Beckmann, Nils; Korte, Thomas

Korrelationen messbarer Einflussfaktoren beim Erlernen und Anwenden einer Programmiersprache

Schmohl, Tobias [Hrsg.]: Situiertes Lernen im Studium. Didaktische Konzepte und Fallbeispiele einer erfahrungsbasierten Hochschullehre. Bielefeld : wbv media 2021, S. 213-225. - (TeachingXchange; 5)

Korrelationen messbarer Einflussfaktoren beim Erlernen und Anwenden einer Programmiersprache

Nils Beckmann, Thomas Korte

Schlagworte: Digitalisierung, Programmierung, Einflussfaktoren, Korrelationen

1 Einleitung

Im heutigen Berufsumfeld kommt den digitalen Grundfähigkeiten eine größere Bedeutung zu als früher und diese wird in Zukunft vermutlich weiter steigen (Stifterverband, 2018). Mit zunehmender Digitalisierung steigt auch die Anzahl der technologischen Berufe, in denen über diese digitalen Grundfähigkeiten hinaus zusätzlich technologische Fähigkeiten gefordert sind, wie insbesondere die Programmierung mit einer Programmiersprache (Loshkareva et al., 2015). Um diesem wachsenden Bedarf an sogenannten „Future Skills" (Stifterverband, 2018) gerecht zu werden, bieten mehr und mehr Hochschulen in Deutschland informationstechnische Studiengänge an wie Informatik sowie Elektrotechnik und daraus hervorgegangen Technische Informatik (ITG & GI, 2018). Das Ziel eines entsprechenden Studiums ist allgemein die Bildung von verantwortungsvollen Persönlichkeiten (Technische Hochschule Ostwestfalen-Lippe, 2018), die die technologischen Fähigkeiten ihres Gebiets fachlich und methodisch beherrschen. Für Lehrende stellt sich in diesem Zusammenhang die Frage, durch welche Faktoren dieses Ziel erreicht werden kann. Zu dieser Fragestellung wurden bereits diverse Faktoren untersucht wie beispielsweise Zeiteinsatz der Studierenden (Schulmeister & Metzger, 2011), aktivierende Methoden und Kommunikationsverhalten der Lehrenden (Schulz, 2010), Organisation der Lehre (Metzger et al., 2012), Kundenorientiertes Denken (Stoyan & Glinz, 2005), Gamification (Detering et al., 2001), Programmiermetriken (Rupp et al., 2017) sowie viele weitere (Hattie, 2015), jeweils mit unterschiedlichen Ergebnissen. Einige dieser Faktoren können von den Studierenden selbstständig und selbstorganisiert beeinflusst werden (Sembill, 2006), andere wiederum nicht oder nur durch die Lehrenden.

In diesem Beitrag wird eine Analyse nach dem studierendenzentrierten Ansatz (Wright, 2011) vorgestellt, bei der bestimmte, ausgewählte Einflussfaktoren auf das Erlernen und Anwenden einer Programmiersprache gemessen und quantifiziert werden. Die Quantifizierung der Einflussfaktoren erfolgt über einen Fragebogen durch die persönliche und subjektive Beurteilung der Studierenden. In der Analyse wird dann geprüft, ob zwischen den Einflussfaktoren Korrelationen bestehen. Hierbei

konnten Einflussfaktoren identifiziert werden, die sowohl miteinander als auch in Bezug zur obigen Zielsetzung positiv korrelieren. Zu diesen gehören die für den Fragebogen gewählten Begriffe „Spaß“, „Zeiteinsatz“, „Verständnis“, „Erfolgserlebnisse“ sowie die Klausurprüfungsbenotung. Basierend auf diesen Erkenntnissen werden Ansatzpunkte für Szenarien entwickelt, die Lehrende zur konstruktiven Beeinflussung dieser Einflussfaktoren in der Lehre einsetzen können. Abschließend werden die Ergebnisse im Kontext zu bisherigen Literaturergebnissen diskutiert.

