiver überfachlicher Kompetenzen. Die theoretisch fundierte Arbeit basiert auf der Grounded Theory und wurde in unterschiedlichen Lehrveranstaltungen evaluiert. Sedelmaier und Landes veröffentlichten ihr Vorgehen, die Methodik und erzielte Ergebnisse in [SL15]. Wesentlicher Inhalt des SWEBOS sind sechs Kompetenzfelder, welche kontextsensitive überfachliche Kompetenzen zusammenfassen. Diese Kompetenzfelder enthalten neben der Benennung der Kompetenz auf relativ abstraktem Niveau eine Beschreibung und mehrere Präzisierungen, welche als Handlungsanker konkrete Handlungen mit der Kompetenz verbinden sollen. Eine genaue Ausführung dieser Felder ist in [SL15] einzusehen. Die im Folgenden vorgestellte Studie basiert auf Auszügen des Kompetenzprofils. Es wurde ein Fokus auf Kollaboration und Softwareentwicklung im Team gelegt und entsprechende Kompetenzfelder und Präzisierungen verwendet und in den Interviews thematisiert.

## 3 Studiendesign und Durchführung

Für die Erhebung der aktuellen Bedingungen in Software-Projektpraktika (SPP) im Bachelorstudium Informatik an der RWTH Aachen wurde eine Interviewstudie mit den Dozierenden ebensolcher Veranstaltungen durchgeführt. Es wurden dafür sowohl Professoren als auch wissenschaftliche Mitarbeiter in Betracht gezogen.

Da oftmals unterschiedliche Rollen im Rahmen der Veranstaltungsorganisation existieren, wurde zu Beginn des Interviews gefragt, ob sich die Probanden die Rolle des formal Verantwortlichen oder des Betreuenden zuordnen. So können später die Antworten anhand der Rollen besser eingestuft werden. Es ist z. B. zu erwarten, dass Betreuende durch die Anleitung und Begleitung der Studierenden angemessenere Antworten auf manche Fragen geben als formal Verantwortliche.

Inhaltlich wurde im Interview zum einen auf die Veranstaltung eines Dozierenden eingegangen sowie auf die Studierenden. Zudem wurden die Probanden bezüglich fachlicher und überfachlicher Kompetenzen in der Softwareentwicklung befragt.

Das Interview beginnt mit einer kurzen Einführung in die Thematik und Absichten des Interviews. Der Aufbau des Interviews lässt sich in mehrere Teile einteilen. Im ersten Teil wurden die Probanden zu Ihrer Person und ihrer Lehrerfahrung befragt. Anschließend wurde über den Aufbau und die Struktur der Veranstaltung gesprochen und es wurden die Teilnehmeranzahl, Arbeitsform, Aufgaben und Ziele ermittelt. Offene Fragen ermöglichten es auf die Dozierenden und ihre Veranstaltung gezielt einzugehen. Geschlossene Fragen bezüglich der Erwartungserfüllung der bisherigen Studierenden sowie zum vorhanden Fachwissen forderten die Probanden auf eine Einteilung in die Kategorien „sehr schlecht“, „schlecht“, „gut“ und „sehr gut“ vorzunehmen.

Im nächsten Teil des Interviews wurden die Probanden bezüglich Kompetenzen in der Softwareentwicklung befragt. Hierbei wurden Fragen ohne expliziten Bezug zur Veranstaltung des Probanden gestellt. Erst wurde gefragt, welche Kompetenzen Studierende in der Softwareentwicklung erwerben sollen. Außerdem wurden die Probanden gefragt, wie sie Kollaboration in der Softwareentwicklung und in ihrer Veranstaltung einschätzen. Dazu wurde eine 6-stufige Likert-Skala („sehr unwichtig“, „unwichtig“, „eher unwichtig“, „eher wichtig“, „wichtig“, „sehr wichtig“) verwendet.

Im letzten Teil, welcher aus vier Bereichen besteht, werden die Probanden zu vier Kompetenzfeldern befragt. Jeweils durch eine kurze Beschreibung des Kompetenzfeldes eingeleitet, werden die Probanden mit drei offenen und drei geschlossenen Fragen konfrontiert. Erst sollen die Erwartungen an die Studierenden formuliert werden. Als nächstes wird gefragt, welche Kompetenzen gefördert werden sollen und mit der dritten Frage sollen die Probanden erläutern, wie sie ggf. die zuvor genannten Kompetenzen fördern. Diese Frage zielt auf spezielle Methoden und Ansätze, die die Dozierenden im Rahmen ihrer Veranstaltung verwenden, um eben gewisse überfachliche Kompetenzen zu fördern. Die drei weiteren Fragen orientieren sich an den im Kompetenzprofil von Sedelmaier und Landes definierten Items bzw. Präzisierungen. Die Probanden wurden gebeten die Studierenden entsprechend der zuvor genannten vierteiligen Skala prozentual einzuteilen.

Die ausgewählten Kompetenzfelder aus dem Kompetenzprofil von Sedelmaier und Landes sind Kompetenzen für die professionelle Zusammenarbeit mit anderen Menschen, Kommunikative Kompetenzen sowie die Fähigkeit, komplexe Vorgänge und Systeme sowie Zusammenhänge zu verstehen (Problembewusstsein), und die Fähigkeit, das eigene Wissen und Können, die eigenen Kompetenzen auf konkrete, neue Situationen flexibel und kreativ anzuwenden (Lösungskompetenz).

Gerade diese Kompetenzfelder geben klare Hinweise auf Kollaboration und Zusammenarbeit. So sind Kommunikative Kompetenzen und Kompetenzen zur professionellen Zusammenarbeit essentiell, aber auch das Problembewusstsein und die Lösungskompetenz sind im Kontext kollaborativer Softwareentwicklung bedeutend.

## 4 Ergebnisse

Es wurden im Rahmen der Interviewstudie Probanden der Informatik an der RWTH Aachen befragt (5 von 6 möglichen Dozierenden). Die Probanden waren sowohl wissenschaftliche Mitarbeiter als auch Professoren. In ihrer Rolle identifizierten sich zwei als formal Verantwortliche und drei als Organisatoren. Jedoch bestätigten alle Probanden eine aktive Beteiligung an der Ausgestaltung der Veranstaltungen. Vier Probanden hatten mehr als neun Jahre Lehrerfahrung und haben bereits mehrere SPP oder ähnliche Veranstaltungen angeboten.

Alle Probanden erwarteten grundlegende Kenntnisse in Programmierung. Fachspezifische Erwartungen wurde aufgrund der frühen Verortung des SPP im Studium nicht geäußert. Studierende haben zum Zeitpunkt des SPP nur wenige bis keine Wahlpflichtveranstaltungen besuchen und somit keine speziellen Kenntnisse erwerben können. Ein Proband äußerte Erfahrungen in Teamarbeit und Einarbeitung in neue Themengebiete. Diese Erwartungen an die Studierenden kann aufgrund eines Proseminars, welches Studierende zuvor im Studium belegen sollen, gestellt werden.

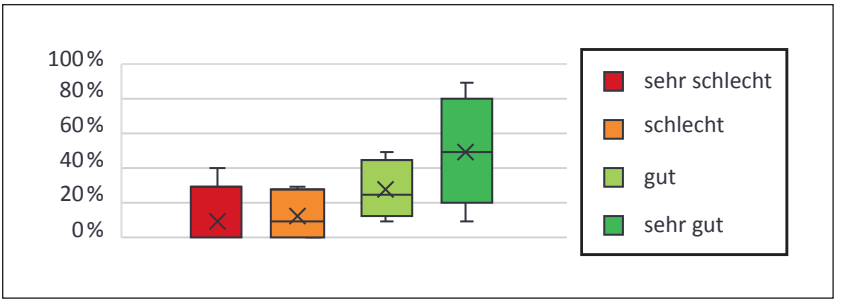


Abb. 1: Inwiefern haben bisherige Teilnehmende insgesamt Ihre Erwartungen erfüllt?

In Bezug auf die Erfüllung der Erwartungen (siehe Abbildung 1) spalten sich die Ansichten der Probanden. Ein Proband gab an, dass 90 % der Studierenden die Erwartungen *sehr gut* erfüllen, ein anderer sagte, dass es nicht mehr als 10 % sind, und 70 % *schlecht* bis *sehr schlecht* sind. Die Relevanz des SPP für das weitere Studium schätzten alle Probanden mindestens *eher hoch* ein. Die Relevanz für das zukünftige Arbeiten in der Softwareentwicklung wurde sogar *hoch* bis *sehr hoch* eingestuft. Auf die Frage, wie wichtig Kollaboration in der Veranstaltung und in der Softwareentwicklung sei, sagten drei Probanden *wichtig* und zwei Probanden *sehr wichtig.*

Alle Probanden gaben Gruppenarbeit mit variierender Gruppengröße als Arbeitsform an. Die Gruppenarbeit wurde gewählt, um verschiedene „Aspekte der Softwareentwicklung zu erleben". Teamarbeit war bei allen Probanden das Kernargument. Einzelnennungen umfassen die Förderung von Kommunikations- oder Organisationsfähigkeiten.

Den Probanden fiel es schwer, überfachliche Kompetenzen zu benennen, die Studierende im Bereich der Softwareentwicklung erwerben sollen. Es wurden Kommunikation und Koordination genannt, aber es konnten keine konkreten Kompetenzen benannt werden. Dies lässt sich dadurch erklären, dass alle Probanden nur wenig mit Kompetenzbegriffen arbeiten und sich im Arbeitsalltag auf Fachinhalte konzentrieren. Modulbeschreibungen nennen meist Kompetenzen, doch sind diese oft fachliche und eben nicht überfachliche Kompetenzen wie z. B. im Profil von Sedelmaier und Landes [SL15].

In nächsten Teil des Interviews wurden die Probanden an vier Kompetenzfelder von Sedelmaier und Landes [SL15] herangeführt. Es wurde eine kurze Erklärung gegeben und anschließend je drei Fragen mit offenen und geschlossenem Antwortformat gestellt. Es wurde jeweils gefragt, welche Fähigkeiten oder Kompetenzen dieses Kompetenzfeldes erwartet werden, welche gefördert werden sollen und wie die genannten Fähigkeiten oder Kompetenzen gefördert werden. Die drei geschlossenen Fragen sind dann je drei Präzisierungen zur Erfassung und Messung der Kompetenzen.

Zur Förderung des Problembewusstseins möchten die Probanden im Rahmen ihrer SPP abstraktes Denken fördern. Die Studierenden sollen lernen Probleme zu erfassen und zu verstehen. Abstraktions- und Modellierungsfähigkeiten sind hier von enormer Bedeutung. Dazu stoßen Probanden den Austausch in Diskussionen an und begleiten die Projektgruppen mittels Rückmeldung. Ein Proband gab an, dass 100 % der Studierenden sehr gute Abstraktions- und Modellierungsfähigkeiten haben. Die verbleibenden Probanden schätzten ein, dass nicht mehr als 18,3 % sehr gute Fähigkeiten haben.

Alle Probanden erwarten, dass Studierende in der Lage sind Probleme zu abstrahieren, um Lösungsideen zu entwickeln. Mit Blick auf das Kompetenzfeld zur Lösungskompetenz wollen die Probanden stärker fachliche Kompetenzen fördern, sehen aber darin auch einen Mehrwert für überfachliche Kompetenzen. Studierende sollen Grundlagenwissen mit Problemen verknüpfen und Lösungsideen entwickeln lernen. Hierbei werden oft Arbeitsaufträge mit offenen Aufgaben gewählt, um nötigen Spielraum für Lösungsideen und deren Vergleich zu ermöglichen.

Die kommunikativen Kompetenzen der Studierenden werden von allen Probanden maßgeblich vorausgesetzt. Studierende sollen in der Lage sein, ihre Ideen offen zu diskutieren, sich im Team abzusprechen und sich angemessen auszudrücken. Gerade die Absprachen im Team und die Diskussionsfähigkeiten der Studierenden sollen im Rahmen des SPP weiter gefördert werden. Probanden implementieren dafür Präsentationen mit Raum für Diskussionen oder folgen einer Projektmanagementmethode mit regelmäßigen persönlichen Treffen.

Zur Förderung professioneller Zusammenarbeit möchten Probanden in ihren SPP die Koordinationsfähigkeiten und Abstimmungsfähigkeiten fördern. Hierzu werden sehr große Aufgaben formuliert, welche nur im Team mit Arbeitsteilung bearbeitbar sind. Die alleinige Bearbeitung solcher Aufgaben ist zwar nicht unmöglich, jedoch deutlich schwieriger und alle Probanden bestehen auf Gruppenarbeit in ihren Veranstaltungen.

Die aggregierten, gemittelten Ergebnisse für die einzelnen Kompetenzfelder sind in Abbildung 2 zu sehen. Es ist erkennbar, dass nach Einschätzung der Probanden die Studierenden im *Problembewusstsein* am schlechtesten sind (sehr schlecht = 9,33 %, schlecht = 19,33 %, summiert 28,66 %). Im Kompetenzfeld *Lösungskompetenz* sind die Studierenden jedoch am besten (schlecht bis sehr schlecht, summiert: 14,67 %; gut bis sehr gut, summiert: 85,33 %). Die beiden verbleibenden Kompetenzfelder zu *Kommunikativen Kompetenzen* und *Kompetenzen zur professionellen Zusammenarbeit* sind ähnlich ausgeprägt (sehr schlecht bis schlecht, summiert: 20 % bzw. 20,67 %).

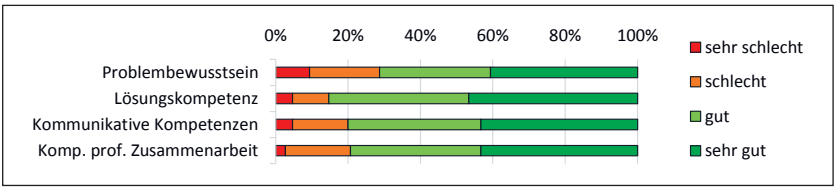


Abb. 2: Durchschnittliche Verteilung pro Kompetenzfeld

## 5 Fazit und Ausblick

Die in diesem Beitrag vorgestellte Studie zu überfachlichen Kompetenzen in Softwareprojektpraktika basiert auf Auszügen eines Kompetenzprofils entwickelt durch Sedelmaier und Landes [SL15]. Die Probanden schätzten die Relevanz ihrer Veranstaltung für das Studium und das spätere Arbeiten in der Softwareentwicklung als *hoch* bis *sehr hoch* ein. Zwar unterscheiden sich die verschiedenen SPP fachlich stark, doch versuchen Dozierende mit ähnlichen Methoden die Kompetenzen – fachlich oder überfachlich – entsprechend zu fördern. Offene Aufgabenstellungen mit ausreichender Komplexität, die in Gruppen von drei bis sieben Studierenden bearbeitet werden sollen, dienen zur Förderung von Teamarbeit, Problembewusstsein und Lösungskompetenz. Auch kommunikative Kompetenzen sollen durch Präsentationen, Diskussion und die Koordination der Gruppenarbeit gefördert werden.

Aufgrund der Stichprobengröße sind diese Ergebnisse nicht verallgemeinerbar oder übertragbar. Weitere Interviews mit Dozierenden anderer Semester oder sogar anderer Universitäten wären denkbar, um die Stichprobengröße zu erhöhen und somit das Potential der Auswertung zu verbessern. Durch die Verankerung von praktischen Softwareentwicklungsprojekten in vielen Informatikstudiengängen lassen sich potentiell Parallelen zwischen Veranstaltungen verschiedener Hochschulen ziehen. Die Interviewstudie kann somit auch mit einer größeren Stichprobe durchgeführt werden.

Schlussendlich zeigt diese Arbeit, dass überfachliche Kompetenzförderung in praktischen Lehrveranstaltungen nicht bewusst und mittels expliziter Methoden durchgeführt wird. Dies kann zum einen daran liegen, dass Informatikdozierende in ihrem Arbeitsalltag deutlich größeres Interesse an der Förderung fachlicher Kompetenzen haben, oder aber an der Unwissenheit über die für die Softwareentwicklung interessanten überfachlichen Kompetenzen.

## Literaturverzeichnis


[Bo14] Bourque, P.; Fairley, R. E.; Abran, A.; Garbajosa, J.; Keeni, G.; Shen, B.; April, A.: Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, 2014.

[MO14] Marques, M.; Ochoa, S. F.: „Improving teamwork in students software projects“, in Software Engineering Education and Training (CSEE&T), 2014 IEEE 27th Conference on, 2014, S. 99–108.

[Se16] Sedelmaier, Y.: Interdisziplinäre Fachdidaktik für Software Engineering: forschungsbasierte Entwicklung und Evaluation eines anwendungsbezogenen didaktischen Ansatzes. Dissertation, Universität Bamberg, 2016.

[SL15] Sedelmaier, Y.; Landes, D.: Überfachliche Kompetenz im Software Engineering. In (Riegel, U. et al. Hrsg.): Kompetenzmodellierung und Kompetenzmessung in den Fachdidaktiken. Waxmann Verlag GmbH, Münster, S. 111–127, 2015.

# Konsistente Lehr-Lern-Prozesse in der dualen IT-Ausbildung

**Claudia Keverpütz und Bastian Küppers**
RWTH Aachen University, IT Center
Seffenter Weg 23
52074 Aachen
keverpuetz@itc.rwth-aachen.de
kueppers@itc.rwth-aachen.de

**Abstract:** In der dualen IT-Ausbildung als Verbindung von beruflicher und akademischer Qualifikation werden die berufstypischen Werkzeuge, wie z. B. Laptops, ebenso in den Lehr-Lern-Prozessen der akademischen Unterrichtseinheiten eingesetzt. Im Prüfungswesen wird oft auf klassische Papierklausuren zurückgegriffen. Unterrichtseinheiten mit hohem Blended-Learning-Anteil ohne E-Prüfung werden dabei als „nicht konsistent" wahrgenommen. In diesem Artikel wird eine empirische Studie dargelegt, die untersucht, welche Einflüsse aus der persönlichen Lernbiografie bei den Lehrenden in einer dualen IT-Ausbildung dazu führen können, die Möglichkeiten eines E-Assessments als summative Modulprüfung anzunehmen oder abzulehnen. Beispielhaft wurden in der dargelegten Studie Interviews mit Dozenten geführt und diese hinsichtlich der Verbindung zwischen Lernbiografie, Gestaltung der Didaktik der Lehr-Lern-Prozesse, Zufriedenheit und Veränderungsbereitschaft untersucht.

**Keywords:** e-Learning, e-Assessment, Blended Learning, duale IT-Ausbildung.

## 1 Einleitung

In den letzten 10 Jahren haben sich duale IT-Ausbildungen etabliert, die in enger Verzahnung von Theorie und Praxis qualifizierte Fachkräfte unter Berücksichtigung der Bedürfnisse des Marktes ausbilden, beispielsweise der Studiengang „Angewandte Mathematik und Informatik" (ehemals „Scientific Programming") an der Fachhochschule Aachen [Kü16]. Vom Beginn an sol-

len die dual Lernenden in dieser Ausbildung die typischen Werkzeuge einsetzen und im Lehr-Lern-Prozess (LLP) der Hochschullehre (HL) für zukünftige Tätigkeiten als Softwareentwickler qualifiziert werden.

Digitalisierung in der HL zeigt sich unter anderem durch E-Learning (EL) und Blended Learning (BL). Welches Maß dabei angemessen ist, speziell für Prüfungen, ist nicht final geklärt, sodass viele Hochschulen eigene Konzepte verfolgen. Die Lehrenden haben dabei eine zentrale Rolle, da sie die Didaktik der Veranstaltung und der Prüfung verantworten. Es wurde empirisch untersucht, ob bzw. wie die Lehrenden E-Prüfungen auf den Geräten der Studierenden (BYOD) in ihre LLP integrieren. Mit strukturierten Interviews wurde erforscht, ob es eine Verbindung zwischen Akzeptanz von E-Prüfungen und der Berufs- bzw. Bildungserfahrung der Lehrenden gibt und ob es weitere, bisher unbekannte Gründe dafür gibt, weiter an Prüfungsszenarien ohne elektronische Geräte festzuhalten.

## 2 Theoretische Grundlagen

In Hochschulen werden digitale Technologien eingesetzt, allerdings zeigt sich bei der Kernaufgabe HL dieser Trend oft verlangsamt. Immer noch ist „Für die Hochschule […] die Integration digitaler Technologien in die Lehre eine zentrale Aufgabe der nächsten Jahre“, die noch nicht abgeschlossen ist [Ho17]. Der Übergang von punktuell eingefügten digitalen Komponenten (*Anreicherung*), über „Blended-Learning-Ansätze, bei denen sich Präsenzphasen und digitalisierte Lernphasen systematisch ergänzen“ (*Integration*) bis hin zu reinem *Online-Lernen* ist fließend. Im Umgang mit digitalen Medien werden bei den Beteiligten neue medienbezogene Kompetenzen erwartet und gebildet. Für die Anwendung des BL gilt: „Digitale Lehre funktioniert vor allem dann, wenn Dozierende sie proaktiv einführen“ und „[d]ort, wo digitale Medien einen obligatorischen Bestandteil des Lernprozesses ausmachen, ist die Verbreitung bereits heute hoch“ [Ho17].

Die Verknüpfung von akademischem Studium mit beruflicher Ausbildung wird im Allgemeinen als „duales Studium“ bezeichnet [Kr15]. Die Dualität bedingt, dass die Gesetze, Normen und Regeln beider beteiligter Systeme – in der Regel IHK und Hochschule – auf die LLP wirken. Studiengänge an Hochschulen unterliegen dem deutschen Qualifikationsrahmen (DQR). Diese Normierung soll zur Vergleichbarkeit der Abschlüsse innerhalb Deutschlands und in Europa führen (EQR) [BBF17]. Die Lehrenden sind Angehörige der Hochschulen und für die didaktische Gestaltung der LLP sowie die Konzep-

tion und Durchführung von Prüfungen bis hin zur Korrektur verantwortlich. Die Prüfungen sind in Prüfungsszenarien gemäß den Vorgaben der RPO/PO [MIN17] vorzunehmen. Eine erfolgreich abgelegte Modulprüfung wird im Studienverlaufsplan der Studierenden mit ECTS bewertet

Die Themengruppe „Innovation in Lern- und Prüfungsszenarien" des Hochschulforums Digitalisierung hat digitale Prüfungsformate erforscht. Es wird zwischen diagnostischen, formativen und summativen Prüfungen unterschieden [Mi17]. Unter E-Assessment wird eine „elektronische Klausur" oder „E-Prüfung" als summative Prüfung am Ende eines LLP verstanden. Bei dieser Art der digitalen Prüfung werden die Aufgabenstellung und deren Erarbeitung in einem geeigneten Softwaresystem auf einem elektronischen Gerät (PC, Laptop) abgelegt. E-Assessment ohne zusätzliche Funktionen kann als Alternative zur Papierklausur verwendet werden, um die Prüfungsdokumentation in digitaler Form zu erzeugen. Zusätzliche Optionen des E-Assessments sind die Verwendung von multimedialen Elementen in der Aufgabenstellung (Audio, Video, u. a.), die Verwendung von alternativen Aufgabenstellungen in stufenförmigen oder adaptiven (sich anpassenden) Verläufen sowie eine automatisierte Auswertung für Aufgaben im Antwort-Wahl-Verfahren (Multiple Choice, MC) [Sc15]. Die Verwendung von Aufgaben als MC-, Freitextaufgaben oder einer Mischform von beidem wurde juristisch innerhalb eines Gutachtes im Projekt „E-Assessment NRW" dargelegt. Die Authentizität des Prüflings und der von ihm erarbeiteten Prüfungsleistung sowie die Datenintegrität müssen gewährleistet sein. Die juristische Absicherung des E-Assessments erfolgt im Regelwerk einer RPO oder PO: hier müssen die verfassungsrechtlichen Grundlagen für alle Phasen eines E-Assessments festgelegt werden. Als Organe sind das Justitiariat, der Datenschutzbeauftragte und der (zentrale) IT-Dienstleister der Hochschule betroffen [FGP17].

Ein LLP in der Qualifikationslehre wird als an den thematischen Lehrzielen ausgerichtetes Lehrkonzept des Lehrenden in Veranstaltungen mit den Lernenden durchgeführt. Die Konsistenz eines LLP zeigt sich dabei in folgenden Aspekten: Der Lehrende richtet sein Lehrkonzept unter Berücksichtigung des Lernstandes auf Lernziele aus. Bei dualen Bildungsinhalten werden die Kompetenzen der Lernenden durchgängig in Theorie und Praxis entwickelt, sodass Kompetenzen für den späteren beruflichen Einsatz erworben werden, was durch eine summative Prüfung nachgewiesen wird.

Das Modell des Constructive Alignment besagt, dass Lern- und Lehrmethoden, Lernergebnisse und Prüfungsmethoden in Beziehung zueinander stehen [Bi96]. Eine Veränderung der Prüfungsmethode hat somit Einflüsse auf Lern- und Lehrmethoden und Ergebnisse. Eine Veränderung des

Prüfungsszenarios kann also Chance oder Risiko bezogen auf den gemeinsamen Erfolg bedeuten, was die Entscheidung, ein Prüfungsszenario zu verändern, beeinflusst. Die Umstellung auf E-Assessment kann als Einführung einer technologischen Innovation gesehen werden, welche akzeptiert werden muss, um erfolgreich eingesetzt zu werden. Akzeptanz setzt sich nach Müller-Böling und Müller [MM86] aus Einstellungs- und Verhaltensakzeptanz zusammen, welche sich unterscheiden können. Das bedeutet, dass eine positive Einstellung zu digitalen Prüfungsszenarien vorhanden sein kann, diese aber dennoch nicht angewendet werden. Nach Rogers setzt sich ein Entscheidungsprozess hinsichtlich der Annahme oder Ablehnung einer Innovation aus den Phasen Wissen, Überzeugung, Entscheidung, Implementierung und Bestätigung zusammen [Ro03]. Dieser Prozess wird auf die vorliegende Situation übertragen: als technologische Innovation wird E-Assessment, als Nutzer und Entscheider werden die Lehrenden in der dualen IT-Ausbildung angesehen.

Die Lehrerbildungsforschung belegt seit längerem, dass Lehrende ihr Lehrhandeln nach komplexen, subjektiven Theorien ausrichten, die auf Wissen und Erfahrungen aus der eigenen Lernbiografie aufbauen. Bei erfolgreichen Lehrenden wird dabei eine hohe Stimmigkeit zwischen Subjektiver Theorie und Handeln festgestellt [Da89]. Gleichzeitig weist die Implementationsforschung gerade bei Top-Down-Strategien, welche durch externe Instanzen oder Experten initiiert werden [GP04], nach, dass diese Veränderungsprozesse langfristig und mühsam zu nachweisbaren Erfolgen führen, was hier auf die Schwierigkeit mit Veränderungen von Erfahrungswissen, Überzeugungssystemen und Handlungsroutinen zurückgeführt wird [GP04].

## 3 Untersuchungsmethodik

Die Untersuchungsmethodik war die qualitative Inhaltsanalyse (QI), da diese die Lehrenden zu Wort kommen lässt, das Gespräch aber nicht auf vorgefertigte Muster einengt. Ein Interview schafft eine entspannte Situation zwischen Interviewer und Befragten, zudem können Unklarheiten und Fragestellungen unter Beibehaltung eines roten Fadens direkt geklärt werden [CU72]. Bei gleichmäßiger Struktur ist es möglich, die Antworten der Befragten zusammenfassend nach den Regeln der QI auszuwerten [Ma00].

In dieser Studie wurde die zusammenfassende Inhaltanalyse angewendet, um eine überschaubare Zusammenfassung der Interviews zu erzeugen

[Ma00]. Grundlagen der Auswertungen sind die transkribierten Interviews. Analyseeinheiten sind sinnzusammenhängende Textpassagen der Interviews. Ziel ist die Vergleichbarkeit von Argumenten, Benennung von Absichten und Verknüpfung zu Lernbiografieerfahrungen der Lehrenden. Es wurde eine induktive Textanalyse durchgeführt, da vom Besonderen auf das Allgemeine geschlossen werden sollte. Gemäß der acht Themencluster des strukturierten Interviews wurden Kategorien und Unterkategorien bestimmt. Anschließend wurden die nach den Transkriptionsregeln verschriftlichten, transkribierten Interviews gemäß dem Ablaufmodell von Mayring bearbeitet. In Kapitel 5 wird der Bezug zu der konkreten Untersuchungssituation mit den Befragten hergestellt.

# 4 Untersuchungskontext

Der Kontext der Studie besteht aus der dualen IT-Ausbildung im Rahmen des dualen Studiengangs „Angewandte Mathematik und Informatik“ sowie der untersuchten Gruppe der Lehrenden, im weiteren Verlauf „Befragte“ genannt. Alle Befragten sind im Auftrag der Fachhochschule Aachen als Lehrbeauftragte im vorgehend genannten Studiengang tätig, männlich und unterscheiden sich in Alter, Lehrerfahrung, Fachrichtung sowie Anzahl der gelesenen Module. In der Befragung wurde ein halbstandardisiertes, strukturiertes Interview verwendet, welches auf einem Interview-Leitfaden mit acht Themenclustern basierte, welches ermitteln soll, welche Argumente für die Entscheidung der Befragten hinsichtlich einer Veränderung im Prüfungsszenario bzw. hinsichtlich der Verwirklichung der Lehrabsichten in ihrem LLP maßgeblich sind:

- Bezug des gelehrten Moduls zum Studiengang und zum Erreichen von Lehrzielen bzw. Kompetenzen der Lernenden
- Blended-Learning in der Didaktik des gelehrten Moduls
- Absichten der eingesetzten Prüfungsszenarien in Form der wahrgenommenen Vor- und Nachteile
- Erwartungshaltung zu digitalen Prüfungsszenarios (E-Assessment)
- Verwendungsabsicht eines digitalen Prüfungsszenarios
- Lernbiografie
- Einflüsse auf die Didaktik unter Berücksichtigung eigener Lernerfahrungen
- Zukunft der dualen IT-Ausbildung und der eigenen Lehre

Unter Lernbiografie wird hier die Biografie des formellen und informellen Lernens eines Menschen verstanden. Sie ist als Referenzsystem für die Erfahrungen des Lernens mit positiven und negativen Erfahrungen zu verstehen [Da89]. Diese subjektiven Faktoren beeinflussen die Entscheidung des Lehrenden hinsichtlich Didaktik und hier speziell hinsichtlich der Veränderung des Prüfungsszenarios. Als LLPs wurden in diesem Kontext alle zu einem Modul eines Befragten gehörenden didaktischen Untereinheiten verstanden. Alle Befragten entscheiden über die Didaktik ihres Moduls und den Einsatz von BL. Ebenso wird von ihnen die verwendete Prüfungsform je Semester für die summative Prüfung festgelegt. Zum Zeitpunkt der Studie können von den Befragten die in der Prüfungsordnung des Studiengangs benannten Prüfungsformen „mündliche Prüfung", „Papierklausur" und eine „hybride Prüfung" (lt. PO „Klausuren […] in multimedial gestützter Form") eingesetzt werden. Um zu ermessen, welche Einflüsse die Veränderung eines Prüfungsszenarios hin zum E-Assessment haben, wurden die vorstehend genannten Prüfungsszenarien gemäß der 12 Dimensionen, die in [Mi17] unterschieden werden, auf den Untersuchungskontext der dualen IT-Ausbildung bezogen bewertet. Betroffen sind bei Veränderungen zum E-Assessment dabei vor allem die folgenden Dimensionen:

- D4: Prüfungsarrangement
- D7: Identitätskontrolle
- D8: Dokumentation
- D9: Kosten- und Arbeitsaufwand
- D10: infrastruktureller Rahmen
- D11: Unterstützungsangebote für die Prüfungsbeteiligten
- D12: rechtlicher Rahmen und Datensicherheit

## 5 Durchführung und Diskussion der Ergebnisse

Die Interviews, als Grundlage der qualitativen Inhaltsanalyse, wurden nach Mayring [Ma00] durchgeführt, transkribiert und ausgewertet. Vor den Interviews wurden die Kerndaten der Befragten erhoben. Dabei ergaben sich drei Gruppen in Bezug auf die Dauer der Lehrerfahrung (1 × 2 Jahre, 4 × 5–7 Jahre, 1 × 14 Jahre) und zwei Gruppen in Bezug auf das Alter (30 Jahre, 50 Jahre).

Es wurde eine Einstufung der Untereinheiten jedes LLP hinsichtlich des Grades an BL vorgenommen (hoch, mittel, niedrig, kein), um den Status

der Veranstaltungen zu beschreiben und herauszuarbeiten, wo der LLP der Module als „nicht konsistent“ zwischen diesen und dem Prüfungsszenarium verstanden wird: es wird dabei zwischen Vorlesung, Übung, asynchronem Lernen, verpflichtender Hausaufgabe und Projektarbeit als Untereinheiten unterschieden.

Bei den 15 betrachteten Modulen wird in drei Fällen als Prüfung ein digitales Prüfungsszenario durchgeführt: in diesen Fällen wird die hybride Prüfung durchgeführt. Eine Modulprüfung wird als mündliche Prüfung durchgeführt, alle anderen Prüfungen sind Papierklausuren. Es wurde die Konsistenz der LLP hinsichtlich des Grades an BL betrachtet. Ein LLP wird als „nicht konsistent“ definiert, wenn im Übungsanteil des LLP ein hoher Grad BL angewendet wird und als Prüfung eine Papierklausur durchgeführt wird. Dies bedeutet, dass in den Übungen elektronische Geräte genutzt werden, um das Gelernte zu vertiefen, diese Arbeitsmittel jedoch nicht in der Prüfung zur Verfügung stehen. Gemäß der oben angegebenen Definition werden 6 von 15 LLP als „nicht konsistent“ eingestuft. Diese Einstufung betrifft vier Befragte. In 9 von 15 Modulen sind die Befragten mit dem angewendeten Prüfungsszenario zufrieden: als Gründe geben sie Aufgabenpassung bzw. Modulzielerreichung oder Einzelfallgerechtigkeit an. Unzufriedenheit zeigt sich, wenn das Prüfungsszenario als nicht passend zum Arbeitskontext der dual Lernenden empfunden wird. Aktuell wird von keinem Befragten reines E-Assessment eingesetzt, weshalb die Einstellung zu einer möglichen Veränderung der Modulprüfungen dahingehend herausgearbeitet wurde. Dabei zeigt sich die Gruppe der Befragten zweigeteilt: drei Befragte haben eine grundsätzlich positive Einstellung, die anderen Befragten haben Einwände wegen ungeklärter Fragen. Um die Handlungsakzeptanz zu ermitteln, wurden die Befragten danach gefragt, ob sie E-Assessment anwenden werden. Hierbei äußerte sich nur einer der Befragten positiv.

Hinsichtlich ihrer Lernbiografie weisen alle Befragten Hochschulabschlüsse vor. Drei der Befragten haben eine duale IT-Ausbildung mit Studium und einen nachfolgenden Masterstudiengang abgeschlossen, sie haben demnach Erfahrungen als dual Lernende. Insgesamt gaben fünf der Befragten an, dass ihnen die duale Situation des „studieren und arbeiten“ aus eigener Erfahrung bekannt ist. Alle Befragten leben aktiv die Rolle der Lernenden in unterschiedlichen Arten der Weiterbildung (Hochschullehre, Sprachen, Führungsqualifikation).

Die einzelnen Befragten benannten folgende Einflüsse der Lernbiografie auf ihr Lehrhandeln:

- Bereitstellung von schriftlichem Lernmaterial (Skript) und Fragesammlungen zum asynchronen Lernen wie es zum persönlichen erfolgreichen Lernen verwendet wurde
- Transparenz der Lernziele und Visualisierung spezieller Lehrinhalte, gemäß dem persönlichen Lern-Motto
- Streben zur Realisierung von Verknüpfungsszenarien der Theorie und Praxis in Form von Übungsslots und E-Assessment gemäß den persönlichen Lernbedürfnissen
- Bereitstellung von asynchronem Lernmaterial zur Vertiefung im persönlichen Lerntempo beim asynchronen Lernen
- Strukturierung der Veranstaltung mittels Kernfragen zur Vermittlung der persönlichen Lernstrategie
- Distanzierung von persönlichen Erlebnissen hin zu den Bedürfnissen der Zielgruppe mit auf diese Zielgruppe abgestimmten Lernanlässen

## 6 Fazit und Ausblick

Ausgehend von der Annahme, dass in dualen IT-Ausbildungen technikaffine Lernende zu Fachkräften für den IT Markt ausgebildet werden und dabei durch Digitalisierung in LLP angemessen unterstützt werden sollen, wurde in dieser Forschungsarbeit die Motivation der Lehrenden in Bezug auf Veränderung der summativen Prüfung hin zum E-Assessment erforscht.

Es zeigte sich, dass sich die Lehre im Untersuchungskontext durchaus typisch für HL gestaltet, speziell hinsichtlich des wichtigsten Ziels, dem Erreichen der Qualifikation. Es fällt auf, dass die Digitalisierung der Lehre selbst in einem MINT-Studiengang zweitrangig ist, an vielen Stellen findet klassische (Frontal-)Lehre statt. Dies bestätigt die Feststellung des Hochschulforums Digitalisierung: „Aktuell zeigt sich flächendeckend eher eine punktuelle Anreicherung der Lehre durch digitale Medien“ [Ho17]. Die Nutzung digitaler Prüfungsszenarien geht einher mit Schaffung von Infrastruktur, Klärung von Finanzen, Absicherung der Rechtsgültigkeit in RPO und PO und Unterstützung bei Abläufen. Wenn ein digitales Prüfungsszenario zum Prüfen im Studienverlauf als „sichere Prüfung“ eingesetzt werden soll, so müssen diese Bedingungen durch die Leitungsebene (Hochschule, Dekanat, Studienstandort, Ausbildungsgeber) geschaffen werden. In der Untersuchung zeigte sich weiterhin, dass Zufriedenheit des Lehrenden mit dem aktuell praktizierten LLP dazu führt, das bewährte Konzept zu wiederholen und keine Veränderung hinsichtlich eines digitalen Prüfungsszenarios anzustreben.

Zufriedenheit herrscht dann, wenn Ideen und Erfahrungen in einem Lehrkonzept realisiert werden können. Ebenso zeigte sich, dass Unzufriedenheit mit dem LLP zu großen Initiativen bei einem Befragten führt: er entwickelt gerade innerhalb seiner Promotion ein mit BYOD nutzbares E-Assessment-System.

Das Hochschulforum Digitalisierung formuliert dies im Abschlussbericht so: „Digitale Lehre funktioniert vor allem dann, wenn Dozierende sie proaktiv einführen“ [Ho17]. An anderen Stellen zeigt sich, dass Veränderungen in LLP nur langsam voranschreiten und die „Brauchbarkeit“ der Veränderung ein wichtiges Entscheidungs-Kriterium ist [GP04]. Veränderung zu digitalen Prüfungsszenarien bedeutet Aufwand für den Lehrenden. Zudem würde der Lehrende das Risiko eingehen, dass es noch nicht einkalkulierte Schwierigkeiten gibt, die das digitale Prüfungsszenario als noch nicht verwendbar einstufen lassen. Die Planung eines Semesters würde von den Umstellungsabsichten eines Prüfungsszenarios beeinflusst – unter Ressourcennot ist dies schwierig. Dagegen sind bei den dual Lernenden weniger Umstellungsprobleme zu erwarten, da das Prüfungsszenario dann eine konsequente Fortsetzung ihres Lernens bedeuten würde: sie könnten dann ihre Lernergebnisse in der veränderten Prüfungsform angemessen zeigen. Dies führt die Problematik der Anwendung von digitalen Prüfungsszenarien auf Grundmotivationen zurück: für Veränderungen muss das System bereit sein und den Einzelnen die entsprechenden Ressourcen innerhalb einer förderlichen Kultur zur Verfügung stellen. Digitalisierung muss in der Strategie der Studiengänge verankert sein – so können an unterschiedlichen Standorten gleichwertige Bedingungen für Digitalisierung geschaffen werden und Lehrende erfahren Unterstützung bei Umstellungsmaßnahmen als Wertschätzung ihrer Veränderungsbestrebungen.

## Literaturverzeichnis


[BBF17] Bundesministerium für Bildung und Forschung, „Der Deutsche Qualifikationsrahmen für lebenslanges Lernen,“ 2017. [Online]. Available: https://www.dqr.de. [Accessed 21 November 2017].

[Bi96] Biggs, J.: „Enhancing Teaching Through Constructive Alignment,“ Higher Education, vol. 32, no. 3, S. 347–364, 1996.

[CU72] Bureau of Applied Social Research, Columbia University, „Das qualitative Interview,“ In: Das Interview. Formen. Technik. Auswertung., Köln, Kiepenheuer & Witsch, 1972, S. 143–160.

[Da89] Dann, H.-D.: „Subjektive Thorien als Basis erfolgreichen Handelns von Lehrkräften“ 1989. [Online]. Available: https://www.pedocs.de/volltexte/2017/13161/pdf/BZL_1989_2_247_254.pdf. [Accessed 19 Juni 2018].

[FGP17] Forgó, N.; Graupe, S.; Pfeiffenbring, J.: „Rechtliche Aspekte von E-Assessment an Hochschulen,“ Juni 2016. https://doi.org/10.17185/duepublico/42871. [Accessed 27 November 2017].

[GP04] Gräsel, C.; Parchmann, I.: „Implementationsforschung – oder: der steinige Weg, Unterricht zu verändern“ 2004. [Online]. Available: https://www.pedocs.de/volltexte/2013/5813/pdf/UntWiss_2004_3_Graesel_Parchmann_Implementationsforschung.pdf. [Accessed 19 Juni 2018].

[Ho17] Hochschulforum_Digitalisierung, „The Digital Turn. Abschlussbericht 2016.,“ Dezember 2016. https://hochschulforumdigitalisierung.de/sites/default/files/dateien/Abschlussbericht.pdf. [Accessed 27 November 2017].

[Kr15] Krone, S.: Dual Studieren im Blick, Wiesbaden: Springer Fachmedien, 2015.

[Kü16] Küppers, B.; Dondorf, T.; Willemsen, B.; Pflug, H.J.; Vonhasselt, C.; Magrean, B.; Müller, M.S.; Bischof, C.: „The Scientific Programming Integrated Degree Program,“ Procedia Computer Science, vol. 80, S. 1–11, 2016.

[Ma00] Mayring, P.: Qualitative Inhaltanalyse – Grundlagen und Techniken, Weinheim: Beltz-Deutscher Studienverlag, 2000.

[Mi17] Michel, L.; Goertz, L.; Radomski, S.; Fritsch, T.B.L.: „Digitales Prüfen und Bewerten im Hochschulbereich,“ März 2015. https://www.che.de/downloads/HFD_Studie_DigitalesPruefen.pdf. [Accessed 30 November 2017].

[MIN17] Ministerium des Inneren des Landes Nordrhein-Westfalen, „Gesetz über die Hochschulen des Landes Nordrhein-Westfalen (Hochschulgesetz – HG),“ Ministerium für Inneres und Kommunales des Landes Nordrhein-Westfalen, 2017. https://recht.nrw.de/lmi/owa/br_text_anzeigen?v_id=10000000000000000654#det379101. [Accessed 21 November 2017].

[MM86] Müller-Böling, D.; Müller, M.: Akzeptanzfaktoren der Bürokommunikation, R. Oldenbourg, 1986, S. 25.

[Ro03] Rogers, E.: Diffusion of Innovations, vol. 5. Edition, New York: Free Press, 2003, S. 169.

[Sc15] Schmees, M.: „E-Assessments und die Qualität von Hochschullehre,“ Greifswalder Beiträge zur Hochschullehre, vol. Heft 1/2015, S. 7–23, 2015.

# Empirische Untersuchungen von Lückentext-Items zur Beherrschung der Syntax einer Programmiersprache

**Michael Striewe und Matthias Kramer**
Universität Duisburg-Essen, paluno – The Ruhr Institute
for Software Technology
Gerlingstraße 16
45127 Essen
michael.striewe@paluno.uni-due.de

Universität Duisburg-Essen, Didaktik der Informatik
Schützenbahn 70
45127 Essen
matthias.kramer@uni-due.de

**Abstract:** Lückentext-Items auf der Basis von Programmcode können eingesetzt werden, um Kenntnisse in der Syntax einer Programmiersprache zu prüfen, ohne dazu komplexe Programmieraufgaben zu stellen, deren Bearbeitung weitere Kompetenzen erfordert. Der vorliegende Beitrag dokumentiert den Einsatz von insgesamt zehn derartigen Items in einer universitären Erstsemestervorlesung zur Programmierung mit Java. Es werden sowohl Erfahrungen mit der Konstruktion der Items als auch empirische Daten aus dem Einsatz diskutiert. Der Beitrag zeigt dadurch insbesondere die Herausforderungen bei der Konstruktion valider Instrumente zur Kompetenzmessung in der Programmierausbildung auf. Die begrenzten und teilweise vorläufigen Ergebnisse zur Qualität der erzeugten Items legen trotzdem nahe, dass Erstellung und Einsatz entsprechender Items möglich ist und einen Beitrag zur Kompetenzmessung leisten kann.

**Keywords:** Programmierausbildung, Codeverständnis, C-Test, Lückentext, Kompetenzmessung, Empirische Untersuchung.

# 1 Einleitung

Beim Erlernen einer Programmiersprache stellt das Erlernen der Syntax oftmals eine Barriere für Anfänger dar [Mc01, SS13]. Da es jedoch nicht ausreichend erscheint, die Beherrschung von Syntaxelementen isoliert zu überprüfen oder aus den Lösungen für komplexe Programmieraufgaben indirekt auf die Beherrschung der Syntax zu schließen, können spezielle Lückentext-Items eingesetzt werden, die an sprachwissenschaftliche C-Test angelehnt sind [SKG17].

Der Einsatz von Lückentext-Items ist dabei grundsätzlich keine neue Idee, sondern wurde bereits in den 1980er Jahren erprobt [SEB82]. Sowohl diese als auch weitere Studien fokussieren jedoch Programmierwissen im weiteren Sinne und damit explizit mehr als die reine Syntax-Beherrschung. Die daraus resultierenden Lückentext-Items sind folglich so gestaltet, dass ein Verständnis der Semantik des vorliegenden Codes notwendig ist, um die Lücken zu schließen. Insbesondere die Ergebnisse von [De11] zu Schwierigkeiten von Anfängern legen jedoch nahe, dass korrekte Programmerstellung unabhängig vom konzeptuellen Wissensstand häufig an der Syntax scheitert. Deshalb soll im Fall von Lückentext-Items zur Beherrschung der Syntax einer Programmiersprache ein tiefgreifendes Verständnis der Semantik nicht notwendig sein. Stattdessen sollen diese Items so konstruiert sein, dass es zur Lösung jeder Lücke ausreicht, die Syntax der Programmiersprache zu beherrschen, d. h. insbesondere Schlüsselwörter zu kennen und Regeln für den Aufbau von Ausdrücken, Anweisungen und Referenzen zu beherrschen. Ein Beispiel für ein solches Item ist in Abbildung 1 zu sehen.