2 Rahmenbedingungen

Im Modul „Programmiersprachen 1“ des Fachbereichs „Elektrotechnik und Technische Informatik“ der Technischen Hochschule Ostwestfalen-Lippe wird den Studierenden des ersten Fachsemesters eine Vorlesung und ein Praktikum angeboten mit je 2 Stunden pro Woche. Diesen zeitlichen Umfang und Einsatz sollen die Studierenden, gemäß Modulhandbuch (Technische Hochschule Ostwestfalen-Lippe, 2019), um 6 Stunden Lernleistung pro Woche außerhalb der Unterrichtseinheiten eigenständig erweitern, was insgesamt einer Arbeitsleistung („Workload“) von 150 Stunden für 5 Credits entspricht bei einer Dauer der Anwesenheits-/Vorlesungszeit von knapp 4 Monaten. Selbstverständlich ermöglicht diese Arbeitsleistung nur einen kleinen Einblick in eine Thematik: Man sagt, dass man nach etwa 5–10 Jahren bzw. 5.000 bis 10.000 Stunden der Übung, Auseinandersetzung und des Trainings eine Tätigkeit wirklich gut beherrscht (Norman, 1978) – oder wie der Volksmund es sagt: „Übung macht den Meister.“ Das gewählte Lernsetting mit deutlichem Praxisanteil ermöglicht den Studierenden, ihr theoretisch erworbenes Wissen (aus der Vorlesung) auch praktisch (im Praktikum eigenständig mit der Möglichkeit zum Fragenstellen) anzuwenden, was sowohl die Fach- wie auch die Methodenkompetenz verbessert (Technische Hochschule Ostwestfalen-Lippe, 2019). „Theorie ohne Praxis ist leer, Praxis ohne Theorie ist blind“ ist ein dazu passendes Zitat des Philosophen Immanuel Kant.

Inhaltlich geht es im Modul um die Grundlagen der „Prozeduralen Programmierung“ in der Programmiersprache C. Diese Sprache wurde bereits 1972 entwickelt, ist aber auch heute noch die zweitpopulärste Programmiersprache der Welt nach Java (TIOBE, 2019). Diverse andere „Hochsprachen“ wie C++, C#, Java, Rust, Python und so weiter sind aus ihr hervorgegangen und verwenden ähnliche Syntax und Regeln (Zuse, 1999). Auch in der heimischen Wirtschaft und Deutschland gesamt ist C nach wie vor regional und überregional stark vertreten aufgrund der Möglichkeiten zur hardwarenahen und ressourceneffizienten Programmierung.

Das Lernsetting ist angelehnt an das Constructive Alignment (Biggs, 1996; Uni Konstanz, 2014). Lernziele der Lehrveranstaltung sind die Beherrschung der Grundelemente der prozeduralen Programmierung in C und die Kompetenz, kleinere Quelltexte, Algorithmen und Programme in dieser Sprache problemorientiert entwickeln zu können. Zudem sollen Syntax, Semantik und Schlüsselwörter der Sprache bekannt sein und die zur Programmentwicklung notwendigen Werkzeuge wie Entwicklungs-

umgebung, Compiler und Debugger angewendet werden können. Die zu erreichenden Lernzielniveaustufen sind das Erinnern, Verstehen und Anwenden (Bloom et al., 1973). Im Praktikum können und sollen die Studierenden ihre aus der Vorlesung vermittelten Inhalte anwenden und vertiefen, indem sie angeleitet Übungszettel mit Übungsaufgaben bearbeiten, bei denen Quelltexte programmiert, überprüft, verstanden und verbessert werden sollen. Jede Woche wird ein neues Thema in der Vorlesung erklärt (z. B. Ausgabe und Eingabe, Berechnungen, Schleifen, Strukturen, Bedingungen, Funktionen, Bibliotheken), wozu ein thematisch passender Übungszettel ausgegeben wird. Die Erreichung der Lernziele wird nach Ende der Vorlesungszeit durch eine Klausur geprüft, bei der die entsprechend gelernten Inhalte aus Vorlesung und Übungsaufgaben abgefragt werden.