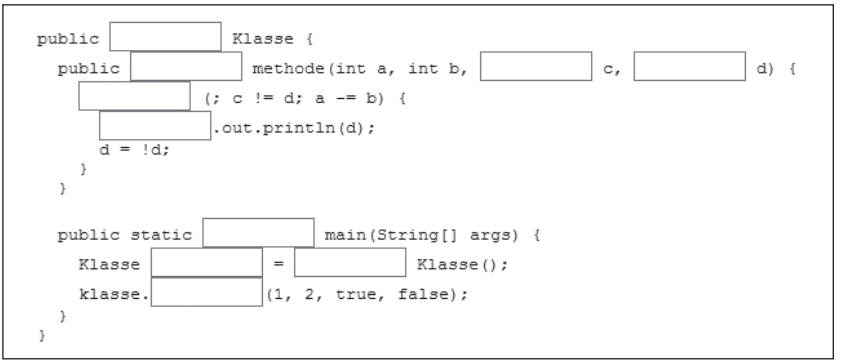


Abb. 1: Beispiel für ein Lückentext-Item in der Programmiersprache Java. In jeder Lücke fehlt genau ein Schlüsselwort oder Bezeichner

Der vorliegende Beitrag dokumentiert die Erstellung und den Einsatz von zehn solcher Items im Rahmen einer Erstsemestervorlesung zur Einführung in die Programmierung am Campus Essen der Universität Duisburg-Essen im Wintersemester 2017/18. Ziel der Diskussion ist die Beantwortung der Frage, ob die Items einen sinnvollen Beitrag zur Kompetenzmessung in der Programmierausbildung leisten können. Der Beitrag ist wie folgt gegliedert: Abschnitt 2 stellt den Forschungskontext und verwandte Arbeiten dar. Abschnitt 3 diskutiert die Erstellung der Items und Abschnitt 4 enthält die statistische Auswertung der empirisch gewonnenen Nutzungsdaten. Abschnitt 5 interpretiert diese Daten und bereitet damit das Fazit in Abschnitt 6 vor.

## 2 Forschungskontext und Related Work

Neben dem Verstehen des Programmablaufs sowie dem Entwickeln allgemeiner Problemlösungsstrategien spielt das Erlernen einer *Notation,* also der Syntax und Semantik einer Programmiersprache, eine wichtige Rolle in der Programmierausbildung [DB86], ist aber allein nicht ausreichend, um funktionierende Programme zu erstellen [Wi96]. Der korrekte Umgang mit einer Programmiersprache, d. h. Programmierbausteine syntaktisch und semantisch sinnvoll miteinander kombinieren zu können, ist daher mutmaßlich eine notwendige Teilfertigkeit, die für das erfolgreiche Erlernen von Programmierung von wesentlicher Bedeutung ist. Betrachtet man Programmierkompetenz daher als multidimensionales latentes Konstrukt [KHB16], so erscheint es sinnvoll, durch vielfältige Testitems im Rahmen eines formativen Assessments festzustellen, welche Ausprägungen in den verschiedenen Teildimensionen zu (Miss-)Erfolgen führen. Lückentext-Items als Adaption sprachwissenschaftlicher C-Tests können als Baustein in einem größeren Assessment wertvolle Hinweise auf die Leistung in Kompetenzfacetten wie *Verstehen gegebener Quelltexte* bzw. *syntaktisches Verständnis einer Programmiersprache* liefern. Zudem stellen sie durch das Format eine ideale Schnittstelle zwischen den Fertigkeiten *Quelltexte interpretieren* und *Quelltexte produzieren* dar, was ebenfalls vermuten lässt, dass durch einen solchen Test verschiedene Fertigkeiten abgeprüft werden [SKG17].

Lückentext-Items wurden allgemein in der Programmierung bereits in verschiedenen Forschungskontexten eingesetzt. Beispiele sind die Erkennung unterschiedlicher Strategien bei der Programmierung zwischen Novizen und Experten [SEB82] oder die Auswirkung von Geschlecht und Lernstil [LY09]. Im Gegensatz zum vorliegenden Beitrag leiten diese Studien aus den Ergeb-

nissen von Lückentext-Tests weitere Erkenntnisse ab, während sich der vorliegende Beitrag mit der Evaluation des Item-Typs als solchem befasst.

## 3 Item-Konstruktion

Alle zehn Items, die im Rahmen dieses Beitrags diskutiert werden, wurden manuell mit Hilfe eines Editors erzeugt, der das passende Datenformat für das verwendete Übungssystem erzeugt. Der Editor ist so gestaltet, dass ein Quellcode eingelesen wird und anschließend durch den Aufgabenautor einzelne Wörter in diesem Quellcode durch Anklicken als Lücke ausgewählt werden können. Als Wörter werden dabei primär Schlüsselworte der Programmiersprache behandelt. Im Kontext von Java ist es zudem sinnvoll, Zeichenketten wie `String`, `System` oder `println` äquivalent zu Schlüsselworten zu behandeln, da sie als vorgegebene Typen und Methoden der Java API ebenfalls eine feststehende Bedeutung unabhängig vom Programmkontext haben. Grundsätzlich erlaubt es der Editor jedoch, spezifisch pro Programmiersprache oder sogar pro Quelltext beliebige Zeichenketten als Kandidaten für Lücken zu definieren. Dadurch wird es auch möglich, beliebige Klassen- oder Methodennamen in Lücken umzuwandeln. Dies ist bei der Item-Konstruktion jedoch höchstens dann sinnvoll, wenn deren Lösung trotzdem noch rein syntaktisch aus dem Programmcode geschlossen werden kann.

Zu Beginn der Untersuchung wurde der Versuch unternommen, die Items automatisiert zu erzeugen, indem einfache Regeln für die Erzeugung der Lücken definiert wurden. Nach diesen Regeln hätte dann beispielsweise jedes zweite Schlüsselwort oder jeder dritte Bezeichner automatisch in eine Lücke umgewandelt werden können. Es stellte sich jedoch schnell heraus, dass auf diese Weise erzeugte Items häufig nicht eindeutig lösbar sind. Beispielsweise dürfen Zugriffsmodifikatoren nicht in Lücken umgewandelt werden, da nicht immer aus dem Kontext ersichtlich ist, ob die entstehende Lücke mit `public`, `private` oder `protected` zu füllen ist. Auch das Offenlassen einer solchen Lücke könnte eine syntaktisch gültige Lösung sein. An anderen Stellen könnte das Entfernen eines primitiven Datentyps eine Lücke hinterlassen, die syntaktisch sowohl mit `int` als auch mit `long` oder sowohl mit `double` als auch mit `float` gefüllt werden könnte. Während in diesen Beispielen die Lösungsmenge zumindest endlich ist, können jedoch auch Situationen entstehen, in denen unendlich viele gültige Lösungen möglich sind. Entfernt man beispielsweise den Ausdruck innerhalb einer `return`-Anweisung, kann die entstande-

ne Lücke syntaktisch korrekt mit jedem beliebigen Ausdruck gefüllt werden, der zum Rückgabetyp der Methode passt.

Da das Abfangen derartiger Sonderfälle eine enorme Komplexität in den Erstellungsregeln erfordern würde, wurde auf weitere Versuche zur automatischen Erstellung der Items verzichtet. Stattdessen wurden zehn Items durch manuelle Auswahl von Lücken erstellt. Das in Abbildung 1 gezeigte Item ist auf diese Weise mit den folgenden Begründungen für die Wahl der Lücken entstanden:

- Die erste Lücke verlangt das Schlüsselwort `class`. Dieses ist für Klassendeklarationen zwingend und aus der Zeile ist ersichtlich, dass es sich hier nicht um eine andere Deklaration (Feld oder Methode) handeln kann.
- Die zweite Lücke verlangt das Schlüsselwort `void`. Aus der Zeile ist ersichtlich, dass es sich hier um eine Methodendeklaration handelt, die eine Angabe zum Rückgabetyp benötigt. Aus der Betrachtung der folgenden Zeilen ist ersichtlich, dass die Methode keine `return`-Anweisung enthält und somit `void` die einzig richtige Angabe zum Rückgabetyp sein muss.
- Die dritte und vierte Lücke verlangen jeweils das Schlüsselwort `boolean`. Aus der Betrachtung des Methodenkörpers ist erkennbar, dass `d` vom Typ `boolean` sein muss, damit `!d` ein erlaubter Ausdruck ist. Dann muss auch `c` `boolean` sein, damit auch `c != d` ein erlaubter Ausdruck ist.
- Die fünfte Lücke verlangt das Schlüsselwort `for`. Dieses ist das einzige Schlüsselwort, das die Zeile zu einer syntaktisch gültigen Anweisung ergänzt.
- Die sechste Lücke verlangt den Bezeichner `System`. Dieser ist Teil der Standard-API von Java und wurde im Rahmen der Vorlesung eingeführt. Da der Quellcode keine weiteren Imports deklariert, kann davon ausgegangen werden, dass `System` als einziges den statischen Zugriff auf ein Feld namens `out` ermöglicht.
- Die siebte Lücke verlangt das Schlüsselwort `void`. Hier greift dieselbe Argumentation wie bei Lücke zwei.
- Die achte Lücke verlangt den Bezeichner `klasse`. Dieser wird in der folgenden Zeile verwendet und kann syntaktisch nur hier eingeführt werden.
- Die neunte Lücke verlangt das Schlüsselwort `new`. Dieses ist das einzige Schlüsselwort, das die rechte Seite der Zuweisung zu einem syntaktisch gültigen Ausdruck ergänzt.
- Die zehnte Lücke verlangt den Bezeichner `methode`. Dieser wird in Zeile 2 als Name einer Methode mit passender Signatur eingeführt. Auf dem Objekt `klasse` vom Typ `Klasse` können zwar auch Methoden der

allgemeinen Oberklasse `Object` aufgerufen werden, aber von diesen hat keine eine passende Signatur.

Es ist zu erwarten, dass die Lücken unterschiedlich schwierig zu schließen sind, da sie eine unterschiedlich umfangreiche Betrachtung des Kontexts erfordern. Damit ist ebenso zu erwarten, dass die Items insgesamt unterschiedlich schwierig zu lösen sind, da sie auf unterschiedlichen Quelltexten basieren und somit unterschiedlich schwierige Lücken enthalten.

# 4 Empirische Untersuchung und statistische Auswertung

## 4.1 Einsatzszenario und Datenerhebung

Die zehn Lückentext-Items wurden parallel zur Durchführung der Erstsemestervorlesung „Einführung in der Programmierung“ entworfen und in einem elektronischen Übungssystem zur Verfügung gestellt. Das in den Lückentexten verwendete Vokabular des Programmcodes (d. h. die Menge an verschiedenen Schlüsselwörtern und Sprachkonstrukten) orientiert sich daher am Fortgang der Vorlesung und ist bei den ersten Items geringer als bei den später erzeugten. Das erste Item wurde kurz nach Beginn der Vorlesung im Oktober freigeschaltet und bis zum Ende des Semesters von 222 Studierenden insgesamt 350 Mal bearbeitet. Das letzte Item wurde Anfang Dezember freigeschaltet und von 70 Studierenden insgesamt 85 Mal bearbeitet. Insgesamt haben 47 Personen alle 10 Items bearbeitet.

Die Bearbeitung der Items erfolgte auf freiwilliger Basis und ohne Verknüpfung mit Bonuspunkten oder sonstigen Anreizen. Als Rückmeldung erhielten die Studierenden die Information, wie viele Lücken sie korrekt ausgefüllt hatten. Musterlösungen oder Hinweise wurden nicht zur Verfügung gestellt. Alle Einreichungen wurden im elektronischen Übungssystem gespeichert und konnten für die statistische Auswertung herangezogen werden.

## 4.2 Datenaufbereitung

Für die Auswertung wurden alle Einreichungen zunächst aus dem Übungssystem exportiert und noch einmal unabhängig vom Ergebnis durch das Übungssystem automatisiert analysiert. Dabei wurden insbesondere Eingabefehler der Testpersonen insofern bereinigt, dass führende und folgende Leerzeichen bei der Feststellung der Korrektheit einer Eingabe ignoriert wurden.

Durch die automatische Analyse entstanden insbesondere zwei Arten von Tabellen: (1) Je eine *Lösungstabelle* pro Item mit allen Eingaben der Lücken und deren jeweiliger Häufigkeit. (2) Eine *Gesamttabelle* mit den Ergebnissen derjenigen Studierenden, die alle zehn Items bearbeitet haben. In dieser Tabelle ist pro Lücke dichotom kodiert, ob die Lücke korrekt gefüllt wurde oder nicht. Ferner ist in dieser Tabelle polytom kodiert, wie viele Lücken eine Testperson in jedem Item korrekt gefüllt hat. Alle Tabellen wurden in Excel gespeichert, in die Statistiksoftware R importiert und dort unter Verwendung des Pakets *psych* [Re17] weiter analysiert.

## 4.3 Deskriptive Statistik – Betrachtung auf Testebene

Von den 47 Testpersonen, die alle zehn Items bearbeitet haben, haben acht Personen die Aufgaben zur Vorbereitung auf die Nachklausur benutzt, d. h. nicht parallel zur Vorlesung sondern nach der Erstklausur bearbeitet. Alle folgenden statistischen Aussagen beziehen sich auf die 47 Personen und die Betrachtung der polytomen Werte. Aufgrund der unterschiedlichen Anzahl der Lücken pro Item (4–12) und den daraus resultierenden möglichen Punktzahlen, war ein direkter Vergleich nicht möglich. Daher erfolgte in einem ersten Schritt eine Normierung der Werte durch Teilen der erreichten Punktzahl pro Item und pro Proband durch die Anzahl der Lücken pro Item. Dadurch ergibt sich für jeden Probanden die prozentuale Korrektheit für jedes Item. Die Ergebnisse dazu sind in Tabelle 1 dargestellt.

Tab. 1: Die normierten Werte der polytomen Rohdaten für die ersten vier Probanden

| TP | LT1 | LT2 | LT3 | LT4 | LT5 | LT6 | LT7 | LT8 | LT9 | LT10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0,857 | 1,000 | 0,400 | 1,000 | 1,000 | 0,750 | 1,000 | 0,900 | 0,857 | 0,667 |
| **2** | 0,857 | 1,000 | 1,000 | 1,000 | 0,778 | 0,750 | 1,000 | 0,900 | 1,000 | 0,917 |
| **3** | 1,000 | 1,000 | 0,800 | 1,000 | 1,000 | 0,750 | 1,000 | 0,800 | 0,857 | 0,833 |
| **4** | 1,000 | 1,000 | 0,800 | 1,000 | 1,000 | 0,750 | 1,000 | 0,600 | 1,000 | 1,000 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Aus diesen Werten lassen sich nun erste deskriptivstatistische Werte durch die Boxplots zu den jeweiligen Items darstellen. Diese sind in Abbildung 2 festgehalten.

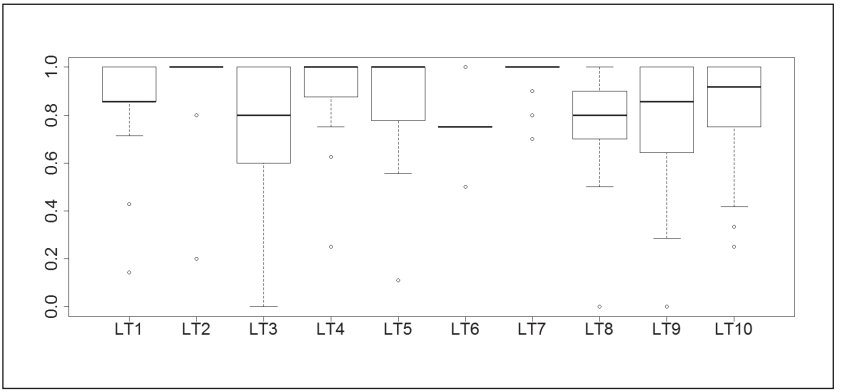


Abb. 2: Boxplot der normierten Ergebnisse. Lückentext-Item 8 (LT8) ist dasjenige, das auch in Abbildung 1 zu sehen ist

Als Maß für die testpsychologische Qualität der zehn Items im Rahmen der internen Konsistenz wurde anschließend Cronbachs Alpha berechnet und ein Wert von 0.77 erreicht. Dabei wiesen die Lückentext-Items 6 und 7 eine auffallend schlechte, tlw. sogar negative Trennschärfe (Korrelation mit dem Gesamtergebnis) auf. Dies weist auf die Notwendigkeit einer inhaltlichen Betrachtung hin und wird im Abschnitt 5 vorgenommen. Auch drei weitere Items (2, 3 und 5) zeigten eine Trennschärfe von unter 0.5, was auf inhaltlicher Ebene zu diskutieren ist.

Aus Homogenitätssicht spricht eine durchschnittliche Inter-Item-Korrelation $r_{ii}$ von 0.21 gerade noch für die Annahme eines eindimensionalen Tests [BD06].

Für alle Testpersonen lagen auch Daten der statischen Tests aus Übungs- und Testataufgaben vor, in die unter anderem Fehlermeldungen des Compilers bzgl. der syntaktischen Korrektheit einer Lösung einfließen. Daher wurde die Korrelation zwischen den Ergebnissen bei den Lückentexten und dem Anteil kompilierfähiger Lösungen in den Programmieraufgaben berechnet. Für Übungsaufgaben wurde eine Trennschärfe von 0.24 erreicht; für Testaufgaben eine Trennschärfe von 0.26. Die kompletten Item-Korrelationen sowie die Korrelationen mit den erreichten Punkten aus Übungen und Testaten sind visualisiert in Abbildung 3.

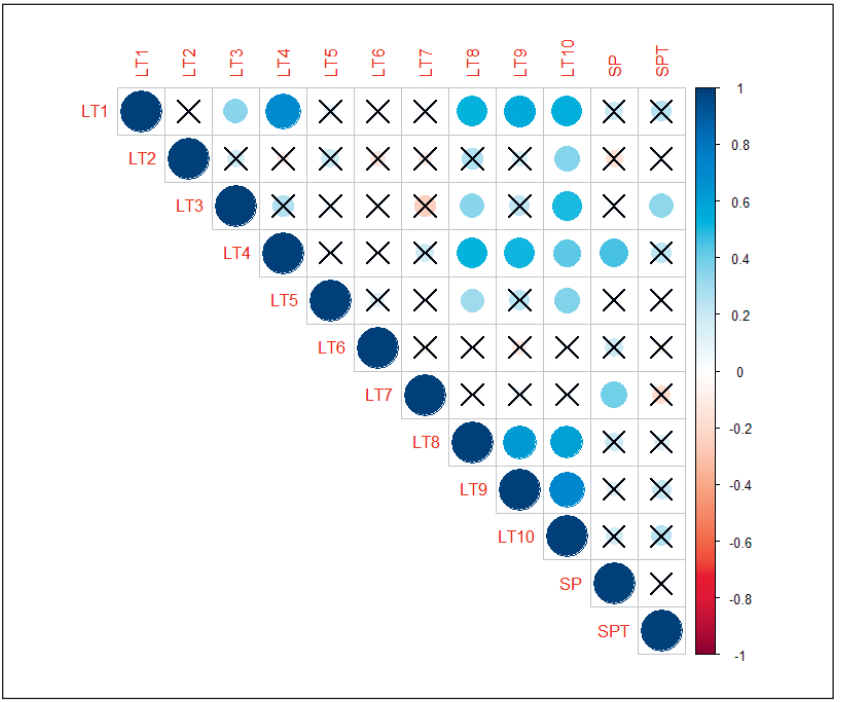


Abb. 3: Visualisierung der Korrelationsmatrix aller standardisierten Items incl. Punktzahlen aus statischen Tests (SP) sowie statischen Tests in Testatsituationen (SPT), mit X sind Korrelationen markiert, die das Signifikanzniveau verletzen ($p < .05$)

## 4.4 Deskriptive Statistik – Betrachtung auf Itemebene

In einem weiteren Schritt wurden die produzierten Ergebnisse auf Itemebene betrachtet. Für jedes Item ergab sich entsprechend eine Matrix, in der dichotom die produzierten Lösungen für alle Lücken und alle Probanden kodiert waren. Anschließend wurden für alle zehn Items die Kovarianzen sowie die sich daraus ergebenen Korrelationen der Lösung zu den jeweiligen Lücken ausgewertet, sowie die biseriellen Korrelationen zwischen den Lücken und der Item-Gesamt-Punktzahl als Maß für die Trennschärfe der Lücke auf Itemebene berechnet. Letztere wurden mit dem Befehl `biserial.cor()` aus dem R-Paket *ltm* [Ri06] ermittelt. Für die zehn Lücken des Items 8 (siehe Abbildung 1) ergaben sich bspw. folgende Varianzen, Kovarianzen und Korrelationen:

Tab. 2: Kovarianzen (untere Dreiecksmatrix) und Korrelationen (obere Dreiecksmatrix) für jede der zehn Lücken (L1 – L10) in Item 8, Varianzen entsprechen den Kovarianzen einer Lücke mit sich selbst. Die letzte Zeile zeigt die Trennschärfe der Lücke als biserielle Korrelation bzgl. der zu erreichenden Gesamtpunktzahl im Item. Die Nummerierung der Lücken ergibt sich aus dem Auftauchen in normaler Lesart.

| | **L1** | **L2** | **L3** | **L4** | **L5** | **L6** | **L7** | **L8** | **L9** | **L10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **L1** | 0.097 | 0.114 | −0.171 | −0.171 | 0.130 | 0.142 | 0.427 | 0.395 | 0.269 | −0.028 |
| **L2** | 0.016 | 0.194 | 0.161 | 0.161 | 0.160 | 0.171 | 0.252 | 0.254 | 0.360 | 0.368 |
| **L3** | −0.027 | 0.036 | 0.253 | 1 | −0.172 | 0.186 | 0.164 | 0.048 | 0.023 | 0.211 |
| **L4** | −0.027 | 0.036 | 0.253 | 0.253 | −0.172 | 0.186 | 0.164 | 0.048 | 0.023 | 0.211 |
| **L5** | 0.020 | 0.035 | −0.043 | −0.043 | 0.241 | 0.240 | 0.116 | −0.109 | 0.166 | 0.125 |
| **L6** | 0.012 | 0.021 | 0.026 | 0.026 | 0.033 | 0.080 | 0.483 | 0.268 | 0.313 | 0.011 |
| **L7** | 0.019 | 0.016 | 0.012 | 0.012 | 0.008 | 0.020 | 0.021 | 0.326 | 0.699 | 0.267 |
| **L8** | 0.047 | 0.043 | 0.009 | 0.009 | −0.020 | 0.029 | 0.018 | 0.144 | 0.465 | 0.151 |
| **L9** | 0.017 | 0.032 | 0.002 | 0.002 | 0.017 | 0.018 | 0.021 | 0.036 | 0.042 | 0.132 |
| **L10** | −0.004 | 0.069 | 0.045 | 0.045 | 0.026 | 0.001 | 0.017 | 0.025 | 0.012 | 0.183 |
| $r_{bis}$ | 0.292 | 0.600 | 0.600 | 0.600 | 0.298 | 0.504 | 0.600 | 0.474 | 0.518 | 0.521 |

# 5 Interpretation & Schlussfolgerungen

Zuerst bleibt festzuhalten, dass die Populationsgröße verallgemeinernde Spekulationen verbietet. Dennoch können erste Hinweise auf sehr gute bzw. schlechte oder nicht funktionierende Items gewonnen werden. Diese werden zuerst auf Ebene des gesamten Testes und anschließend auf Itemebene diskutiert.

## 5.1 Interpretation auf Testebene

Insgesamt lässt sich erkennen, dass die Items auf Gesamttestebene für die untersuchte Population relativ leicht zu lösen waren. Dies wird ersichtlich am Deckeneffekt, der sich im Boxplot widerspiegelt. Dies kann vor dem Hintergrund der freiwilligen Bearbeitung einerseits dahingehend interpretiert werden, dass die zehn Items nur von den Studierenden bearbeitet wurden, welche

sich auch selbst zugetraut haben, alle Items zu bearbeiten und diese auch gut zu lösen. Studierende, für die der Umgang mit Syntax ein Problem darstellt, sind daher evtl. nur teilweise oder gar nicht durch die Items erfasst worden. Wie zuverlässig diese Interpretation ist, könnte in zukünftigen Untersuchungen durch Betrachtung externer Faktoren ergänzt werden, wie bspw. der Abschlussnote. Die Ergebnisse können deshalb auch nicht als Nachweis dafür gewertet werden, dass der Umgang mit Java-Syntax als besonders einfach zu bewerten ist. Gegen eine solche Interpretation sprechen auch einschlägige Ergebnisse der fachdidaktischen Forschung. So konnten [De11] zeigen, dass Probleme mit der Syntax unabhängig vom konzeptuellen Wissen auf allen Fähigkeitsebenen zu finden sind. Neben dem Stichprobenumfang sind des Weiteren auch die Korrelationen mit den Punkten aus den statischen Tests zu gering. D. h. Probanden, die hohe Werte in den Items erreicht haben, lieferten nicht zwangsweise syntaktisch fehlerfreie Testataufgaben ab (und umgekehrt). Daraus folgt, dass die Items nicht als Prädiktor zulässig sind, sondern allenfalls als weiteres Hilfsinstrument um Rückschlüsse auf die Ausprägung der Programmierkompetenz zu ziehen. Gleichzeitig muss jedoch auch erwähnt werden, dass die Punktzahlen aus statischen Tests allgemein nur ein wenig geeignetes Maß für die Beherrschung der Syntax sein können, da hier nur derjenige Anteil an Lösungen erfasst wird, die ins Übungssystem hochgeladen werden. Über die Häufigkeit von Syntaxfehler insgesamt können daraus nur sehr begrenzte Rückschlüsse gezogen werden.

Bei der Analyse der Korrelationen untereinander fällt zudem auf, dass einzelne Items sowohl schlecht mit dem Test insgesamt korrelieren, und daher eine schlechte Trennschärfe (siehe bspw. Item 6) aufweisen, als auch sehr schlecht mit einer Großzahl an anderen Items korrelieren. Dies muss nicht zwangsweise auf ein gänzlich ungeeignetes Item hinweisen, sondern könnte auch ein Indiz für eine zu geringe Anzahl oder eine zu homogene Gruppe von Probanden sein. Bei zukünftigen höheren Probandenzahlen wäre es nach Überarbeitung problematischer Items zu überlegen, mittels faktorenanalytischer Maßnahmen die Itemanzahl zu minimieren und nur solche Items zuzulassen, die sowohl hohe Trennschärfen, als auch hohe durchschnittliche Item-Korrelationen garantieren.

Exemplarisch wurde dies bereits über iteratives Entfernen derjenigen Items durchgeführt, die nur sehr geringe positive ($r < .1$) bzw. negative Trennschärfen aufweisen. Als Richtmaß für die Verbesserung wurden wiederum Cronbachs Alpha sowie die durchschnittliche Inter-Item-Korrelation $r_{ii}$ betrachtet. Durch Entfernen der Items 6 und 7 ergeben sich ein Alpha von 0.81 sowie ein $r_{ii} = .34$, was durchaus als Verbesserung des Tests gewertet werden

kann. Entfernt man zusätzlich noch die Items mit mittleren Trennschärfen bzw. betrachtet nur solche, mit sehr starken Trennschärfen (Items 1, 4, 8, 9, und 10), ergibt sich zwar ein Alpha von 0.87 aber dafür auch eine durchschnittliche Inter-Item-Korrelation $r_{ii}$ von 0.58, was bereits als partiell redundanter Test interpretiert werden kann [BD06, S. 220]. Von einer Reduktion der Testitems bis zu dieser Stufe muss daher abgesehen werden.

## 5.2 Interpretation auf Itemebene

Auf Itemebene gilt es mehrere Punkte zu beachten. Dazu gehört zum einen die Betrachtung der Lückenanzahl als möglicher Einfluss auf die Sekundärvarianz. Wie im Boxplot zu sehen ist, sind die Items 2 und 6 nicht nur besonders leicht zu lösen, sie leisten auch keinen Beitrag zur Unterscheidung zwischen den Probanden (bis auf wenige Ausnahmen). Dies kann durch die geringe Anzahl der Lücken erklärt werden. Bereits wenige korrekte Einträge liefern hohe prozentuale Anteile. Im Gegensatz bspw. zum Item 8 mit zehn Lücken, weist Item 2 nur fünf Lücken und Item 6 nur vier Lücken auf. Es muss daher bereits auf dieser Ebene hinterfragt werden, inwiefern sich die Items durch die vorgenommene Normierung vergleichen lassen. In zukünftigen Untersuchungen muss daher berücksichtigt werden, welchen Einfluss die Variation der Lückenanzahl potentiell auf das Ergebnis und dessen Interpretation hat.

Für eine detailliertere Betrachtung wurden zum anderen die Kovarianzen und die daraus resultierenden Korrelationen der Items untereinander betrachtet (siehe Tabelle 2 beispielhaft für Item 8). Niedrige Varianzen lassen an dieser Stelle auf Lücken schließen, die entweder offensichtliche oder sehr schwere Wörter verlangen. So musste in der siebenten Lücke das Schlüsselwort `void` hinzugefügt werden. Dies wurde aber nicht durch das Fehlen des Schlüsselwortes `return` im Methodenrumpf signalisiert, sondern vermutlich bei einem Großteil durch den stets gleichen Aufbau des `main`-Methodenkopfes induziert. Bestätigt wird dies durch die eher niedrige Korrelation von 0.252 mit der Lücke 2, die ebenfalls das Schlüsselwort `void` verlangt. Die siebente Lücke prüft also eher, wie gut jemand den Methodenkopf der `main`-Methode in Java auswendig gelernt hat. Es ist daher wenig verwunderlich, dass sich gute Korrelationen mit den Lücken 1, 6 und 9 ergeben, da diese ebenfalls nach bestimmten Schlüsselworten in „Standard-Situationen“ fragen (Klassendeklaration, Ausgabe auf der Konsole, Objektinstanziierung). Diese Lücken lassen sich ausfüllen, ohne den restlichen Quelltext zu betrachten. Ebenfalls wenig überraschend sind die perfekten Korrelationen der Lücken 3

und 4 miteinander, da beide dieselbe Lösung im selben Kontext verlangen. Interessanter sind jedoch negative Korrelationen. Diese geben zwar Hinweise auf problematische Kombinationen, verbieten jedoch eine generelle Identifikation des problematischen Konzeptes. Für Item 8 ergeben sich bspw. (schwache) negative Korrelationen zwischen der ersten Lücke sowie den Lücken 3, 4 und 10. Gerade die erste Lücke weist auch bei weiteren Items negative Korrelationen mit den restlichen Lücken auf, was den Schluss zulässt, dass häufig erste, einfache Schlüsselworte gerade noch gewusst werden, jedoch mit steigender Komplexität keine Antworten mehr gegeben werden können. Negative Korrelationen sind jedoch zusätzlich insofern problematisch, als dass dadurch möglicherweise miteinander vermischte Konzepte abgefragt werden. Im Item 7 wurden bspw. fünf von zehn Lücken von allen Probanden korrekt gelöst und wiesen daher gar keine Varianz auf, was wiederum die schlechte Trennschärfe des Items erklärt.

Schlussendlich wurden für jedes Item auch die biseriellen Korrelationen als Maß für die Trennschärfen der Lücken bzgl. der Gesamtpunktzahl pro Item betrachtet. Diese weisen mittlere bis gute Trennschärfen auf, so dass der zukünftige Ausschluss potentieller Lücken inhaltsgeleitet geschehen muss. Dabei ist des Weiteren darauf zu achten, welche Schlüsselworte an welchen Positionen abgefragt werden, um nicht möglicherweise auswendig gelernte Muster, sondern die syntaktisch korrekten Inhalte abzufragen.

## 6 Fazit und Ausblick

Es wurden zehn Items nach dokumentierten Richtlinien erstellt und in einer ersten empirischen Erhebung semesterbegleitend von 47 Testpersonen vollständig bearbeitet. Die Items wurden mittels Berechnung der internen Konsistenz sowie der mittleren Inter-Item-Korrelation auf Testebene und mittels Kovarianz- und Korrelationsberechnungen auf Itemebene ausgewertet. Dabei wurden zum einen für die untersuchte Population schlecht funktionierende Items identifiziert und durch iteratives Entfernen eine Subgruppe von Items gebildet, die starke Korrelationen untereinander aufweisen. Teilweise sind die daraus resultierenden Korrelationen allerdings so stark, dass über eine redundante Testung spekuliert werden muss. Zum anderen wurden Lücken mit niedriger Varianz und schlecht oder gar negativ korrelierenden Lücken identifiziert. Der aus der Itemkonstruktion gewonnene Eindruck, dass die Erstellung der Items nicht automatisiert erfolgen kann, wurde damit bestätigt. Durch die Spezifizierung auf syntaktische Konstrukte sowie das erste Auf-

zeigen einer Subgruppe von Items mit guter Inter-Item-Korrelation und guter interner Konsistenz vertreten die Autoren die Auffassung, dass Lückentext-Items sinnvoll zu einer gesamtheitlichen Kompetenzmessung beitragen können. Im Rahmen weiterer Analysen ist zu untersuchen, inwiefern die Items genutzt werden können, um bessere und gezieltere Rückschlüsse auf generelle Fehlvorstellungen und Probleme der Lernenden zu ziehen, was wiederum zu einer möglichen Verbesserung der Lehre beiträgt.

Des Weiteren wurden nützliche Erkenntnisse für die Item-Konstruktion abgeleitet. So ist im Rahmen der Varianzanalyse zu überlegen, ob weiterhin Items eingesetzt werden, welche unterschiedliche Anzahlen von Lücken aufweisen, oder ob eine andere Normalisierung bzw. Standardisierung zu einer hinreichenden Verbesserung der Datenqualität führen würde. Möglicherweise wird die Schwierigkeit eines Items teilweise durch die abgefragten Inhalte und teilweise durch die Lückenanzahl beeinflusst, welche als Sekundärvarianz in die Gesamtvarianz eingeht. Die geforderte Lückenreduktion kann dabei durch Verwendung verschiedener Kriterien, bspw. der Varianz einzelner Lücken selbst, der Korrelation mit anderen Lücken, der Trennschärfe sowie aus inhaltlichen Aspekten erfolgen. Für die nächsten Experimente erscheint es daher insbesondere sinnvoll, mehrere Items mit identischer Anzahl von Lücken und unter Verzicht auf Lücken in bekannten Mustern zu erstellen. In jedem Fall muss bei einem erneuten Einsatz der Items die Unkorreliertheit der Fehlervariablen und der essentiellen $\tau$-Äquivalenz als Voraussetzung für die Verwendung von Cronbach's Alpha überprüft werden, um die Teststatistik sinnvoll interpretieren zu können. Dass Interpretationen von Alpha trotz Verletzung dieser Voraussetzungen valide sein können, wurde berichtet in [Wa15].

Die bisher erhaltenen Rohdaten lassen zusätzlich im Rahmen einer Sekundäranalyse die Auswertung nach Konzepten zu, die in mehreren Items vorkommen.

Die Items sind über den Erstautor für den Einsatz in nicht-kommerziellen Untersuchungen kostenfrei erhältlich.

## Literaturverzeichnis


[BD06] Bortz, J.; Döring, N.: Quantitative Methoden der Datenerhebung. In (Bortz, J.; Döring, N. Hrsg.): Forschungsmethoden und Evaluation. 4. Auflage, Springer, Berlin, S. 137–293, 2006, S. 220.

[De11] Denny, P. et al.: Understanding the syntax barrier for novices. In: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE), Darmstadt 2011. ACM, New York, S. 208–212, 2011.

[KHB16] Kramer, M.; Hubwieser, P.; Brinda, T.: A Competency Structure Model of Object-Oriented Programming. In: Proceedings of the International Conference on Learning and Teaching in Computing and Engineering (LaTICE), Mumbai. IEEE, S. 1–8, 2016.

[LY09] Lau, W.; Yuen, A.: Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. British Journal of Educational Technology 40/4, S. 696–712, 2009.

[Mc01] McCracken, M. et al.: A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. ACM SIGCSE Bulletin 33/4, S. 125–180, 2001.

[Re17] Revelle, W.: psych: Procedures for Personality and Psychological Research. Northwestern University, Evanston, Illinois, 2017.

[Ri06] Rizopoulos, D.: ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses. Journal of Statistical Software 17/5, S. 1–25, 2006.

[SEB82] Soloway, E.; Ehrlich, K.; Bonar, J.: Tapping into Tacit Programming Knowledge. Proceedings of the 1982 Conference on Human Factors in Computing Systems. ACM, New York, S. 52–57, 1982.

[SS13] Stefik, A.; Siebert, S.: An empirical investigation into programming language syntax. ACM Transactions on Computing Education (TOCE) 13/4, 19, 2013.

[SKG17] Striewe, M.; Kramer, M.; Goedicke, M.: Ein Lückentext-Test zur Beherrschung einer Programmiersprache. In: DeLFI 2017 – Die 15. e-Learning Fachtagung Informatik der Gesellschaft für Informatik, Volume 273 of Lecture Notes in Informatics. Köllen, Bonn, S. 261–266, 2017.

[Wa15] Warrens, M.J.: Some relationships between Cronbach's alpha and the Spearman-Brown formula. Journal of Classification 32/1, S. 127–137, 2015.

[Wi96] Winslow, L.: Programming Pedagogy – A Psychological Overview. ACM SIGCSE Bulletin 28/3, S. 17–22, 1996.

# Spezielle Themen des Informatikstudiums

# Data Science, Theoretische Informatik und Wissenschaftliches Arbeiten

# Was ist Data Science? Ermittlung der informatischen Inhalte durch Analyse von Studienangeboten

**Andreas Grillenberger und Ralf Romeike**
Friedrich-Alexander-Universität Erlangen-Nürnberg
Didaktik der Informatik
Martensstraße 3
91058 Erlangen
andreas.grillenberger@fau.de
ralf.romeike@fau.de

**Abstract:** In Zusammenhang mit den Entwicklungen der vergangenen Jahre, insbesondere in den Bereichen *Big Data, Datenmanagement* und *Maschinenlernen,* hat sich der Umgang mit Daten und deren Analyse wesentlich weiterentwickelt. Mittlerweile wird die Datenwissenschaft als eigene Disziplin angesehen, die auch immer stärker durch entsprechende Studiengänge an Hochschulen repräsentiert wird. Trotz dieser zunehmenden Bedeutung ist jedoch oft unklar, welche konkreten Inhalte mit ihr in Verbindung stehen, da sie in verschiedensten Ausprägungen auftritt. In diesem Beitrag werden daher die hinter der Data Science stehenden informatischen Inhalte durch eine qualitative Analyse der Modulhandbücher etablierter Studiengänge aus diesem Bereich ermittelt und so ein Beitrag zur Charakterisierung dieser Disziplin geleistet. Am Beispiel der Entwicklung eines Data-Literacy-Kompetenzmodells, die als Ausblick skizziert wird, wird die Bedeutung dieser Charakterisierung für die weitere Forschung expliziert.

**Keywords:** Data Science, Big Data, Inhalte, Studiengänge, Data Literacy, Kompetenzen.

## 1 Data Science als Gegenstand informatischer Bildung

In Zusammenhang mit der zunehmenden Verarbeitung immer komplexerer Daten in verschiedensten Kontexten werden seit kurzem vielerorts neue Studiengänge entwickelt und eingerichtet, die sich mit der *Data Science* beschäf-

tigen. Innerhalb dieser Studiengänge ist eine deutliche inhaltliche Vielfalt erkennbar. Allgemein werden jedoch üblicherweise zumindest Mathematik, Statistik und Informatik als Fundamente angesehen. Je nach konkreter Ausgestaltung spielen diese jedoch eine unterschiedlich große Rolle, sodass zum Teil deutlich unterschiedliche inhaltliche und methodische Ausrichtungen erkennbar sind. Obwohl die Datenwissenschaft sowohl in der Eigen- als auch der Fremdwahrnehmung oft als eigene Disziplin betrachtet wird, als eine Wissenschaft der (Arbeit mit und Verwaltung von) Daten, weist sie jedoch auch starke Bezüge zur Informatik auf und greift auf vielfältige informatische Grundlagen zurück.

Wie die Gesellschaft für Informatik [GI18] verdeutlicht, ist *Data Science* heute mehr als nur ein Buzzword. Im Gegenteil kommt diesem Wissenschaftsfeld eine wachsende Bedeutung zu, auch aufgrund ihrer klaren Bezüge zu der Entwicklung, die in Alltag, Gesellschaft und Politik häufig unter dem Begriff *Digitalisierung* zusammengefasst wird: Indem versucht wird, alle Bereiche des täglichen Lebens in Daten abzubilden, entsteht die Herausforderung, diese adäquat zu verwalten und zu nutzen. Diese Entwicklung macht selbst vor dem Privatleben, in dem oft Ergebnisse der kontinuierlichen Datenerfassung und -verarbeitung genutzt werden, nicht Halt: Im Sinne einer *Data Literacy* muss heute jeder einen fundierten Umgang mit Daten pflegen, solche speichern und verarbeiten. Aber auch in den meisten anderen Wissenschaftsdisziplinen gewinnt Data Science bzw. ihre Methoden an Bedeutung und wird sogar als neues viertes Wissenschaftsparadigma gehandelt (z. B. [HTT09]). Entsprechend entstehen, beispielsweise in den Geisteswissenschaften unter dem Begriff *Digital Humanities,* neue Ausrichtungen, die mindestens grundlegende Kompetenzen aus der Datenwissenschaft benötigen.

Für die informatische Bildung eröffnen sich in diesem Bereich vielfältige Forschungs- und Handlungsfelder, u. a. bei der Ermittlung und Fundierung der Kompetenzen sowohl von Data Science als auch von Data Literacy, aber auch bei der Integration dieser Bereiche in die (Hochschul-)Ausbildung und der Gestaltung und Weiterentwicklung von Data-Science-Studiengängen bzw. Data-Literacy-Modulen. Als Basis für derartige Arbeiten muss eine Klarheit darüber geschaffen werden, welche Aspekte der Informatik für die Data Science relevant bzw. notwendig sind. In diesem Beitrag erfolgt daher eine Klärung des informatischen Gegenstandsbereichs *Data Science,* indem die hinter dem Begriff stehenden informatischen Inhalte durch eine explorative Analyse bereits implementierter Data-Science-Studiengänge ermittelt werden. Somit wird ein Überblick über die Data Science gegeben und diese durch ihre informatischen Aspekte charakterisiert.

## 2 Verwandte Arbeiten

Trotz der verbreiteten Nutzung des Begriffs *Data Science* besteht keine klare Einigkeit über deren charakteristische Inhalte und Kompetenzen: Aus disziplinorientierter Sicht wird sie beispielsweise als *„data science = statistics + informatics + computing + communication + sociology + management | data + environment + thinking“*[1] definiert [Ca17]. Andere Definitionen legen den Schwerpunkt auf den praktischen Nutzen: *„Data science is the extraction of actionable knowledge directly from data through a process of discovery, or hypothesis formulation and hypothesis testing.“* [NI15] In verschiedenen Arbeiten wurde daher bereits versucht, diese neue Wissenschaftsdisziplin zu charakterisieren, beispielsweise durch ein Kompetenzmodell (vgl. [DBM17]) sowie einen Body of Knowledge (vgl. [DBW17]). Dabei wurde zum Teil empirisch vorgegangen, indem als Basis für das Kompetenzmodell die Anforderungen, die in Stellenanzeigen an Datenanalysten gestellt werden, herangezogen und Studien zu deren Kompetenzen und Fertigkeiten sowie Blogartikel und Forenbeiträge miteinbezogen wurden. Das auf dieser Basis entstandene Kompetenzmodell [DBM17] gibt somit insbesondere eine externale Sicht auf das wieder, was Data Science bzw. Data Scientists aus Sicht von Unternehmen sein bzw. können sollen. Im Gegensatz dazu wird in diesem Beitrag angestrebt, die wissenschaftlich geprägte Selbstwahrnehmung des Fachs mit Fokus auf die informatischen Aspekte abzubilden.

Auch in anderen Arbeiten wird die Bedeutung der Data Science deutlich: So konnte das HIS-Institut für Hochschulentwicklung in Deutschland bereits eine relativ große Vielfalt von rund 25 Studiengängen zur Data Science und eine steigende Tendenz ermitteln (vgl. [GI18]). In der informatikdidaktischen Forschung bleibt die Disziplin Data Science bisher jedoch nahezu unbeachtet: Zwar wurden bereits auf Bildung hin ausgerichtete Arbeiten durchgeführt und Richtlinien für Data-Science-Curricula erarbeitet (z. B. [Ve17]). Eine tiefergehende Betrachtung aus informatikdidaktischer Sicht fand bisher jedoch nicht statt. Verwandte Arbeiten aus der Informatikdidaktik stammen damit eher aus anderen Bereichen der Informatik: Insbesondere dienen Arbeiten zum Fachgebiet Datenmanagement, beispielsweise das Modell der Schlüsselkonzepte des Datenmanagements [GR17], aufgrund der starken Überschneidung mit der Data Science im Rahmen der weiteren Forschung zu Data Science und Data Literacy als wertvolle Grundlage.

---

1 Das Zeichen „|“ ist dabei als „conditional on“ zu verstehen [Ca17].

## 3 Ermittlung der Kerninhalte von Data-Science-Studiengängen

Zur fachlichen Fundierung der (informatikdidaktischen) Data-Science-Forschung ist es hilfreich und notwendig, die informatischen Aspekte hinter dem Begriff *Data Science* zu ermitteln. Einen wichtigen Einblick kann hier eine Untersuchung bereits etablierter Data-Science-Studiengänge geben.

Im Rahmen einer Voruntersuchung wurden Modulhandbücher weniger Data-Science-Studiengänge[2] betrachtet, die einen Einblick sowohl in die enthaltenen Inhalte als auch Kompetenzen geben. Dabei hat sich gezeigt, dass die Inhalte, die von den die Studiengänge gestaltenden Experten als zentral für die Data Science angesehen werden, aus den betrachteten Dokumenten klar ersichtlich sind. Die Untersuchung der angestrebten Kompetenzen wäre auf dieser Basis jedoch stark subjektiv geprägt: Aufgrund der unterschiedlichen Formulierung und Detaillierung der jeweils festgehaltenen Kompetenzen, beispielsweise als zum Teil nicht operationalisierte Lernziele oder auch rein inhaltsorientierte Modulbeschreibungen, würden interpretative Aspekte eine wichtige Rolle spielen. Die Objektivität der Untersuchung wäre somit stark eingeschränkt. Entsprechend wird in dieser Arbeit der Schwerpunkt auf die objektiver untersuchbaren Inhalte gelegt. Im Rahmen der Vorstudie wurden zusätzlich deutschsprachige und internationale Studiengänge[3] kontrastiert: Dabei zeigte sich, dass – vermutlich aufgrund unterschiedlicher rechtlicher Rahmenbedingungen – die Dokumente nicht nur einen sehr unterschiedlichen Aufbau haben, sondern sich auch im Detailgrad stark unterscheiden. Um auch an dieser Stelle eine möglichst nachvollziehbare Methodik anzuwenden und subjektive Einflüsse zu vermeiden, wird der Fokus daher auf Studiengänge im deutschsprachigen Raum gelegt, bei denen ein ähnlicher Data-Science-Begriff und ähnliche Grundlagen zur Ausgestaltung der charakteristischen Dokumente zu erwarten sind und somit alle Dokumente auf dieselbe Weise untersucht werden können. Um die Validität der Ergebnisse auch außerhalb des deutschsprachigen Raums zu überprüfen, werden die Ergebnisse der Untersuchung jedoch im Nachgang mit internationalen Studiengängen kontrastiert.

---

2 In der Vorabuntersuchung wurden Data-Science-Studiengänge der Universitäten Marburg und Stuttgart sowie der Hochschule der Medien Stuttgart und der Hochschule Albstadt-Sigmaringen betrachtet.

3 Zur Kontrastierung wurden die Data-Science-Studiengänge der Universität Berkeley und der IT-Universität Copenhagen betrachtet.

Die zentrale Forschungsfrage dieser Arbeit lautet daher: *Welche zentralen Inhalte charakterisieren Data-Science-Studiengänge im deutschsprachigen Raum?* Um dieser nachzugehen, wurde ein empirischer Ansatz basierend auf einer qualitativen Inhaltsanalyse nach Mayring [Ma10] gewählt. Dieser erlaubt es unter anderem, einen Literaturkanon systematisch zu explorieren und darauf basierend ein Kategoriensystem aufzubauen, das hier der inhaltlichen Charakterisierung der Data Science entsprechen wird. Im Allgemeinen kann dieses Kategoriensystem induktiv, aus dem Material heraus, oder deduktiv, auf Basis existierender Arbeiten, aufgebaut werden. Das angestrebte Ziel dieser Analyse legt klar einen induktiven Ansatz nahe, da damit das analysierte Material adäquat repräsentiert werden kann, ohne dass die Einordnung in ein existierendes Kategoriensystem zu einer Beeinflussung und zu Einschränkungen bei der Exploration kommen kann.