<table>
<tr><th colspan="3">Modul „Programmiersprachen 1“</th></tr>
<tr><th colspan="2">Rahmenbedingungen</th><th>Inhalte</th><th>Materialien</th></tr>
<tr><td>Vorlesung
2 SWS</td><td>Praktikum
2 SWS</td><td>Quelltexte kompilieren</td><td rowspan="2">Übungsaufgaben
(jede Woche neue ÜA)</td></tr>
<tr><td colspan="2" rowspan="2">Präsenzzeit 60 h
Eigenstudium 90 h</td><td>Eingabe und Ausgabe</td></tr>
<tr><td>Syntax und Semantik</td><td>Beispiel-/Musterlösungen</td></tr>
<tr><td colspan="2" rowspan="2">Arbeitsleistung
10 h/Woche</td><td>Datentypen und Variablen</td><td>Quelltextbeispiele</td></tr>
<tr><td>Operatoren und Ausdrücke</td><td>Skript</td></tr>
<tr><td colspan="2" rowspan="2">Pflichtveranstaltung
1. Fachsemester</td><td>Kontrollstrukturen</td><td>Online-Lernplattform</td></tr>
<tr><td>Bibliotheksfunktionen</td><td rowspan="2">Links und Hinweise
Fachbücher
Sprachreferenzen</td></tr>
<tr><td colspan="2" rowspan="2">Pro Jahr
>200 Studierende
aus 4 Studiengängen</td><td>Arrays</td></tr>
<tr><td>Zeiger</td><td>Selbsteinschätzungstests</td></tr>
<tr><td colspan="2" rowspan="2">Begleitung durch
3–4 Dozierende</td><td>Funktionen und Parameter</td><td rowspan="2">Softwareanleitungen</td></tr>
<tr><td>Zahlensysteme</td></tr>
<tr><td colspan="2" rowspan="2"></td><td>Speicherverwaltung</td><td rowspan="2">Prüfungsfragen/Altklausuren</td></tr>
<tr><td>Rekursion</td></tr>
<tr><td colspan="2"></td><td>Anwendungsbeispiele</td><td>Forum</td></tr>
</table>

Abbildung 1: Überblick Modul „Programmiersprachen 1“ (SWS steht für „Semesterwochenstunden“, also die Anzahl der Lehrstunden pro Woche)

3 Analysemethode

Abgleitet von o. g. Lernsetting und Lernzielen wurde ein Fragebogen entwickelt, um von den Kursmitgliedern die subjektiven Einschätzungen ihrer Arbeitsweisen, Einstellungen, Erfahrungen und Erfolge in Bezug auf das Modul abzufragen. Dieser ist so gestaltet, dass mathematische Korrelationen zwischen den einzelnen Punkten hergestellt werden können. Konkret abgefragt wurden unter anderem der Zeiteinsatz, differenziert nach den verschiedenen theoretischen und praktischen Anteilen, Jahre und Art der Vorerfahrung sowie subjektive Einschätzungen zur Art des eigenen Lernprozesses und zum Stand der eigenen erlangten Fähigkeiten in Bezug auf die prozedurale Programmierung. Letzteres wurde zuvor bereits durch eine Klausurprüfung bewertet, deren Note die Fragebogenteilnehmenden auch angeben konnten. Mit diesen abgefragten Informationen werden im Folgenden daraus ermittelte aussichtsreiche Einflussfaktoren beschrieben, um für die in der Einleitung genannten Ziele Ansätze zur konstruktiven Beeinflussung zu finden. Die Fragebogen wurden am Ende des Wintersemesters 2017/18 verteilt und die Studierenden kamen dabei aus den Bachelorstudiengängen „Elektrotechnik", „Technische Informatik" und dem gerade neu begonnenen Studiengang „Medizin- und Gesundheitstechnologie"[1]. Der Rücklauf betrug 77 Fragebogen. Deren Auswertung und Analyse wird nachfolgend vorgestellt.

4 Ergebnisse der Analyse

4.1 Zeiteinsatz und Vorerfahrung

Der in den Antworten angegebene Gesamtzeiteinsatz für das Modul beläuft sich auf durchschnittlich 58 Stunden (also grob 4 h pro Woche), in Einzelfällen maximal etwa doppelt so viel, und verteilte sich mit etwa 60 zu 40 Prozent auf Theorie (Skript und Fachbücher lesen, Vorlesung besuchen usw.) und Praxis (Programmieren, Praktikum besuchen, Übungsaufgaben lösen usw.). In den Zeitangaben enthalten sind sowohl Anwesenheitszeiten in den Veranstaltungen (Vorlesung und Praktikum) als auch das Selbststudium (Vorlesungsfolien, Übungsaufgaben, Skript, Fachbücher, Lerngruppen, Programmbeispiele, Klausurvorbereitung, Programmieren, Internetvideos usw.). Laut diesen Angaben erreicht damit keiner der Studierenden die oben angegebene Arbeitsleistung von 150 Stunden für das Modul. Zusätzlich geben die Studierenden eine durchschnittliche Vorerfahrung in der Programmierung von etwa 1 ½ Jahren an, in Einzelfällen auch über 5 Jahre, wobei knapp die Hälfte keinerlei Programmiervorerfahrung aufweist. Demnach herrscht zu Beginn der Lehrveranstaltung eine hohe Heterogenität im Wissensstand.