Im Sinne der Methodik nach Mayring wird die Analyse in mehrere Schritte zerlegt: Zuerst wird der Literaturkanon und, zur Erhöhung der Genauigkeit und Objektivität, auch die Kodiereinheit und Analysekriterien festgelegt. Darauf basierend erfolgt die Analyse der Dokumente, die typischerweise in die Erstellung eines hierarchisch organisierten Kategoriensystems mündet. In dieser Arbeit wird jedoch vorerst auf die Hierarchisierung verzichtet und stattdessen eine flache Kategorienmenge aufgebaut und erst im Nachgang in einer expliziten Strukturierungsphase hierarchisiert. Dies wird für das angestrebte Ziel als vorteilhaft erachtet, da auf diese Weise frühe Beeinflussungen der Analyse durch die Hierarchisierung und den Versuch, neue Aspekte eher in das bereits existierende System einzuordnen anstatt die Hierarchie zu ergänzen, vermieden werden.

## 3.1 Festlegung von Literaturkanon, Kodiereinheit und Analysekriterien

Als Basis für die Analyse werden Modulhandbücher verschiedener Data-Science-Studiengänge aus dem deutschsprachigen Raum ausgewählt. Um einen Überblick über diese zu gewinnen, wurde der Hochschulkompass[4] herangezogen. Unter den dort auffindbaren Studiengängen sind jedoch verschiedene, die zwar Themen der Data Science anschneiden, aber eher am Rande betrachten und andere Schwerpunkte setzen: beispielsweise Informatikstudiengänge,

4 http://www.hochschulkompass.de, zugegriffen am 25. 06. 2018

die lediglich wenige Data-Science-Module verpflichtend beinhalten oder nur die Möglichkeit bieten, sich in Wahlpflichtmodulen darauf zu spezialisieren. Um eine Beeinflussung der Ergebnisse durch diese zu vermeiden, wurden solche Studiengänge nicht miteinbezogen, da in diesen nicht klar erkennbar ist, welche Inhalte speziell für den Bereich Data Science als zentral erachtet werden.

Innerhalb der auf diese Weise vorselektierten Studiengänge wurde eine weitere Filterung der betrachteten Module vorgenommen: Da das Ziel die Ermittlung zentraler Inhalte der Data-Science-Studiengänge war, wurden nur Module betrachtet, die von den Hochschulen als Pflicht – und somit als Mindestanforderungsprofil für Absolventen des jeweiligen Studiengangs – erachtet werden. Wahlpflicht- und Wahlmodule blieben somit unberücksichtigt, genauso wie konsekutive Masterstudiengänge, die als Vertiefung gegenüber dem vorherigen Data-Science-Bachelorstudiengang betrachtet wurden. Nicht-konsekutive Master- wurden jedoch genauso wie Bachelorstudiengänge miteinbezogen, da sie gleichermaßen die Grundlagen der Data Science thematisieren müssen.

Diesen Kriterien folgend wurden Studiengänge der folgenden Hochschulen, unabhängig von ihrer Ausgestaltung (bspw. mit Fokus auf Informatik oder Mathematik), als Basis für die Analyse ausgewählt: *Beuth Hochschule Berlin, Hochschule Darmstadt, Technische Universität Dortmund, Universität Jena, Universität München/LMU, Universität Mannheim*[5]*, Hochschule Albstadt-Sigmaringen, Universität Salzburg, Hochschule der Medien Stuttgart* (alle M. Sc.), *Universität Marburg, Universität Stuttgart* (beide B. Sc.).

Die Kodiereinheit wurde auf semantische Weise so festlegt, dass jede Kodierung, unabhängig von ihrer Länge im Text, sich jeweils auf genau einen inhaltlichen Aspekt beziehen soll. Als Auswahlkriterium wurde festgelegt, dass alle in den Dokumenten genannten Inhalte betrachtet werden, jedoch wurden aufgrund des Analyseziels Inhalte, die nicht charakteristisch für Data Science sind, sondern allgemeine informatische oder mathematische Grundlagen repräsentieren, nur in geringem Detailgrad erfasst. Somit wurden beispielsweise *endliche Automaten* und *elementare Statistik* nicht im Detail erfasst, sondern nur die Kategorien *Statistik* bzw. *Theoretische Informatik* eingeführt. Hingegen wurde beispielsweise die Methode *Klassifikation,* die in der Datenanalyse eine wichtige Rolle spielt und in der Betrachtung in den Curricula

5 Der *Mannheim Master in Data Science* ist im Folgenden nicht erkennbar, da für diesen keine Pflichtmodule definiert sind und somit im Sinne der Auswahlkriterien keine Module berücksichtigt werden konnten.

über die rein mathematische Sichtweise hinausgeht, zunächst explizit in das Kategoriensystem aufgenommen.

## 3.2 Analyse der Dokumente und Strukturierung der Ergebnisse

Als nächstes wurden die ausgewählten Dokumente unter Berücksichtigung der festgelegten Kriterien systematisch analysiert. Statt direkt ein hierarchisches Kategoriensystem aufzubauen, wurde eine Liste aller genannten Inhalte erstellt. Dabei wurden Kodierungen vermieden, die zu detailliert sind, soweit dies bereits an dieser Stelle klar ersichtlich war: Beispielsweise wurden bei gemeinsamer Nennung einer großen Anzahl verschiedener Methoden zum überwachten Lernen nicht alle Methoden einzeln aufgenommen, sondern gesammelt unter dem geeigneten Überbegriff, da diese im nächsten Schritt sowieso zusammengefasst worden wären. Nicht in allen Fällen war dies jedoch bereits zu diesem Zeitpunkt ersichtlich, insbesondere wenn verwandte Begriffe nicht gemeinsam, sondern in unterschiedlichen Bereichen oder Dokumenten genannt wurden. Diese potentiellen Zusammenfassungen wurden vorerst unterlassen. Somit resultierte die Analyse in einer Liste von 106 inhaltlichen Aspekten der Data Science, die jedoch nicht überschneidungsfrei und auf unterschiedlichem Detailgrad angesiedelt waren. Daher wurde diese Liste im folgenden Schritt strukturiert und zusammengefasst.

Dazu wurde diese Liste hierarchisiert: Es wurde entschieden, dass auf der obersten Ebene nur eher abstrakte Begriffe genannt werden sollen, die die großen Themenbereiche der Data Science repräsentieren. Entsprechend wurden möglichst wenige Codierungen auf oberster Ebene angestrebt. Zusätzlich wurden die beiden Bereiche *informatische Grundlagen* sowie *mathematische Grundlagen* eingeführt, unter denen alle Aspekte zusammengefasst werden, die eher übergreifend bzw. nicht speziell der Data Science zugehörig sind, der Vollständigkeit halber aber nicht unerwähnt bleiben sollen. Trotzdem werden diese Grundlagen bewusst weniger tiefgehend betrachtet, sodass die darunter angeordneten Begriffe auf einem anderen Niveau angesiedelt sind als die im Fokus dieser Arbeit stehenden Inhalte der Data Science. Auf den tieferen Ebenen wurden Begriffe unter dem jeweiligen Oberbegriff zusammengefasst, die von ihrer Bedeutung zusammengehörig sind und häufig gemeinsam genannt wurden, jedoch nur wenn durch die Zusammenfassung keine relevanten Details verloren gehen. Entsprechend wurden die Methoden der Datenanalyse in vier Kategorien subsumiert: *Methoden des unüberwachten Lernens* (insbesondere *Clustering, Assoziation*), *Methoden des überwachten*

*Lernens* (insbesondere *Klassifikation, Entscheidungsbäume, Regression*), *Komplexere Methoden* (beispielsweise *Neuronale Netze*) sowie *Verknüpfung von Methoden/Ensemble-Learning.* Auf eine prinzipiell mögliche weitere Zusammenfassung, beispielsweise unter dem Begriff *Methoden der Datenanalyse,* wurde bewusst verzichtet, da dadurch die Unterscheidung und die unterschiedlichen Zwecke und Charakteristika in den Hintergrund rücken würden und so insbesondere auch der Einblick in die unterschiedliche Abdeckung dieser Methoden in den verschiedenen Studiengängen verloren gehen würde.

Zusätzlich wurden bei der Strukturierung alle Aspekte aus der Charakterisierung ausgefiltert, die nicht mit anderen zusammenfassbar waren, aber durch ihre geringe Repräsentation in weniger als einem Fünftel der analysierten Dokumente kaum geeignet sind, um den Kern der Data Science zu charakterisieren. Als Ergebnis der Untersuchung steht daher ein Kategoriensystem, das 31 inhaltliche Aspekte aus sechs großen Themenbereichen berücksichtigt. Diese Charakterisierung der Data Science wird in Tab. 1 zusammen mit der Abdeckung in den verschiedenen Studiengängen dargestellt.

## 3.3 Diskussion der Ergebnisse

Das entstandene Kategoriensystem beschreibt die Data Science aus informatischer Sicht durch vier große Bereiche: *Datenanalyse und Maschinenlernen, Big Data, Datenschutz, Ethik* und *Datenspeicher.* Hinzu kommen *informatische* und *mathematische Grundlagen*.

Ohne detaillierter auf die eigentlichen Inhalte einzugehen, kann bereits ein wesentlicher Einblick in die Data Science gewonnen werden, indem diese Bereiche diskutiert werden:

**Notwendige Grundlagen**

Obwohl die meisten der analysierten Data-Science-Studiengänge zu einem Masterabschluss hinführen und daher als Aufbaustudium konzipiert sind, ist klar erkennbar, dass sie ein unterschiedlich ausgeprägtes Fundament an informatischen und mathematischen Grundlagen voraussetzen und diese in entsprechenden Modulen thematisieren.

Obwohl die mathematischen Grundlagen nur oberflächlich untersucht wurden, kann ein erster Einblick gewonnen werden: Während alle untersuchten Studiengänge Kenntnisse in *Statistik* als notwendig erachten, werden *Lineare Algebra* und *Analysis* nur in jeweils 40 % der Studiengänge (beide

gemeinsam in 30 %) genannt. Eine detaillierte Untersuchung der mathematischen Aspekte würde jedoch den Rahmen dieser Analyse sprengen.

Auch der Blick auf die *informatischen Grundlagen* zeigt ein ähnliches Bild: Es ist erkennbar, dass insbesondere *Programmierung* sowie *Algorithmen und Datenstrukturen* als besonders zentral erachtet werden und in 90 % bzw. 50 % der analysierten Dokumente genannt wurden. Außerhalb dieser beiden Bereiche besteht jedoch nur eine geringe Übereinstimmung der Studiengänge, beispielsweise werden in 20 % Grundlagen in *Betriebssystemen* und *Theoretischer Informatik* ausgebildet, während nur in einem auch *Rechnerkommunikation* thematisiert wird. Im Bereich der Grundlagen besteht daher zwar ein grundlegendes Fundament, über das große Einigkeit herrscht, aber gleichzeitig eine breite Vielfalt an potentiellen Inhalten die, je nach Studienort und Ausrichtung des Studiengangs, von einem Data-Science-Absolventen beherrscht werden sollen.

**Inhaltliche Charakterisierung der Data Science**

Neben den notwendigen Grundlagen konnten vier für die Data Science spezifischere Inhaltsbereiche ermittelt werden. Diese umfassen *Datenspeicher,* insbesondere Aspekte des Fachgebiets Datenbanken/Datenmanagement wie (relationale) Datenbanken, Datenmodellierung und Abfragesprachen, den Bereich *Datenschutz und Ethik,* der bei Datenanalysen aus gesellschaftlicher aber auch informatischer Sicht eine wichtige Rolle spielt, sowie mit *Datenanalyse und Maschinenlernen* und *Big Data* Aspekte, die im Kontext der Data Science eine neue Bedeutung in der Informatik erlangen.

Insbesondere *Datenanalyse und Maschinenlernen* sind in der Datenwissenschaft zentral und werden in allen betrachteten Dokumenten umfangreich thematisiert. Dabei liegt ein Schwerpunkt auf der *Auswahl, Beurteilung und Anpassung von Analysemodellen* (in 100 % der Studiengänge), dem *Prozess der Datenanalyse* (70 %), sowie verschiedenen Methoden, insbesondere des *überwachten Lernens* (80 %), wie Klassifikation, Entscheidungsbäume und Regression. Durch die stark analytisch geprägte Sichtweise dieses Bereichs stellt er gleichzeitig den Bezug zur zentralen Aufgabe von Datenwissenschaftlern dar, die sich weniger um die konkrete Datensammlung bzw. langfristige Speicherung kümmern, sondern eher um die Gewinnung neuer Informationen aus Daten durch Nutzung entsprechender Methoden, sowie um die Aufbereitung der Analyseergebnisse.

Der zweite besonders zentrale Bereich ist *Big Data:* Die Erfassung, Verarbeitung und Analyse großer und vielfältiger Datenmengen innerhalb kurzer

Tab. 1: Kategoriensystem zur Beschreibung der Inhalte der Data Science zusammen mit ihrer jeweiligen Abdeckung in den analysierten Studiengängen

| | HdM Stuttgart | Univ. Salzburg | HS Albst.-Sigmar. | Univ. Stuttgart | LMU München | Univ. Marburg | Univ. Jena | TU Dortmund | HS Darmstadt | Beuth HS Berlin | Anz. Nennungen | % der Studiengänge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Mathematik** | | | | | | | | | | | | |
| Analysis | | | | × | | × | × | | | × | 4 | 40 % |
| Lineare Algebra | | | | × | | × | | | × | × | 4 | 40 % |
| Statistik | × | × | × | × | × | × | × | × | × | × | 10 | 100 % |
| **Informatische Grundlagen** | | | | | | | | | | | | |
| Algorithmen und Datenstrukturen | | × | | × | | × | × | × | | | 5 | 50 % |
| Betriebssysteme | | | | | | | × | | | × | 2 | 20 % |
| IT-Security | | | | | × | × | | | | × | 3 | 30 % |
| Programmierung | × | × | × | × | | × | × | × | × | × | 9 | 90 % |
| Rechnerarchitektur | | | | | | × | | | | | 1 | 10 % |
| Rechnerkommunikation | | | | | | | × | | | × | 2 | 20 % |
| Software Engineering | | | | × | | × | | | | × | 3 | 30 % |
| Theoretische Informatik | | | | × | | × | | | | | 2 | 20 % |
| Verteilte Systeme | | | | | | | | | | × | 1 | 10 % |
| **Datenanalyse und Maschinenlernen** | | | | | | | | | | | | |
| Analyseprozess | × | × | × | | × | | × | | × | × | 7 | 70 % |
| Datenvorverarbeitung | | | × | | | × | | | | × | 3 | 30 % |
| Ensemblelernen | × | | × | × | × | | | | × | × | 6 | 60 % |
| Komplexere Methoden | × | | × | | | | | | × | | 3 | 30 % |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methoden überwachtes Lernen | × | × | × | × | × | × | | | × | × | 8 | 80 % |
| Methoden unüberwachtes Lernen | × | | × | | × | × | | | | × | 5 | 50 % |
| Modellauswahl, -beurteilung, -anpassung | × | × | × | × | × | × | × | × | × | × | 10 | 100 % |
| Text Mining | × | | × | × | | | | | | × | 4 | 40 % |
| Visualisierung | × | × | | | | | × | | | × | 4 | 40 % |
| **Big Data** | | | | | | | | | | | | |
| Algorithmen und Methoden der Big-Data-Verarbeitung | × | | × | | × | | × | | | × | 5 | 50 % |
| Big-Data-Architekturen und -Systeme | × | | × | × | × | | × | | | × | 6 | 60 % |
| Prinzipien der Big-Data-Analyse | × | | × | × | | | | | | × | 4 | 40 % |
| Webdaten | × | | × | | | | | | | × | 3 | 30 % |
| **Datenschutz, Ethik** | | | | | | | | | | | | |
| Datenschutz | × | × | | | × | | | | × | × | 5 | 50 % |
| Ethische Aspekte | × | × | | | × | | | | × | × | 5 | 50 % |
| Sicherheit | × | | | × | × | | | | × | × | 5 | 50 % |
| **Datenspeicher** | | | | | | | | | | | | |
| Daten(bank)modellierung | × | × | × | × | | × | × | | | | 6 | 60 % |
| Datenbanken (insb. relational) | × | × | × | | | × | × | | × | | 6 | 60 % |
| Datenbanksprachen | × | × | × | | | × | × | | | | 5 | 50 % |

Zeiträume stellt eine wesentliche Herausforderung für die Data Science dar. Daher ist es nicht verwunderlich, dass insbesondere *Systeme* (in 60 % der Studiengänge) und *Methoden* (50 %) zur Beherrschung großer Datenmengen eine wichtige Rolle spielen. Je nach Interpretation kann *Big Data* jedoch auch anderen Bereichen Fachgebieten, insbesondere dem Datenmanagement, zugeordnet werden. Dies liefert eine mögliche Erklärung für die vergleichsweise geringe Repräsentation in vielen Studiengängen und teils deutlichen Unterschiede.

**Internationaler Vergleich**

Zum internationalen Vergleich der Ergebnisse wurden drei Studiengänge ausgewählt, die klar definierte Inhalte vorweisen und somit für den Vergleich geeignet sind: Der an der University of Berkeley angebotene *Master of Information and Data Science* sowie die *Master of Science in Data Science* der University of Washington und der Columbia University. Auch hier wurden entsprechende Modulhandbücher bzw. andere Dokumente, die die Studiengänge und deren Inhalte klar beschreiben, in einer qualitativen Inhaltsanalyse untersucht. Dieser lagen dieselben Grundsätze wie zuvor zugrunde, jedoch mit dem Unterschied, dass kein neues Kategoriensystem induktiv aufgebaut, sondern, die zuvor aufgebaute Charakterisierung deduktiv an die Materialien herangetragen wurde, um einen Vergleich zu ermöglichen. Die Ergebnisse der Analyse sind in Tab. 2 abgebildet.

Bei Auswertung der drei Studiengänge wurden keine Begriffe ermittelt, die nicht sinnvoll in die zuvor entwickelte Charakterisierung eingeordnet werden konnten. Es kann daher eine hohe Vollständigkeit dieser Charakterisierung angenommen werden. Gleichzeitig zeigt sich, dass diese Studiengänge zwar nicht die identischen Schwerpunkte setzen, Abweichungen aber relativ gering sind und nur geringfügig unterschiedlichen Schwerpunktsetzungen entsprechen sowie dem unterschiedlichen Bildungssystem geschuldet sind. Gerade dieser letzte Aspekt zeigt, dass die getrennte Analyse sinnvoll war, da so unterschiedliche Charakteristika und Voraussetzungen klar erkennbar bleiben.

Es ist auch erkennbar, dass ähnliche mathematische und informatische Vorkenntnisse vorausgesetzt werden. Außerdem stellt auch in diesen Studiengängen der Bereich *Datenanalyse und Maschinenlernen* den zentralen Schwerpunkt dar.

Tab. 2: Einordnung dreier US-amerikanischer Curricula in das entwickelte Kategoriensystem

| | Berkeley Univ. | Columbia Univ. | Univ. of Washington | Anz. der Nennungen | % der Studiengänge |
|---|---|---|---|---|---|
| **Mathematik** | | | | | |
| Analysis | | | | | |
| Lineare Algebra | | | | | |
| Statistik | × | × | | 2 | 67 % |
| **Informatische Grundlagen** | | | | | |
| Algorithmen und Datenstrukturen | | × | | 1 | 33 % |
| Betriebssysteme | | | | | |
| IT-Security | | | | | |
| Programmierung | × | × | | 2 | 67 % |
| Rechnerarchitektur | | | | | |
| Rechnerkommunikation | | | | | |
| Software Engineering | | | | | |
| Theoretische Informatik | | | | | |
| Verteilte Systeme | | | | | |
| **Datenanalyse und Maschinenlernen** | | | | | |
| Analyseprozess | × | | | 1 | 33 % |
| Datenvorverarbeitung | | × | × | 2 | 67 % |
| Ensemblelernen | × | × | × | 3 | 100 % |
| Komplexere Methoden | × | × | | 2 | 67 % |
| Methoden überwachtes Lernen | o | × | × | 2 | 67 % |
| Methoden unüberwachtes Lernen | o | | × | 1 | 33 % |
| Modellauswahl, -beurteilung, -anpassung | | × | | 1 | 33 % |
| Text Mining | | | | | |
| Visualisierung | | | × | 1 | 33 % |
| **Big Data** | | | | | |
| Algorithmen und Methoden der Big-Data-Verarbeitung | | × | | 1 | 33 % |
| Big-Data-Architekturen und -Systeme | × | × | × | 3 | 100 % |
| Prinzipien der Big-Data-Analyse | × | | × | 2 | 67 % |
| Webdaten | | × | | 1 | 33 % |
| **Datenschutz, Ethik** | | | | | |
| Datenschutz | | | × | 1 | 33 % |
| Ethische Aspekte | | | × | 1 | 33 % |
| Sicherheit | | | | | |
| **Datenspeicher** | | | | | |
| Daten(bank)modellierung | | × | × | 2 | 67 % |
| Datenbanken (insb. relational) | × | × | × | 3 | 100 % |
| Datenbanksprachen | | × | × | 2 | 67 % |

## 4 Ausblick: Entwicklung eines Data-Literacy-Kompetenzmodells

Die beschriebene Arbeit schafft durch eine fundierte Charakterisierung der Data Science einen wichtigen Beitrag für zukünftige Forschungsarbeiten: Neben einer Fundierung der Studiengangs- und Curriculums(weiter)entwicklung, können die ermittelten Inhalte der Data Science auch zur Fundierung der fachdidaktischen Forschung beitragen. In den nächsten Jahren wird das heute noch relativ neue Thema *Data Literacy* vermutlich ein wichtiges Thema der fachdidaktischen Forschung werden. Dieser insbesondere aus dem Hochschulkontext stammende neue Begriff zur Beschreibung einer Sammlung an Grundkompetenzen zum fundierten und zielgerichteten Umgang mit Daten ist insbesondere durch Aspekte des Datenmanagements, das eine eher statische Sichtweise auf Daten beinhaltet, und der Data Science, die die dynamischen Aspekte berücksichtigt, geprägt.

Zur Charakterisierung dieser Kompetenzen ist ein fachlich fundiertes Kompetenzmodell unabdingbar, welches bis dato noch nicht existiert. Bisher werden hingegen entsprechende Kompetenzen oft aus eher bedarfsorientierter Sichtweise betrachtet (vgl. [Ri15]). Ein solches Modell kann insbesondere unter Rückgriff auf die hier beschriebenen Inhalte der Data Science und weitere existierende Arbeiten zum Datenmanagement entwickelt werden, da diese Bereiche die zentralen Ursprünge von Data-Literacy-Kompetenzen darstellen. Um sowohl praktische Kompetenzen, die dieser Bereich beinhaltet, als auch deren theoretische fachliche Fundierung zu berücksichtigen, kann die im Bereich der informatischen Bildung in Deutschland bereits bewährte Zweiteilung in Inhalts- und Prozessbereiche (vgl. [GI08]) übernommen werden. Zur Ableitung der Inhaltsbereiche dienen dabei die hier ermittelten Inhalte der Data Science gemeinsam mit den Schlüsselkonzepten des Datenmanagements (vgl. [GR17]), während die Prozessbereiche aus dem Datenlebenszyklus (vgl. [GR17]) bestimmt werden können, der die prozessorientierten Aspekte des Umgangs mit Daten widerspiegelt. Ein Prototyp für ein solches Kompetenzmodell der Data Literacy, der in der weiteren Forschung noch stärker zu fundieren und zu evaluieren ist, kann auf dieser Basis wie in Abbildung 1 abgebildet aussehen.

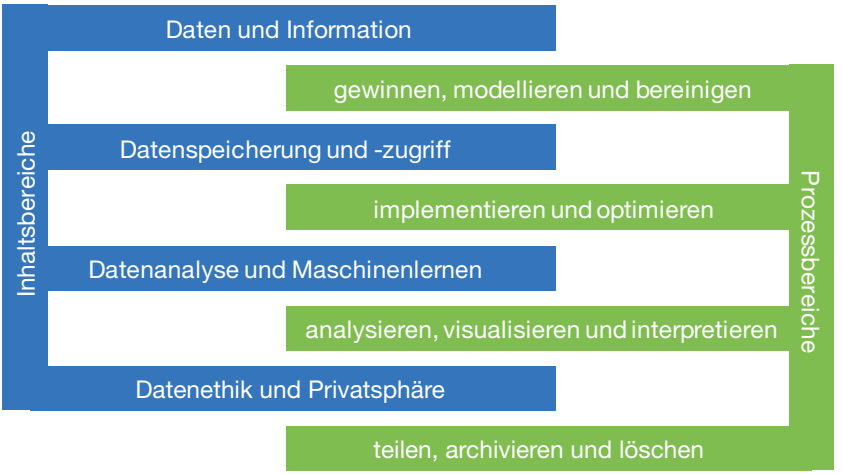


Abb. 1: Entwurf eines Data-Literacy-Kompetenzmodells

## 5 Zusammenfassung und Ausblick

Die durchgeführte Analyse gibt einen klaren Überblick über die Inhalte der Data Science und trägt zu ihrer Charakterisierung bei. Es zeigt sich, dass Data Science durch verschiedene informatische Aspekte geprägt ist, aber auch mathematische Aspekte nicht zu vernachlässigen sind. Aus informatischer Sicht kann die Data Science, basierend auf den in dieser Arbeit gewonnenen Ergebnissen, als Schnittmenge der beiden Themenbereiche *Datenbanken/Datenmanagement* sowie *Maschinenlernen/Datenanalyse* betrachtet werden: Das erste Gebiet berücksichtigt die eher statischen Aspekte, das zweite konzentriert sich auf die dynamischen. Zusätzlich zu diesen beiden Bereichen müssen jedoch auch Aspekte von *Big Data,* welches den speziellen Umgang mit großen und vielfältigen Datenmengen thematisiert, sowie die *Datenethik,* die gesellschaftliche und individuelle Wirkungen berücksichtigt, miteinbezogen werden. Durch die starke Verwurzlung der Data Science in der Informatik, die durch die hier beschriebene Arbeit expliziert wird, zeigt sich, dass die Datenwissenschaft ein Thema für die Forschung im Kontext der informatischen Bildung darstellt. Die hier beschriebene Explikation der Inhalte der Data Science kann einen Beitrag zu einer einheitlicheren Betrachtung dieser neuen Wissenschaft liefern, die bisher oft stark unterschiedlich geprägt ist, gleichzeitig aber auch zur gezielten Curriculumsentwicklung und zum Vergleich verschiedener Studiengänge in diesem Bereich eingesetzt werden.

Zusätzlich liefert sie Impulse für die weitere Forschung und kann, wie durch die skizzierte Entwicklung eines Data-Literacy-Kompetenzmodells gezeigt, als Basis für diese eingesetzt werden.

## Literaturverzeichnis


[Ca17] Cao, L.: Data Science: A Comprehensive Overview. ACM Comput. Surv. 50/3, 43:1–43:42, Juni 2017.

[DBW17] Demchenko, Y.; Belloum, A.; Wiktorski, T.: EDISON Data Science Framework: Part 2. Data Science Body of Knowledge (DS-BoK) Release 2, 2017.

[DMB17] Demchenko, Y.; Manieri, A.; Belloum, A.: EDISON Data Science Framework: Part 1. Data Science Competence Framework (CF-DS) Release 2, 2017.

[GI08] Gesellschaft für Informatik e. V., AK Bildungsstandards: Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I. LOG IN 150/151, 2008.

[GI18] Gesellschaft für Informatik e. V., PAK Data Science: Data Literacy und Data Science Education: Digitale Kompetenzen in der Hochschulausbildung, 2018.

[GR17] Grillenberger, A.; Romeike, R.: Key Concepts of Data Management: An Empirical Approach. In (Suero Montero, C.; Joy, M.; Eds.): Proceedings of the 17th Koli Calling International Conference on Computing Education Research, Koli, Finland, November 16–19, 2017, ACM, New York, NY, USA, 2017.

[HTT09] Hey, T.; Tansley, S.; Tolle, K.: The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, 2009.

[Ma10] Mayring, P.: Qualitative Inhaltsanalyse: Grundlagen und Techniken. Beltz, 2010.

[NI15] NIST Big Data Public Working Group: NIST Big Data Interoperability Framework: Volume 1, Definitions, 2015.

[Ri15] Ridsdale, C.; Rothwell, J.; Smit, M. et al.: Strategies and Best Practices for Data Literacy Education: Knowledge Synthesis Report, 2015.

[Ve17] Veaux, R. D. D.; Agarwal, M.; Averett, M. et al.: Curriculum Guidelines for Undergraduate Programs in Data Science. Annual Review of Statistics and Its Application 4/1, S. 15–30, 2017.

# Explorative Datenanalyse der Studierendenperformance in der Theoretischen Informatik

**Christiane Frede, Maria Knobelsdorf**
Universität Hamburg, Fachbereich Informatik
Vogt-Kölln-Straße 30
22527 Hamburg
frede@informatik.uni-hamburg.de

Universität Wien, Zentrum für LehrerInnenbildung/Fakultät für Informatik
Währinger Straße 29
1090 Wien
maria.knobelsdorf@univie.ac.at

**Abstract:** In diesem Artikel werden die Ergebnisse einer explorativen Datenanalyse über die Studierendenperformance in Klausur- und Hausaufgaben eines Einführungskurses der Theoretischen Informatik vorgestellt. Da bisher empirisch wenig untersucht ist, welche Probleme Studierenden in den Einführungskursen haben und die Durchfallquoten in diesen Kursen sehr hoch sind, soll auf diesem Weg ein Überblick gegeben werden. Die Ergebnisse zeigen, dass alle Studierenden unabhängig von ihrer Klausurnote die niedrigste Performance in den Klausur- und Hausaufgaben aufweisen, in denen formale Beweise gefordert sind. Dieses Ergebnis stärkt die Vermutung, dass didaktische Ansätze und Maßnahmen sich insbesondere auf das Erlernen formaler Beweismethoden fokussieren sollten, um Informatik-Studierende nachhaltiger dabei zu unterstützen, in Theoretischer Informatik erfolgreich zu sein.

**Keywords:** Theoretische Informatik, Explorative Datenanalyse, Formale Sprachen und Automaten, Studierendenperformance, Beweisaufgaben.

## 1 Einleitung und Verwandte Arbeiten

Die mathematischen und theoretischen Fachanforderungen in Einführungsveranstaltungen der Informatik-Bachelorstudiengänge stellen in Deutsch-

land nach wie vor eine große Herausforderung dar. Im Jahrgang 2010/2011 brachen 45 % aller Studierenden an deutschen Hochschulen ihr Studium u. a. aufgrund von Leistungsproblemen ab [He17]. Dies zeichnet sich vor allem in der Studieneingangsphase ab, in der die Abbruch- und Durchfallquoten in entsprechenden Einführungsmodulen sehr hoch sind. An vielen Standorten gilt dies insbesondere für Einführungskurse der Theoretischen Informatik. Es ist nicht ungewöhnlich, dass gerade hier die überwiegende Mehrheit der Informatik-StudienanfängerInnen mittelmäßige bis schlechte Leistungen erzielt und nur eine kleine Teilgruppe der Gesamtkohorte bei der Abschlussklausur sehr gute Ergebnisse erreicht. Es stellt sich daher die Frage, was die hohen Abbruch- und Durchfallquoten in Theorie-Kursen bedingt. Aus Sicht der Lehrenden werden oftmals folgende Vermutungen geäußert: Die Studierenden investieren nicht genug Zeit, sich mit den Themen und den Übungsaufgaben auseinanderzusetzen, sie sind nicht genügend motiviert, interessiert oder intelligent genug, um die Themen und Fachkonzepte zu durchdringen und zu verstehen (vgl. [CGM04][Pi10][SM11]). Diese Annahmen basieren in der Regel auf Beobachtungen der Lehrpersonen und Einzelgesprächen mit beteiligten Studierenden und TutorInnen, jedoch selten auf gesicherten Daten (vgl. [CGM04][Sc13). Aufbauend auf diesen Annahmen wurden verschiedene fachdidaktische Ansätze vorgestellt, um die Eingangslehre zu verbessern, z. B. [CGM04], [Ko07], [Sc13]. Diese Ansätze arbeiten mit teils sehr unterschiedlichen didaktischen Konzepten und Methoden, sind jedoch sehr erfolgreich in der Gestaltung einer konstruktivistischen Lehr-Lern-Umgebung. Zusammenfassend versuchen diese Ansätze, die Lehre der Theoretischen Informatik praktischer und anwendungsorientierter zu gestalten, um die Motivation der Studierenden für die mathematischen und abstrakten Themen zu erhöhen. Weitere existierende Einzelfallstudien [KF16][PL14] bieten hingegen Einblicke in die tatsächlichen Schwierigkeiten der Studierenden, ohne sich dabei auf ihre Fähigkeiten zur Selbstauskunft zu verlassen. Knobelsdorf und Frede (2016) haben Studierendengruppen dabei beobachtet, wie sie einen Reduktionsbeweis zur NP-Vollständigkeit bearbeitet haben [KF16]. Während dieser Studie hatten die Studierenden beispielsweise das Konzept der NP-Vollständigkeit durchaus verstanden, aber waren nicht in der Lage, den für die Aufgabe benötigten formalen Reduktionsbeweis zu entwickeln. Auch Armoni (2009) hat eine Studie zu Reduktionsbeweisen durchgeführt. Durch eine Dokumentenanalyse über Studierendenergebnisse hat er herausgefunden, dass es für Studierende schwierig war, Reduktion als eine Problemlösestrategie zu fassen, die vielseitig anwendbar ist [Ar09]. In einer anderen Studie analysierte Pillay (2010) drei Tests und wöchentliche Übungen

von dreizehn Studierenden [Pi10]. Da die Studierenden auch hier die größten Schwierigkeiten während des Problemlöseprozesses hatten, schlägt sie als didaktische Maßnahme vor, Studierende während der verschiedenen Schritte dieses Prozesses mittels Experten-Feedback dezidierter zu unterstützen. Die genannten Studien haben den Nachteil, dass sie sich nur auf einzelne Themen der zugehörigen Kurse beschränken. Eine umfassendere Analyse darüber, mit welchen Themen Studierende in einem Kurs über Algorithmen und Datenstrukturen besondere Probleme haben, hat Enström (2014) vorgelegt [En14]. Hierzu hat sie ein automatisiertes Beurteilungssystem für Studien verwendet und weitere Informationen aus Fragebögen, mündlichem Feedback der Studierenden und ihren Noten gewonnen. Als Ergebnis hält sie fest, dass die Studierenden Schwierigkeiten mit vielen Beweisaufgaben hatten, wobei explizit die Komplexitätsbeweise eine große Herausforderung darstellten. Bis auf die Arbeit von Enström beschränken sich alle erwähnten Studien auf ein oder wenige ausgewählte Themen, Konzepte und fachspezifische Kompetenzen aus dem entsprechenden Einführungskurs. Aus diesem Grund können diese Arbeiten keinen detaillierten und differenzierten Überblick darüber geben, welche Themen von allen Themen des entsprechenden Einführungskurses den Studierenden die meisten Probleme bereiten.

Dieser Artikel soll einen dezidierten Überblick darüber geben, welche Themen der Theoretischen Informatik den Studierenden die meisten Schwierigkeiten bereiten. Dafür wurde eine Studie durchgeführt, welche die Gesamtleistung der Studierenden untersucht. Als Datenquellen wurden die Ergebnisse aus den Hausaufgaben und der Abschlussklausur in einem Einführungskurs der Theoretischen Informatik von ca. 500 Informatik-Studierenden an der Technischen Universität Berlin aus dem Wintersemester 2016/2017 verwendet. Diese Daten wurden mittels explorativer Datenanalyse untersucht, die Einblicke in die Studierendenperformance liefert. Im folgenden Abschnitt wird das Forschungsdesign vorgestellt, sowie notwendige Informationen über den Aufbau des Kurses und die Datenquellen erläutert (Abschnitt 2). Anschließend werden die Ergebnisse präsentiert und interessante Erkenntnisse diskutiert (Abschnitt 3), bevor der Artikel in Abschnitt 4 mit einem Fazit schließt.

## 2 Forschungsdesign

In diesem Abschnitt wird der Forschungsansatz vorgestellt, die dafür verwendeten Datenquellen, sowie die Besonderheiten über den Aufbau der Lehrveranstaltung, aus der unsere Daten hervorgingen.

## 2.1 Forschungsmethode

Der Begriff der Explorativen Datenanalyse (EDA) wurde in den 1970er Jahren von Tukey vorgestellt [Tu77][TMH00]. Im Zuge dieses Forschungsansatzes werden oftmals leicht zugängliche Stichproben von Daten analysiert, um dadurch offene Fragestellungen zu beantworten. Über das betrachtete Forschungsfeld liegen dabei bisher keine Hypothesen, Modelle oder detaillierte Einblicke vor. Die EDA fasst hierbei verschiedene Techniken und Vorschläge zusammen, um die vorhandenen Daten zu strukturieren und zu untersuchen. Im Gegensatz zur Hypothesenüberprüfung, bei der Daten speziell auf die für das Testen der Hypothesen notwendigen Daten beschränkt werden, wird bei der EDA versucht, einen möglichst vollständigen und detaillierten Überblick des Datensatzes zu geben [DB16]. Um sich einem Datensatz zu nähern, können u. a. zusammenfassende Statistiken [CH86][Mo09] verwendet werden. Diese ermöglichen einen Überblick über die allgemeine Datenstruktur, wodurch erste Auffälligkeiten sichtbar werden können. Darauf aufbauend können anschließend weitere Analysen durchgeführt werden. Auch Diagramme können je nach Datensatz und Fragestellungen ein nützliches Werkzeug sein, um einen Überblick über die zusammengefassten Daten zu geben und Besonderheiten visuell darzustellen. Die EDA kann als Kombination aus verschiedenen Erkundungstechniken verwendet werden, um sich einem wenig erforschten Forschungsfeld zu nähern und die Grundlage für Hypothesen zu bilden, auf denen weiterführende Analysen durchgeführt werden können.

Für die statistische Analyse dieser Studie wurde die Software SPSS[1] verwendet. Während der explorativen Datenanalyse wurden die Daten zusammengefasst, um einen Überblick über die Datenstruktur zu bekommen und Auffälligkeiten zu erkennen. Diese wurden mit Hilfe von Tabellen und Diagrammen visualisiert. Zusätzlich dazu wurde Qualitative Inhaltsanalyse [Ma10] verwendet, um die Hausaufgaben nach Aufgabentypen zu kategorisieren.

## 2.2 Aufbau des Kurses und Datenquellen

Die für die Studie verwendeten Daten stammen aus einem Einführungskurs zur Theoretischen Informatik für Bachelorstudierende der Informatik der Technischen Universität Berlin, an dem ca. 500 Studierende teilgenommen

1 https://www.ibm.com/products/spss-statistics, zugegriffen am 27. 06. 2018

haben und der im Wintersemester 2016/2017 stattgefunden hat. Es wurden nicht reaktiv erhobene Daten verwendet, um eine Studierendenperformance zu untersuchen, wie sie in vergleichbaren Kursen an deutschen Universitäten auftritt. Als Indikatoren für die Performance wurden die bearbeiteten Hausaufgaben und die Ergebnisse der Abschlussklausur verwendet. Die Hausaufgaben geben einen Überblick über die Lernleistung während des Kurses, während die Klausurergebnisse ein möglicher Indikator für die bis dahin entwickelten bzw. erreichten Fachkompetenzen darstellt.

Der betrachtete Einführungskurs zur Theoretischen Informatik behandelt die Themen Formale Sprachen und Automaten und ist für alle Informatik-Studierenden verpflichtend, während die Anwesenheit in Vorlesung und Übungsgruppen nicht verpflichtend ist. Parallel dazu nahmen die Studierenden an einem verpflichtenden Mathematikkurs über Lineare Algebra und Analysis teil. Um den Kurs zu bestehen, mussten die Studierenden sog. Portfolio-Punkte sammeln. Dafür konnten sie verteilt über das Semester vier Blöcke von Hausaufgaben (insgesamt 31 Aufgaben) in Kleingruppen bearbeiten und zur Prüfung abgeben. Jeder Hausaufgabenblock konnte maximal fünf Punkte zum Portfolio beitragen, so dass insgesamt 20 Punkte durch Hausaufgaben zu erreichen waren. Weitere 30 Punkte konnten über einen online Multiple-Choice Test in der Mitte des Semesters gesammelt werden. Die sechs Aufgaben der Abschlussklausur am Ende des Semesters konnten bis zu 50 Punkte zum Portfolio beitragen. Das Modul galt als bestanden, wenn am Ende insgesamt mehr als 49 Portfolio-Punkte erreicht wurden.

Der erste Hausaufgabenblock wurde von 571 Studierenden abgegeben, während sich die Zahl der Abgaben für den vierten und damit letzten Hausaufgabenblock auf 491 verringert hatte. Es wurden auch nicht vollständige Hausaufgabenblöcke in der Analyse berücksichtigt, da fast alle Studierendengruppen zwischendurch immer wieder einzelne Aufgaben nicht bearbeitet haben. Die Abgaben wurden dennoch einbezogen, da keine Daten über die Gründe erhoben wurden, warum die Studierenden bestimmte Aufgaben nicht bearbeitet haben. Mangelnde Zeit, mangelndes Verständnis oder mangelndes Interesse sind nur einige der möglichen Gründe.

Unabhängig von den Hausaufgaben haben 419 Studierende an der Abschlussklausur teilgenommen. Die Wiederholungsklausur vier Wochen später wurde von sechzehn Studierenden bearbeitet, wobei dies für acht Studierende der erste Versuch war. Damit haben 427 Studierende der ursprünglichen 571 Studierenden an einer der beiden Klausuren teilgenommen (75 %). Von diesen 427 Studierenden, haben 295 (69 %) mit mehr als 49 % der Punkte bestanden. Nach den endgültigen Portfoliopunkten gerechnet, haben 339 (59 %)

von den ursprünglichen 571 Studierende den Kurs bestanden und 232 Studierende (41 %) nicht.

Die Daten wurden nach Absprache mit der Datenschutzbeauftragten der Technischen Universität Berlin nur anonymisiert verwendet, so dass kein Rückschluss auf einzelne Personen möglich war und ist. Außerdem waren die Autorinnen in keiner Form in die Lehrveranstaltung involviert.

## 3 Ergebnisse und Diskussion

Im Folgenden werden zuerst die Ergebnisse der Abschlussklausur vorgestellt und diskutiert, um mögliche Auffälligkeiten in der Studierendenperformance zu erkennen. Die Analyse der Performance wird anschließend mit den Hausaufgaben weitergeführt. Während dieser explorativen Datenanalyse werden die einzelnen Schritte der Analyse einzeln diskutiert, bevor anschließend die nachfolgenden Schritte präsentiert und diskutiert werden.

### 3.1 Ergebnisse der Abschlussklausur

Zu Beginn der Analyse wurden die Klausurergebnisse der Studierenden betrachtet. Abbildung 1 zeigt die Verteilung der erreichten Punkte auf die 419 Studierenden, die an der Abschlussklausur teilgenommen haben.

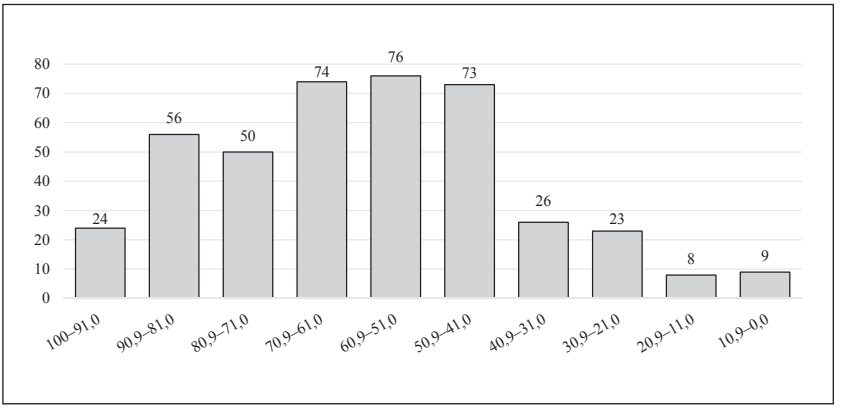


Abb. 1: Verteilung der erreichten Punkte in der Abschlussklausur: Die x-Achse zeigt den Punktebereich, die y-Achse die Anzahl der Studierenden.

Die mehr oder weniger typische Gaußkurve der Klausurergebnisse lässt annehmen, dass die Schwierigkeit der Abschlussklausur angemessen war. Um die Notenverteilung darzustellen, wird der allgemeine Notenschlüssel des Fachbereichs Informatik der Technischen Universität Berlin verwendet:

- Note 1: Notwendige Punkte 100–81. Erreicht von 80 Studierenden.
- Note 2: Notwendige Punkte 80–70. Erreicht von 67 Studierenden.
- Note 3: Notwendige Punkte 69–59. Erreicht von 88 Studierenden.
- Note 4: Notwendige Punkte 58–50. Erreicht von 61 Studierenden.
- Note 5: Unter 50 Punkten. Erreicht von 123 Studierenden.

Von den insgesamt 419 Studierenden haben 80 Studierende eine sehr gute Note erzielt, während 216 Studierende eine gute bis ausreichende Leistung erreicht haben. Um die Frage zu beantworten, ob sich diese Gesamtleistung in den Abschlussklausuren auch in den Einzelaufgaben wiederspiegelt, wurden die sechs Klausuraufgaben detaillierter betrachtet. Hierfür wurden die Studierenden abhängig von ihrer Note in der Abschlussklausur in die Gruppen 1 bis 5 aufgeteilt. Da für jede Aufgabe eine unterschiedliche maximale Punktzahl erreicht werden konnte, wurde diese Punktzahl in Prozentwerte umgerechnet. Anschließend konnte die durchschnittliche Prozentzahl der erreichten Punkte pro Notengruppe und Aufgabe berechnet werden, so dass die Performance in den einzelnen Aufgaben miteinander verglichen werden kann. Tab. 1 zeigt die Werte pro Notengruppe und Aufgabennummer und K1 bis K6 stellen die sechs Klausuraufgaben dar.

Tab. 1: Durchschnittlich erreichte Punkte in Prozent für jede Klausuraufgabe nach Notengruppe

| Gruppe / Aufgabe | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **K1** | 91 % | 86 % | 78 % | 74 % | 43 % |
| **K2** | 95 % | 90 % | 82 % | 79 % | 44 % |
| **K3** | 95 % | 89 % | 81 % | 78 % | 44 % |
| **K4** | 74 % | 52 % | 29 % | 16 % | 5 % |
| **K5** | 89 % | 64 % | 43 % | 27 % | 10 % |
| **K6** | 76 % | 54 % | 41 % | 32 % | 14 % |

In Tab. 1 ist wie erwartet auf den ersten Blick zu erkennen, dass Gruppe 1 in jeder der sechs Klausuraufgaben die höchsten Werte hat. Die Gruppen 2 und 3 haben durchschnittliche Werte, während die Gruppen 4 und 5 die niedrigsten Werte aufweisen. Auf den zweiten Blick ist jedoch zu erkennen, dass die Gruppen 1 bis 4 in den ersten drei Aufgaben gut bis sehr gut abgeschnitten haben, während in den Aufgaben K4, K5 und K6 ein z. T. deutlicher Punktabfall für alle Notengruppen sichtbar ist. Um genauer zu verstehen, was die Aufgaben K4, K5 und K6 von den ersten drei Aufgaben unterscheidet und was der Grund für den Punktabfall sein könnte, müssen die Themen und Aufgabenstellungen der Klausuraufgaben betrachtet werden:

- K1: Das Thema war reguläre Sprachen. Angeben eines nicht-deterministischen Automaten (NFA), einer Grammatik für eine gegebene Sprache und Ableitungen von Wörtern.
- K2: Das Thema war Automaten. Es war notwendig, einen deterministischen Automaten (DFA) von einem gegebenen NFA zu konstruieren.
- K3: Minimierung eines DFA. Es war notwendig den Table-filling-Algorithmus anzuwenden, um einen DFA zu minimieren. Außerdem sollten Äuqivalenzklassen angegeben und der DFA visualisiert werden.
- K4: Reguläre Sprachen. Einen Beweis mit Hilfe des Pumping Lemmas entwickeln. Myhill-Nerode Äquivalenzklassen für eine Sprache angegeben.
- K5: Kontextfreie Sprachen. Es musste eine Typ-2-Grammatik und ein Kellerautomat (PDA) angegeben werden.
- K6: Kontextfreie Sprachen. Es mussten Ableitungen eines PDA angegeben werden und ein Beweis geführt werden, dass eine Sprache nicht regulär ist.

Werden nun die letzten drei mit den ersten drei Aufgaben verglichen, ist festzustellen, dass es in K4 und K6 u. a. notwendig war einen formalen Beweis zu entwickeln. In K1, K2 und K3 hingegen mussten Grammatiken und Wörter angegeben, Automaten konstruiert sowie spezifische Algorithmen angewendet werden. Hierbei wurde mit vorgegebenen Sprachen und Grammatiken und oftmals an einem konkreten Beispiel gearbeitet. Für alle Notengruppen ist K4 die Aufgabe mit dem höchsten Punktabfall. Hier mussten die Studierenden das Pumping Lemma im Rahmen eines formalen Beweises anwenden. Daraus lässt sich insgesamt schließen, dass alle Studiereden unabhängig von ihrer Klausurnote Schwierigkeiten mit dieser Art der Aufgabenstellung hatten. Dieses Ergebnis unterstützt die Ergebnisse anderer Studien, dass Studierende besondere Schwierigkeiten mit Beweisaufgaben haben (vgl. Abschnitt 2).

## 3.2 Hausaufgabenperformance

Ausgehend von den Ergebnissen aus Abschnitt 3.1 stellte sich die Frage, ob die Studierenden schon vor der Abschlussklausur in den 31 Hausaufgaben Schwierigkeiten hatten, formale Beweis zu entwickeln. Wie im vorherigen Abschnitt wird auch hier bei den Hausaufgabenpunkten mit Prozentwerten gearbeitet und die Verteilung der Punkte bezogen auf die Notengruppe in der Klausur berechnet (Tab. 2).

Auch wenn der Unterschied zwischen den einzelnen Notengruppen dieses Mal geringer ausfällt, ist auch in Tab. 2 zu erkennen, dass Gruppe 1 in jeder Aufgabe die beste Performance aufweist. Insgesamt sind die auffälligsten Punktabfälle für die Aufgaben A2, A5, A6, A7, A8, A10, A15, A20, A21, A24 und A29 zu sehen (die entsprechenden Zeilen sind in Tab. 2 fett markiert). Bei diesen elf Aufgaben liegt der Punktabfall für jede Notengruppe bis zu 30 % unter der sonstigen durchschnittlichen Performance pro Gruppe. Um auch zwischen diesen Aufgaben Unterschiede und Gemeinsamkeiten erkennen zu können, wurden die Themen und Aufgabenstellungen dieser elf einzelnen Aufgaben analysiert und zusammengefasst dargestellt:

- A2: Mengen. Beweisen von Eigenschaften.
- A5: Aussagenlogik. Beweisen von Variablenbelegungen.
- A6: Prädikatenlogik. Eine gegebene Aussage für zwei Prädikate beweisen.
- A7: Prädikatenlogik. Die ersten Schritte für einen Widerspruch angeben.
- A8: Mengen. Einen Induktionsbeweis über eine Zahlenmenge führen.
- A10: Relationen und Funktionen. Ordnungen beweisen.
- A15: Äquivalenzklassen. Beweisen, dass zwei Variablen äquivalent sind.
- A20: Wörter und Sprachen. Induktionsbeweis über ein gegebenes Alphabet und enthaltende Wörter.
- A21: Grammatiken. Ableitungen für Wörter und Sprachen angeben.
- A24: Reguläre Sprachen: Einen Beweis mit dem Pumping Lemma entwickeln.
- A29: Reguläre Sprachen: Alle Myhill Nerode-Äquivalenzklassen angeben.

Bei einem ersten Vergleich der Aufgaben wird deutlich, dass ein formaler Beweis in acht von elf Aufgaben gefordert ist. Nur in den Aufgaben A7, A21 und A29 musste kein formaler Beweis entwickelt werden. Thematisch hingegen werden verschiedene Themen abgedeckt. In Aufgabe A24 war wie in Klausuraufgabe K4 ein Beweis mit Hilfe des Pumping Lemmas gefordert.

Tab. 2: Durchschnittlich erreichte Prozente für jede Hausaufgabe aufgeteilt nach Notengruppe

| Gruppe / Aufgabe | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **A1** | 95 % | 94 % | 91 % | 90 % | 86 % |
| **A2** | **77 %** | **69 %** | **69 %** | **57 %** | **70 %** |
| **A3** | 92 % | 86 % | 87 % | 83 % | 90 % |
| **A4** | 85 % | 85 % | 83 % | 82 % | 79 % |
| **A5** | **80 %** | **69 %** | **63 %** | **51 %** | **56 %** |
| **A6** | **50 %** | **46 %** | **46 %** | **40 %** | **41 %** |
| **A7** | **69 %** | **59 %** | **64 %** | **53 %** | **56 %** |
| **A8** | **79 %** | **79 %** | **72 %** | **65 %** | **68 %** |
| **A9** | 92 % | 92 % | 84 % | 83 % | 85 % |
| **A10** | **77 %** | **72 %** | **69 %** | **65 %** | **68 %** |
| **A11** | 86 % | 80 % | 74 % | 72 % | 71 % |
| **A12** | 87 % | 82 % | 69 % | 70 % | 63 % |
| **A13** | 97 % | 96 % | 85 % | 88 % | 87 % |
| **A14** | 84 % | 74 % | 64 % | 58 % | 63 % |
| **A15** | **66 %** | **46 %** | **45 %** | **43 %** | **35 %** |
| **A16** | 83 % | 70 % | 67 % | 64 % | 59 % |
| **A17** | 98 % | 94 % | 89 % | 89 % | 91 % |
| **A18** | 90 % | 77 % | 75 % | 66 % | 68 % |
| **A19** | 99 % | 91 % | 90 % | 84 % | 87 % |
| **A20** | **72 %** | **62 %** | **60 %** | **49 %** | **50 %** |
| **A21** | **77 %** | **64 %** | **64 %** | **54 %** | **58 %** |
| **A22** | 92 % | 82 % | 79 % | 71 % | 71 % |
| **A23** | 82 % | 78 % | 73 % | 63 % | 66 % |
| **A24** | **82 %** | **70 %** | **65 %** | **52 %** | **63 %** |
| **A25** | 97 % | 86 % | 86 % | 80 % | 80 % |
| **A26** | 89 % | 78 % | 77 % | 72 % | 76 % |
| **A27** | 99 % | 91 % | 88 % | 79 % | 85 % |
| **A28** | 99 % | 93 % | 92 % | 88 % | 50 % |
| **A29** | **76 %** | **62 %** | **57 %** | **49 %** | **30 %** |
| **A30** | 86 % | 75 % | 67 % | 47 % | 35 % |
| **A31** | 99 % | 93 % | 88 % | 82 % | 50 % |

Bei A24 konnte zwar ein starker Punktabfall registriert werden, doch dieser war nicht so deutlich wie bei K4. Auch hier kann es verschiedene Gründe für den Unterschied geben, für die in dieser Analyse keine Daten erhoben wurden, z. B. Hausaufgabenabgabe in Gruppen, weniger Zeitdruck als bei der Abschlussklausur.

Zusätzlich stellt sich nun die Frage, was die Aufgaben mit Punktabfall von denen unterscheidet, die eine höhere Performance aufweisen. Aus Platzgründen werden die Informationen über Themen und Aufgabentyp für alle Aufgaben zusammengefasst dargestellt (Tab. 3). Die Aufgaben mit auffälligem Punktabfall sind erneut fett markiert. Um die Aufgaben einem Thema zuzuordnen, wurden die korrespondieren Themen aus der Formelsammlung des Kurses verwendet. Durch Anwendung einer zusammenfassenden qualitativen Inhaltsanalyse angelehnt an Mayring [Ma10] wurden die Aufgaben außerdem einem der folgenden Aufgabentypen zugeordnet: *Beweisen* (Entwickeln eines Beweises), *Angeben* (von z. B. Ableitungen, Sprachen, Relationen, Grammatiken, etc.), *Konstruieren* (von Automaten), *Berechnen* (von z. B. Mengen). Diese ersten Kategorien bilden eine abstrakte Zusammenfassung von möglichen Gemeinsamkeiten der Aufgabenstellungen.

Da auch hier drei Beweisaufgaben (A3, A4, A12) zu finden sind, stellt sich die Frage, was diese Beweisaufgaben von denen unterscheidet, bei denen ein auffälliger Punktabfall zu beobachten war. In A3 und A4 war die Entwicklung eines Beweises gefordert, der als schematisch und weniger formal bewertet werden kann (Wahrheitstabellen, Äquivalenzumformungen). In Aufgabe A12 sollte Kardinalität bewiesen werden. Die hier betrachteten Daten können allerdings nicht verwendet werden, um eine Aussage darüber zu treffen, was A12 von den anderen Beweisaufgaben unterscheidet. Der einzige ersichtliche Unterschied besteht darin, dass A12 der einzige Beweis zu dem Thema „Funktionen/Abbildungen“ war. Hier ist eine genauere Analyse der verschiedenen Beweisarten nötig.

Insgesamt wurden elf Aufgaben mit auffälligem Punktabfall erkannt. Von diesen elf Aufgaben war in acht Aufgaben eine formale Beweisentwicklung gefordert. Beweisart und Thema unterschieden sich zum Großteil für die einzelnen Aufgaben. In zwei der drei Beweisaufgaben, in denen die Performance der Studierenden höher war, musste ein schematischer und weniger formaler Beweis entwickelt werden. Zusammenfassend lässt sich sagen, dass sich die geringe Performance in den Beweisaufgaben der Klausur auch in den Hausaufgaben wiederfinden lässt.

Tab. 3: Kategorisierung der Hausaufgaben ohne auffälligen Punktabfall

| Aufgabe | Thema (Formelsammlung) | Aufgabentyp |
|---|---|---|
| **A1** | Mengen | Berechnen |
| **A2** | **Mengen** | **Beweisen** |
| **A3** | Aussagenlogik | Beweisen |
| **A4** | Aussagenlogik | Beweisen |
| **A5** | **Aussagenlogik** | **Beweisen** |
| **A6** | **Prädikatenlogik** | **Beweisen** |
| **A7** | **Mengen** | **Angeben** |
| **A8** | **Mengen** | **Beweisen** |
| **A9** | Relationen/Ordnungen | Angeben |
| **A10** | **Relationen/Ordnungen** | **Beweisen** |
| **A11** | Funktionen/Abbildungen | Angeben |
| **A12** | Funktionen/Abbildungen | Beweisen |
| **A13** | Relationen/Ordnungen | Angeben |
| **A14** | Relationen/Ordnungen | Angeben |
| **A15** | **Funktionen/Abbildungen** | **Beweisen** |
| **A16** | Funktionen/Abbildungen | Angeben |
| **A17** | Wörter/Sprachen | Angeben |
| **A18** | Wörter/Sprachen | Angeben |
| **A19** | Wörter/Sprachen | Angeben |
| **A20** | **Wörter/Sprachen** | **Beweisen** |
| **A21** | **Grammatiken** | **Angeben** |
| **A22** | Grammatiken | Angeben |
| **A23** | Grammatiken | Angeben |
| **A24** | **Reguläre Sprachen** | **Beweisen** |
| **A25** | Automaten | Angeben |
| **A26** | Automaten | Konstruieren |
| **A27** | Minimierung von Automaten | Angeben |
| **A28** | Minimierung von Automaten | Konstruieren |
| **A29** | **Reguläre Sprachen** | **Angeben** |
| **A30** | Reguläre Sprachen | Angeben |
| **A31** | Minimierung von Aut. | Angeben |

## 4 Fazit und Zusammenfassung

In diesem Artikel wurde explorative Datenanalyse verwendet, um die Leistung von Informatik-Studierenden in einem Einführungskurs der Theoretischen Informatik differenzierter zu bewerten. Für die Analyse wurden die Studierenden abhängig von ihrer Note in der Abschlussklausur in die Gruppen 1 bis 5 aufgeteilt. Dadurch wurden interessante Muster und Auffälligkeiten in den Klausurergebnissen und Hausaufgaben der Studierenden entdeckt. Zusammenfassend gab es für alle Gruppen unabhängig von ihrer Note in der Abschlussklausur eine schlechtere Performance in denselben Klausur- und Hausaufgaben. Dies waren zum Großteil Aufgaben, in denen ein formaler Beweis gefordert war, während die Aufgaben sich thematisch unterschieden. Damit zeigt die Analyse der vorliegenden Daten, dass in einem Einführungskurs der Theoretischen Informatik formale Beweise unabhängig vom Thema oder zugrundeliegendem Fachkonzept für alle Informatik-Studierenden die größten Herausforderungen darstellen. Die Leistung der Studierenden ist dabei umso schlechter, je formaler der Beweis ist. Dieses Ergebnis unterstützt die diskutierten Ergebnisse der Einzelfallstudien aus Abschnitt 2. Es deutet außerdem darauf hin, dass die Studierenden nicht per se Schwierigkeiten mit Themen der Theoretischen Informatik als solcher haben, sondern mit formalen Beweistechniken. Fachdidaktische Ansätze und Maßnahmen zur Senkung von Durchfallquoten in entsprechenden Lehrveranstaltungen, sollten sich daher explizit auf die Vermittlung und intensive Einübung formaler Beweismethoden fokussieren. Dies steht im Gegensatz zu den bisherigen Ansätzen, die vorschlagen, dass die theoretischen Konzepte praktischer und anwendungsorientierter gelehrt werden sollten.

## Danksagung

Wir möchten uns bei Prof. Dr. Uwe Nestmann und seiner Arbeitsgruppe „Modelle und Theorie Verteilter Systeme“ am Institut für Softwaretechnik und Theoretische Informatik der Technischen Universität Berlin für die Unterstützung unserer Studie bedanken.

## Literaturverzeichnis


[Ar09] Armoni, M.: Reduction in CS: A (mostly) quantitative analysis of reductive solutions to algorithmic problems. Journal on Educational Resources in Computing (JERIC) 8.4: 11, 2009.

[CGM04] Chesñevar, C.; González, M.; Maguitman, A.: Didactic strategies for promoting significant learning in formal languages and automata theory. In: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, ITiCSE '04, S. 7–11, 2004.

[Ch86] Chatfield, C.: Exploratory data analysis. European journal of operational research 23.1: 5–13, 1986.

[DB16] Döring, N.; Bortz, J.: Forschungsmethoden und Evaluation, 5. Auflage, Springer, 2016.

[En14] Enström, E.: On difficult topics in theoretical computer science education. Diss. KTH Royal Institute of Technology. 2014.

[He17] Heublein, U. et al.: Zwischen Studienerwartungen und Studienwirklichkeit. DZHW, Hannover, 2017.

[HMU01] Hopcroft, J. E.; Motwani, R.; Ullman J. D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 2. Auflage, 2001.

[KF16] Knobelsdorf, M.; Frede, C.: Analyzing Student Practices in Theory of Computation in Light of Distributed Cognition Theory. In: Proceedings of the Conference on International Computing Education Research (ICER '16). S. 73–81, ACM, 2016.

[Ko07] Korte, L. et al.: Learning by Game-building: A Novel Approach to Theoretical Computer Science Education. In: SIGCSE Bull. 03/07, S. 53–57, 2007.

[Ma10] Mayring, P.: Qualitative Content Analysis. Forum: Qualitative Social Research 2/2010, http://www.qualitative-research.net/index.php/fqs/article/view/1089, Abrufdatum: 19. 05. 2018.

[Mo09] Morgenthaler, S.: Exploratory data analysis. Wiley Interdisciplinary Reviews: Computational Statistics 1.1: 33–44, 2009.

[Pi10] Pillay, N.: Learning difficulties experienced by students in a course on formal languages and automata theory. SIGCSE Bulletin 41.4: 48–52, 2010.

[PL14] Parker, M.; Lewis, C.: What Makes big-O Analysis Difficult: Understanding How Students Understand Runtime Analysis. J. Comput. Sci. Coll., 04/14, S. 164–174, 2014.

[Sc13] Schäfer, A. et al.: From boring to scoring – a collaborative serious game for learning and practicing mathematical logic for computer science education. Computer Science Education 23: 87–111, 2013.

[SM11] Schulmeister, R.; Metzger, Ch.: Der Workload im Bachelor: Zeitbudget und Studierverhalten. Eine empirische Studie. Waxmann. Münster, 2011.

[Tu77] Tukey, J. W.: Exploratory data analysis. Vol. 2, 1977.

[TMH00] Tukey, J. W.; Mosteller, F.; Hoaglin, D. C.: Understanding robust and exploratory data analysis. Wiley Classics Library ed New York, 2000.

# Wissenschaftliches Arbeiten lernen – Eine Problemanalyse

**Christoph Greven und Ulrik Schroeder**
RWTH Aachen, Lehr- und Forschungsgebiet Informatik 9
Ahornstraße 55
52074 Aachen
greven@informatik.rwth-aachen.de
schroeder@informatik.rwth-aachen.de

**Abstract:** Die Lehre von wissenschaftlichem Arbeiten stellt einen zentralen Aspekt in forschungsorientierten Studiengängen wie der Informatik dar. Trotz diverser Angebote werden mittel- und langfristig Mängel in der Arbeitsqualität von Studierenden sichtbar. Dieses Paper analysiert daher das Profil der Studierenden, deren Anwendung des wissenschaftlichen Arbeitens, und das Angebot von Proseminaren zum Thema „Einführung in das wissenschaftliche Arbeiten" einer deutschen Universität. Die Ergebnisse mehrerer Erhebungen zeigen dabei diverse Probleme bei Studierenden auf, u. a. bei dem Prozessverständnis, dem Zeitmanagement und der Kommunikation.

**Keywords:** Wissenschaftliches Arbeiten, Hochschullehre, Informatik, Forschung, Informationskompetenz, forschungsorientiertes Lernen, Analyse.

## 1 Einleitung

In der heutigen Informationsgesellschaft wird es immer wichtiger, sich ständig in neue Wissensdomänen einarbeiten zu können. Die grundlegenden Kompetenzen dafür sind vielseitig und umfassen beispielsweise Medienkompetenz, Digitalkompetenz, Recherchekompetenz, und ähnliche, die nur schwer voneinander abgegrenzt werden können. Anwendung finden diese in allen Bereichen des lebenslangen Lernens, beispielsweise im Berufsleben oder im akademischen Bereich. Speziell beim wissenschaftlichen Arbeiten in der Forschung ist ein kritischer Umgang mit Informationen unerlässlich [Tr16]. Insbesondere schnelllebige Fachbereiche wie die Informatik, in denen die Halbwertszeit von Wissen immer weiter sinkt und gleichzeitig die Infor-

mationsflut immer weiter zunimmt, stellen besondere Herausforderungen dar. Entsprechend sollte die grundlegende Informationskompetenz, die den kritischen Umgang mit Informationen zusammenfasst [Ha16], schon früh im Studium gefördert werden. Entsprechende Lehrveranstaltungen werden in forschungsorientierten Studiengängen oft in Form von Seminaren realisiert, in denen Studierende sich in unbekannte Themen einarbeiten, Informationen aggregieren, Erkenntnisse erlangen, und die resultierenden Mehrwerte für die Forschungsgemeinschaft kommunizieren sollen. Letzteres geschieht häufig in Form von schriftlichen, wissenschaftlichen Beiträgen (Paper) und einem Vortrag. Solche Seminare sind auch in den Bachelor- und Master-Informatikstudiengängen der RWTH Aachen verankert. Sie gehören zu den Pflichtveranstaltungen und sollen vor allem für bevorstehende Abschlussarbeiten qualifizieren, sodass die Konzentration der Lernenden dann komplett auf den Inhalten liegen kann. Diesen voraus, sieht der Lehrplan Anfang des Bachelorstudiums ein Proseminar („Einführung in das wissenschaftliche Arbeiten") vor, in denen Studierende an den Prozess des wissenschaftlichen Arbeitens herangeführt werden sollen. Beobachtbare Qualitätsmängel in den Ergebnissen der Proseminare führen im späteren Verlauf des Studiums zu massiven Problemen.

Im Nachfolgenden wird daher näher auf Informationskompetenz als Kernkompetenz wissenschaftlichen Arbeitens eingegangen und es werden Ergebnisse von Erhebungen in den genannten Proseminaren vorgestellt, um die auftretenden Schwierigkeiten zu identifizieren.

## 2 Informationskompetenz als Kern wissenschaftlichen Arbeitens

Verschiedene internationale Standards definieren, welche Fähigkeiten von informationskompetenten Individuen zu erwarten sind. Sie unterscheiden sich dabei nur wenig und verstehen Informationskompetenz (IK) als „Fähigkeit, Informationen selbstorganisiert und problemlösungsorientiert effizient zu suchen, zu finden, zu bewerten und effektiv zu nutzen" [De09]. Zu deren Förderung existieren bundesweit viele Angebote. Alleine für 2017 listet die Plattform informationskompetenz.de[1] über 17 000 IK-Maßnahmen auf. Aus ihnen geht ein relativ eindeutiges Bild gängiger Lehrveranstaltungen hervor

1 Vermittlungs- und Forschungsaktivitäten zur Informationskompetenz. http://www.informationskompetenz.de, zugegriffen am 27. 06. 2018

[In17]. Dabei handelt es sich um einmalige Sitzungen bis 90 Minuten mit bis zu 10 Personen, die eigenständig von Bibliotheken angeboten werden. Ihre Ausrichtung ist fachübergreifend und findet in Form von Präsentationen, Führungen und Vorträgen statt. Die Angebote sind freiwillig und haben als Zielgruppe Bachelorstudierende, wobei die Veranstaltungen einen einführenden Charakter haben und den Schwerpunkte auf die Bibliotheksnutzung und einzelne Kataloge oder Datenbanken legen [In16]. Über hochschulinterne Angebote hinaus, wurden in den letzten Jahren einige Anwendungen zur Unterstützung und Förderung von IK entwickelt. Neben traditionellen Büchern, gibt es online Guidelines, Video Tutorials, Quizze, und ganze MOOCs; z. B. LOTSE[2], Compas[3], oder VISION[4]. Viele dieser Angebote sind allgemein gehalten und gehen nicht auf die speziellen Charakteristika eines Fachgebietes ein und sind überwiegend von der eigentlichen Anwendung gelöst. Daneben gibt es nur wenige simulationsähnliche Umgebungen, die auf fachspezifische Situationen eingehen, z. B. das FIT der UB Heidelberg[5]. Spezielle Angebote für das Studienfach Informatik konnten dabei nicht ausgemacht werden. Allen Angeboten gemein ist aber, dass sie einen Kontextwechsel zwischen der Lehre und der Anwendung erfordern, und nicht im eigentlichen Arbeitsprozess selber unterstützen. Der Transfer in eine fachspezifische, konkrete Anwendungssituation ist für Studierende daher oft schwierig.

## 2.1 (Fort-)Bildungsangebote der RWTH

Auch die RWTH Aachen realisiert Bildungsangebote zum Thema Informationskompetenz, im Wesentlichen durch die Hauptbibliothek der Universität. Diese sind den oben beschriebenen gängigen IK-Angeboten sehr ähnlich. Sie beschränken sich hauptsächlich auf die Einführung in die Bibliothek und die verfügbaren Beschaffungsoptionen für Literatur. Im Fokus stehen dabei der bibliothekseigene Katalog und die lokalen Angebote. Andere Kurse gehen gezielt auf z. B. Literaturverwaltung mit entsprechenden Softwareprogrammen

2 LOTSE – Wegweiser zur Literatursuche und zum wissenschaftlichen Arbeiten. https://www.ulb.uni-muenster.de/ulb-tutor/lotse/, zugegriffen am 27. 06. 2018

3 Compas – Strukturiertes Forschen im Web. http://www.compas.infoclio.ch/de, zugegriffen am 27. 06. 2018

4 VISION – VIrtual Services for Information Online. http://www.vision.tu-harburg.de, zugegriffen am 27. 06. 2018

5 Fachbezogenes Informationskompetenz-Training der Universitätsbibliothek Heidelberg. https://www.uni-heidelberg.de/studium/a-z/fit_ub.html, zugegriffen am 27. 06. 2018

ein. Auch hier sind alle Angebote freiwillige, einmalige Kurse. Spezifisch für den Studiengang Informatik existieren darüber hinaus weitere Angebote: Proseminare. Vorteil gegenüber der Standard-IK-Angebote ist eine direkte Integration in das Curriculum als spezialisiertes, fachspezifisches Angebot. Entsprechend handelt es sich auch um eine Pflichtveranstaltung. Proseminare dienen der „Einführung in das wissenschaftliche Arbeiten" und sind im 3. Bachelorsemester verankert. Sie werden von den verschiedenen Instituten der Informatik selber durchgeführt, und bieten entsprechend diverse Themen an. Die Gruppengrößen variieren und umfassen in der Regel 20–30 Studierende. Diese sollen sich während eines Semesters in ein unbekanntes Themengebiet der Informatik und den Stand der Forschung einarbeiten, diesen reflektieren, und schließlich wiedergeben. Dies geschieht über eine schriftliche Ausarbeitung in Form eines wissenschaftlichen Papers und eines Abschlussvortrages. Letzterer wird meistens zum Semesterende in einer Blockveranstaltung gehalten. In der Regel haben Teilnehmende zwei bis drei Kontaktpunkte mit einem Betreuer im Semester, bei denen sie Rückmeldung zu ihrem bisherigen Fortschritt bekommen. Dieses Feedback fokussiert größtenteils auf die schriftliche Ausarbeitung und deren Strukturierung. Darüber hinaus gibt es meist weitere, freiwillige Kontakt- und Feedbackangebote. Gewöhnlich erhalten Teilnehmende auch Startliteratur als Einstiegspunkt, bspw. in Form eines Papers oder Buchkapitels, an dem sie sich orientieren können. Als Teil des Proseminars müssen Studierende an einer Schulung „Einführung in die Literaturrecherche" der Informatikbibliothek teilnehmen. Diese setzt vor allem Schwerpunkte bei der Nutzung der Bibliothek selber und bei der lokalen Beschaffung von Literatur.

## 3 Erhebung in Proseminaren der Informatik

Um zukünftig genauer auf die Bedürfnisse von Studierenden bei der Lehre wissenschaftlichen Arbeitens eingehen zu können, werden nachfolgend die Ergebnisse einer Studie in Proseminaren als Lehrveranstaltung zur „Einführung in das wissenschaftliche Arbeiten" vorgestellt. Es handelt sich um ein exploratives und deskriptives Vorgehen, mit dem im Wesentlichen folgende Fragen beantwortet werden sollen: Wodurch ist das Profil der Zielgruppe (Studienanfänger der Informatik) charakterisiert? Wie findet der Prozess des wissenschaftlichen Arbeitens und dessen Lehre im Rahmen der Proseminare statt? Und welche Probleme treten dabei auf?

Die in diesem Paper vorgestellten Ergebnisse sind Teil mehrerer Erhebungen zum Thema wissenschaftliches Arbeiten in Proseminaren. Durch Triangulation wird versucht ein möglichst klares Bild der wirklichen Situation des wissenschaftlichen Arbeitens zu erhalten. Diese umfassen beispielsweise Umfragen mit Studierenden, Interviews mit Experten, und Observationen der Veranstaltungen. Auch wenn hier teilweise Erkenntnisse aus den anderen Erhebungen erwähnt werden, fokussiert dieser Beitrag auf die Ergebnisse einer Vor- und Nachstudie mit Teilnehmenden der Proseminare im WS 17/18. Diese wurden in Form von Fragebögen mit quantitativen und qualitativen Teilen und entsprechenden geschlossenen und offenen Fragen umgesetzt.

Der Pre-Test soll in erster Linie zur Profilbildung von Teilnehmenden beitragen. Entsprechend wurden diese u. a. zu ihrer Studiensituation, ihren Vorerfahrungen bzgl. wissenschaftlichen Arbeitens, ihren Sprachfertigkeiten, ihrer Einstellung zum Proseminar und zum Studium allgemein, ihrem thematischen Interesse, der Relevanz des Proseminars für das weitere Studium und zukünftiges, wissenschaftliches Arbeiten, ihre Fach- und Prozesswissen, ihrer allgemeinen Selbstwirksamkeitserwartung (nach Schwarzer und Jerusalem [SJ99]), ihrer Informationskompetenz (Selbsteinschätzung mit den Niveaus des Referenzrahmens Informationskompetenz [De16]), sowie den Erwartungen bzgl. wissenschaftlichen Arbeitens, dem Prozess, Strategien, und Werkzeugen befragt. Der Post-Test soll aufzeigen, wie der Prozess des wissenschaftlichen Arbeitens tatsächlich stattfindet und entsprechende Probleme ermitteln. Daher greift er die Aspekte des Pre-Tests auf und vertieft viele Details.

Für die quantitativen Fragen wurden größtenteils symmetrische, voll beschriftete, sechsstufige Ratingskalen verwendet. Diese orientieren sich stark an Likert-Typen, und wurden für die Datenanalyse als intervallskaliert angenommen. Die quantitative Analyse der offenen Fragen geschah über Inhaltsanalyse mit offener, aus den Antworten abgeleiteter Kodierung durch zwei unabhängige Experten zur Steigerung der Reliabilität. Die entwickelte Kodierung ist größtenteils hierarchischer Struktur.

# 4 Ergebnisse und Diskussion

Insgesamt nahmen 208 Studierende an der Erhebung teil, davon $N_{pre}$ = 166 an dem Vor- und $N_{post}$ = 117 an dem Nachtest. Die Teilnehmenden verteilten sich auf 9 der 12 angebotenen Proseminare im Wintersemester 2017/2018. Die dem Folgenden zu Grunde liegenden quantitativen und qualitativen Daten sind online in Gänze einsehbar.[6]

## 4.1 Profil der Studierenden

Die Teilnehmenden der Proseminare haben am Ende ihres 2. Semesters/am Anfang des 3. Semesters kaum Vorerfahrungen im wissenschaftlichen Arbeiten. Für ihr Studium ist ihnen wichtig etwas Neues zu lernen und sich mit interessanten Themen zu beschäftigen. Die Studierenden können sowohl das Proseminar und somit das Oberthema als auch das individuelle Thema selbst wählen. Entsprechend hoch ist ihr thematisches Interesse. Außerdem liegt ihnen besonders viel daran das Proseminar mit einer guten Note abzuschließen. Insg. kann also von einer hohen Motivation der Studierenden ausgegangen werden.

Die allgemeine Selbstwirksamkeitserwartung der Teilnehmenden liegt genau im erwarteten Mittel, obgleich, gemäß den Aufgaben und Herausforderungen im akademischen Bereich, eventuell ein höherer Wert erwartet werden könnte. Ein verhaltenes Auftreten wird u. a. durch fehlende Courage und Initiative sichtbar, z. B. beim Ausprobieren von neuen Lösungswegen oder Werkzeugen, oder bei der Inanspruchnahme von zusätzlichen Hilfeangeboten. Stattdessen greifen Sie auf bewährte Verfahren aus dem Alltag zurück, die sich aber nicht einfach auf die komplexen, akademischen Probleme übertragen lassen.

Mit einer Informationskompetenzeinschätzung von B2 (Sekundarstufe II) liegen die Studierenden nach Definition der Kompetenzniveaus [De16] genau im Soll. Dabei sind die Bereiche Prüfen und Wissen am stärksten ausgeprägt, Suchen und Weitergeben am schwächsten. Das spiegeln auch die von Teilnehmenden selber erkannten Probleme wieder, bei denen Literaturrecherche, u. a. geeignete Literatur zu finden und relevante Informationen zu extrahieren, meistgenannt war. Aus weiteren, bislang unveröffentlichten Erhebungen mit Dozierenden geht allerdings auch hervor, dass das Selbstbild der Studierenden

6 christophgreven.de/hdi2018, verfügbar unter URL ab September 2018

mit B2 und das Fremdbild dieser Kompetenzen mit A2 (Sekundarstufe I) nach Experteneinschätzung sehr stark differenzieren. Möglicherweise deutet dies auf eine Überschätzung der eigenen Fähigkeiten hin, was mit den qualitativen Erkenntnissen zu Prozessverständnis, Strategie, etc. übereinstimmt, und auch zuvor schon in anderen Studien beobachtet wurde [PBR12]. Experten ordnen Studierende bzgl. der Kriterien sachliche und formale Richtigkeit, sowie bei kognitiver Konflikt am schwächsten ein. Hier zeigt sich auch die größte Diskrepanz mit dem Selbstbild der Studierenden. Experten sehen ferner die Informationskompetenz bei Wissensbedarf formulieren und thematische Relevanz am stärksten ausgeprägt.

Ihr Fach- und Prozesswissen stufen die Teilnehmenden als mittelmäßig ein. Während sich das Fachwissen sehr stark verbessert, bewirkt das Proseminar nach Einschätzung geringe Änderungen bzgl. des Prozesswissens, obgleich eine solche Verbesserung natürlich Ziel der Proseminare sein muss.

## 4.2 Anwendung des wissenschaftlichen Arbeitens

Durch die qualitativen Analysen der Prozessvorstellungen der einzelnen Teilnehmenden lassen sich folgende Prozessbereiche identifizieren: Prozessorganisation, Suchen/Finden, Aufnehmen, Evaluieren, Verwerten, Austauschen, Kommunizieren. Dabei wird klar, dass die Vorstellungen der Einzelnen unzureichend sind. Sie lassen entweder viele wichtige Teile aus, z. B. Prozessorganisation oder Austausch, oder haben nur eine sehr oberflächliche Vorstellung der einzelnen Arbeitsschritte, z. B. aufgrund fehlender Fokussierung bei der Suche. Der ständige Informationsgewinn während des wissenschaftlichen Arbeitens erfordert darüber hinaus eine kontinuierliche Neubewertung der Informationen und des Vorgehens. Bspw. können mit neu gefundenen Suchbegriffen neue, spezifischere Suchanfragen gestellt werden, sodass Teile des Arbeitsprozesses wiederholt werden müssen. Dieser iterative Aspekt wird von den Teilnehmenden allerdings vernachlässigt.

Wirkliche Strategien verfolgen Studierende bei der Suche nach Literatur nicht; nur wenige nennen beispielsweise die Rückwärtsreferenzsuche als Möglichkeit weitere Literatur zu finden. Stattdessen vertrauen sie auf bekannte Verfahren, z. B. einfache Google-Suchen. Auch suchen sie Hilfe bei Personen in ihrem sozialen Umfeld. Dabei fragen Sie z. B. Freunde und Familie um Rat, die im spezifischen akademischen Kontext für gewöhnlich aber nicht immer über das benötigte Wissen verfügen. Weiter gaben Teilnehmende auch die Bibliothek an. Während jeder 2. für das Proseminar eine Bibliothek auf-

sucht, leiht nur ca. jeder 4. Literatur aus. Bemerkenswerterweise scheinen die Studierenden damit primär Bücher in Verbindung zu bringen. Gegensätzlich dazu ist die Auswertung der Suchobjekte, wobei eine sehr klare Konzentration auf Paper erkennbar ist. Diese werden vermutlich vermehrt aus dem Internet bezogen. Sie ermöglichen zwar Zugriff auf sehr aktuelle und detaillierte Informationen zum Stand der Technik, sind aber z. B. für die Gewinnung eines Überblicks und initialen Verständnisses einer Forschungsdomäne für Anfänger eher ungeeignet. Bei der Auswahl geeigneter Quellen fehlt Teilnehmenden darüber hinaus die benötigte Erfahrung, was eine wesentliche Schwierigkeit darstellt. Die Anzahl an genutzter Quellen variiert dabei enorm. Generell könnten Studierende durch das Anlesen von mehr Werken, z. B. beschränkt auf Titel und Abstract, einen besseren Eindruck und ein Gefühl für die Spezifika und das Glossar einer Domäne bekommen. Darüber hinaus ist auch die Extraktion relevanter Informationen problematisch, sodass mögliche gute Quellen nicht adäquat verwertet werden können. Ferner stellen Zeitmanagement und Kommunikation Studierende vor neue Herausforderungen. Besonders hier lässt sich ein Grund für die oft hohen Dropout-Zahlen vermuten, z. B. wegen verpasster Deadlines.

## 4.3 Das Angebot der Proseminare

Die Erwartungen von Studierenden an Proseminare sind durchaus realistisch. Erfüllt werden diese insg. aber nicht. Widererwartend fokussieren die Proseminare hauptsächlich auf die schriftliche Ausarbeitung als Produkt, und nicht auf den wissenschaftlichen Arbeitsprozess an sich. Dabei werden vor allem die Form der Ausarbeitung, das Erstellen eines Vortrages sowie dessen Präsentation thematisiert. Zwar gaben Studierende auch Selbstorganisation und Selbstständigkeit als Proseminarinhalte an, weitere Observationen der Proseminarveranstaltungen zeigen aber, dass Studierende zwangsläufig mit diesen Themen konfrontiert sehen, auf diese aber nicht aktiv durch die Veranstalter eingegangen wurde. Insg. sind diese Angaben daher kritisch zu beurteilen und liegen vermutlich eigentlich noch unter den genannten. Das lassen auch die angegebenen Lernergebnisse vermuten, denn hier nannten Teilnehmende nur wenige Dinge, die ihnen beigebracht wurden; der Großteil geschah im Selbststudium. Am wenigsten Beachtung fanden grundlegende Themen wie die Rolle/Aufgaben von Forschenden, Mehrwerte schaffen, oder eigene Ideen entwickeln.

Insgesamt stellt das Proseminar eine hochrelevante Lehreinheit dar, die auf weitere Seminare und Arbeiten vorbereiten soll. Es sollte zum eigenständigen wissenschaftlichen Arbeiten ermächtigen, sodass Absolventen sich zukünftig im Wesentlichen auf Inhalte konzentrieren können. Allerdings ordnen die Studierenden die Nützlichkeit der Veranstaltung, das heißt wie sehr sie auf zukünftiges eigenständiges wissenschaftliches Arbeiten vorbereitet, nur als „eher gut“ ein. Unter Betrachtung der angegebenen Lernerfolge einzelner Studierender wird ersichtlich, dass in der Lehrveranstaltung neben fachlichen Inhalten nur wenig zum Arbeitsprozess vermittelt wurde, sondern klar auf Formalien wissenschaftlicher Ausarbeitungen fokussieren. Entsprechend niedrig stufen die Teilnehmenden auch die Relevanz des Proseminars ein, die bzgl. zukünftigem wissenschaftlichen Arbeiten im Verlauf sogar noch abnimmt. Anscheinend mangelt es an generell an der Motivation der Veranstaltung.

## 4.4 Zusammenfassung

Zusammenfassend lässt sich erkennen, dass Studierende realistische Erwartungen an wissenschaftliches Arbeiten in Proseminaren haben, dabei aber tendenziell ihre eigenen Fähigkeiten überschätzen, und versuchen Verhaltensmuster des Alltags auf die komplexen akademischen Probleme zu übertragen. Dabei kennen sie den Prozess des wissenschaftlichen Arbeitens nicht genau und vermissen wesentliches Detailwissen zu konkreten Schritten. Im Prozess zeigten sich kaum Suchstrategien und von verfügbaren Werkzeugen und Hilfen wird kein Gebrauch gemacht. Darüber hinaus wissen Studierende vor allem nicht, wonach sie suchen müssen und wofür. Neben dem kritischen Umgang mit Informationen, bei denen ein gewisses Maß an Erfahrung oder Übersicht für eine fundierte Entscheidung fehlt, haben sie auch wesentliche Probleme mit dem Zeitmanagement. Der Austausch untereinander aber auch mit den Betreuern fällt schwer. So werden freiwillige Angebote für Feedback kaum angenommen und Kontaktstellen mit Betreuern beschränken sich auf wenige Pflichttermine. Insgesamt fokussieren die Proseminare nicht ausreichend auf den wissenschaftlichen Arbeitsprozess, sodass der Lernerfolg bei Studierenden gering ist. Dabei erkennen Teilnehmende überhaupt nicht die Wichtigkeit von Proseminaren für zukünftiges wissenschaftliches Arbeiten.

## 5 Zusammenfassung und Ausblick

Diese Arbeit hat die akademische Grundlehre des wissenschaftlichen Arbeitens und dessen Probleme im Rahmen der Hochschullehre analysiert. Dies geschah anhand des Beispiels von Proseminaren, die eine „Einführung in das wissenschaftliche Arbeiten“ ermöglichen sollen. Wie dieser Ausschnitt an Ergebnissen aus mehreren Erhebungen zeigt, gibt es dabei erhebliche Probleme, sodass die eigentliche Intention der Veranstaltung nicht bedient wird. Es ist erkennbar, dass Studierende zukünftig mehr Unterstützung im Arbeitsprozess des wissenschaftlichen Arbeitens brauchen. Dabei bedarf es ihres Verständnisses konkreter Schritte und deren Absichten, um selber sinnvolle Strategien entwickeln zu können. Es existieren schon diverse Hilfen, z. B. in Form von Softwaretools, welche die Studierenden nutzen könnten. Diese im Arbeitsprozess situativ zugänglich zu machen, ohne dabei eine zusätzliche Last zu sein, ist eine große Herausforderung. Auch zeigten die Erhebungen, dass Kommunikation und Austausch verbessert werden können, sowohl zwischen Lernenden und Lehrenden, als auch bei Lernenden untereinander. Da die Teilnehmenden in Proseminaren das erste Mal über einen längeren Zeitraum frei arbeiten müssen, sollten sie außerdem bei dem konkreten Zeitmanagement innerhalb der Veranstaltung unterstützt werden. Insg. müssen die Studierenden zu einem selbstreflektierten Arbeitsprozess ermächtigt werden, bei dem sie Informationen und Arbeitsweise immer wieder in Frage stellen.

## Literaturverzeichnis


[De09] Deutscher Bibliotheksverband: Standards der Informationskompetenz für Studierende, 2009.

[De16] Deutscher Bibliotheksverband: Referenzrahmen Informationskompetenz, 2016.

[Ha16] Hapke, T.: Informationskompetenz anders denken – zum epistemologischen Kern von „information literacy“. Handbuch Informationskompetenz, S. 9–21, 2016.

[In16] Informationskompetenz – Bundesstatistik 2016, 20. 04. 2018, http://zpidlx54.zpid.de/wp-content/uploads/2018/03/IK_Bundesstatistik_2016.pdf

[In17] Informationskompetenz Veranstaltungsstatistik – Auswertung 2017, 20. 04. 2018, http://www.informationskompetenz.de/eventsdb/index.php?s=res&y=2017

[PBR12] Purcell, K.; Brenner, J.; Rainie, L.: Search Engine Use 2012, 2012.

[SJ99] Schwarzer, R.; Jerusalem, M. (Hrsg.): Skalen zur Erfassung von Lehrer- und Schülermerkmalen. Dokumentation der psychometrischen Verfahren im Rahmen der Wissenschaftlichen Begleitung des Modellversuchs Selbstwirksame Schulen, 1999, S 13.

[Tr16] Tremp, P.: Informationskompetenz und forschungsorientiertes Studium – ein Beitrag aus der Hochschuldidaktik. Handbuch Informationskompetenz, S. 219–226, 2016.

**Bisher sind in dieser Reihe erschienen:**

1 Schwill, A. (Hrsg.): Hochschuldidaktik der Informatik. HDI 2008 – 3. Workshop des GI-Fachbereichs Ausbildung und Beruf/Didaktik der Informatik 2008
2009 | ISBN 978-3-940793-75-1

2 Stechert, P.: Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht
2009 | ISBN 978-3-86956-024-3

3 Freischlad, S.: Entwicklung und Erprobung des Didaktischen Systems Internetworking im Informatikunterricht
2010 | ISBN 978-3-86956-058-8

4 Engbring, D.; Keil, R.; Magenheim, J.; Selke, H. (Hrsg.): HDI 2010 – Tagungsband der 4. Fachtagung zur „Hochschuldidaktik Informatik“
2010 | ISBN 978-3-86956-100-4

5 Forbrig, P.; Rick, D.; Schmolitzky, A. (Hrsg.): HDI 2012 – Informatik für eine nachhaltige Zukunft
2013 | ISBN 978-3-86956-220-9

6 Diethelm, I.; Arndt, J.; Dünnebier, M.; Syrbe, J. (Eds.): Informatics in Schools – Local Proceedings of the 6th International Conference ISSEP 2013 – Selected Papers
2013 | ISBN 978-3-86956-222-3

7 Brinda, T.; Reynolds, N.; Romeike, R.; Schwill, A. (Eds.): KEYCIT 2014. Key Competencies in Informatics and ICT
2015 | ISBN 978-3-86956-292-6

8 Dörge, C.: Informatische Schlüsselkompetenzen. Konzepte der Informationstechnologie im Sinne einer informatischen Allgemeinbildung
2015 | ISBN 978-3-86956-262-9

9 Forbrig, P.; Magenheim, J. (Hrsg.): HDI 2014 – Gestalten von Übergängen. 6. Fachtagung Hochschuldidaktik der Informatik. 15.–16. September 2014, Universität Freiburg
2015 | ISBN 978-3-86956-313-8

10 Schwill, A.; Lucke, U. (Hrsg.): Hochschuldidaktik der Informatik. HDI 2016 – 7. Fachtagung des GI-Fachbereichs Informatik und Ausbildung/Didaktik der Informatik. 13.–14. September 2016 an der Universität Potsdam
2016 | ISBN 978-3-86956-376-3

11 Müller, D.: Der Berufswahlprozess von Informatiklehrkräften 2017
ISBN 978-3-86956-392-3

12 Bergner, N.; Röpke, R.; Schroeder, U.; Krömker, D. (Hrsg.): Hochschuldidaktik der Informatik. HDI 2018 – 8. Fachtagung des GI-Fachbereichs Informatik und Ausbildung/Didaktik der Informatik. 12.–13. September 2018 an der Goethe-Universität Frankfurt
2018 | ISBN 978-3-86956-435-7

In dieser Reihe erscheinen Tagungsbände und ausgewählte Forschungsberichte zu Themen aus der Didaktik der Informatik in Schule und Hochschule.

---

Die 8. Fachtagung für Hochschuldidaktik der Informatik (HDI) fand im September 2018 zusammen mit der Deutschen E-Learning Fachtagung Informatik (DeLFI) unter dem gemeinsamen Motto „**Digitalisierungs**wahn**sinn**? – Wege der Bildungstransformationen" in Frankfurt statt.

Dabei widmet sich die HDI allen Fragen der informatischen Bildung im Hochschulbereich. Schwerpunkte bildeten in diesem Jahr u. a.:

- Analyse der Inhalte und anzustrebenden Kompetenzen in Informatikveranstaltungen
- Programmieren lernen & Einstieg in Softwareentwicklung
- Spezialthemen: Data Science, Theoretische Informatik und Wissenschaftliches Arbeiten

Die Fachtagung widmet sich ausgewählten Fragestellungen dieser Themenkomplexe, die durch Vorträge ausgewiesener Experten und durch eingereichte Beiträge intensiv behandelt werden.

ISSN 1868-0844
ISBN 978-3-86956-435-7