1 Ein neuer Studiengang „Data Science" startete erst im Wintersemester 2018/19.

4.2 Spaß und Kenntnisgewinne

Ein Aspekt, der auch abgefragt wurde, war der Spaß an der Programmierung: Es wird deutlich, dass über die Hälfte der Studierenden (56 %) viel oder sehr viel Spaß an der Programmierung hat (bei denen mit Vorerfahrung sogar 73 %), knapp ein Drittel (32 %) nur wenig oder sehr wenig. Etwa 61 % der Teilnehmenden gaben an, dass sie nach der Veranstaltung die Grundlagen der Programmierung gut oder sehr gut beherrschen würden und sich in der Lage sähen, kleinere Programmieraufgaben spontan lösen zu können (Beispiel: Programmierung einer Funktion, die prüft, ob eine Zahl eine Primzahl ist). Dieser Anteil war bei den Programmieranfängern kleiner (24 %) als bei den Studierenden mit Vorerfahrung (89 %) und insbesondere auch größer bei denen, die Spaß an der Programmierung hatten (83 %) oder dabei Erfolgserlebnisse hatten (89 %).

4.3 Klausurergebnis

In den Fragebogen haben 70 % ihr Klausurergebnis angegeben und 86 % von diesen hatten die Klausur auch bestanden. Durchschnittlich erreichten die Programmieranfänger die Klausurnote 3,5, die Studierenden mit Vorerfahrung die Note 2,1. Programmieranfänger, die nur selten Vorlesung oder Praktikum besucht und sonst nur wenig zusätzliche Zeit in das Fach investiert hatten, bestanden die Klausur in der Regel nicht. Wie aus den zuvor erwähnten hohen positiven Korrelationen ersichtlich ist, ist die Klausurnote umso besser, je mehr Zeit investiert wurde. Eine Verbesserung der Klausurnote (hier mit dem Symbol Δ bezeichnet) um durchschnittlich je $\Delta = 0{,}3$ erreichen Programmieranfänger durch einen Gesamtzeiteinsatz von je 1 h/Woche und Studierende mit Vorerfahrung mit je 2 h/Woche. Ein Jahr der Vorerfahrung geht einher mit einer Verbesserung um etwa $\Delta = 0{,}4$ Klausurnoten. Weiterhin kann hier statistisch differenziert werden nach Zeiteinsatz für Theorie und Praxis: Der praktische Zeiteinsatz zahlt sich in Bezug auf das Klausurergebnis wie 2 zu 1 gegenüber dem theoretischen Zeiteinsatz aus. Außerdem finden sich statistisch auch Anteile, die nur durch theoretischen und praktischen Zeiteinsatz gemeinsam beschrieben werden können.

Tabelle 1: Ergebnisse der statistischen Auswertung der Fragebogen

Eingegangene Fragebogen	77
Durchschnittlicher Gesamtzeiteinsatz [in Stunden]	58 ± 65 (≈ 4 pro Woche)
Maximaler Gesamtzeiteinsatz [in Stunden]	129
Gesamtzeiteinsatz, Theorie zu Praxis	Etwa 60 zu 40
Durchschnittliche Vorerfahrung [in Jahren]	1,3 ± 2,0
Anteil der Studierenden mit Vorerfahrung	57%
Habe Spaß an der Programmierung	56%
Habe Grundlagen der Programmierung verstanden	61%

(Fortsetzung Tabelle 1)

Verbesserung der Klausurnote durch Zeiteinsatz	Δ = 0,3 pro 1h/Woche
Verbesserung der Klausurnote durch Vorerfahrung	Δ = 0,4 pro 1 Jahr
Verbesserung der Klausurnote durch Zeiteinsatz für praktisches Lernen gegenüber theoretischem Lernen	Etwa 2 zu 1
Angegeben sind Mittelwerte mit Standardabweichung (±).	
Das Symbol Δ bezeichnet eine Verbesserung der Klausurnote.	

4.4 Korrelationen

Für die genannten Faktoren kann paarweise der lineare, diskrete Korrelationskoeffizient C ϵ [–1; +1] empirisch berechnet werden, der angibt, inwieweit ein Zusammenhang zwischen den Verläufen von zwei Bezugsgrößen besteht. In Bezug auf das Klausurergebnis gibt es hinsichtlich der Faktoren Zeiteinsatz (C = +0,53), Verständnis und Fähigkeiten (+0,68), Erfolgserlebnisse (+0,57) und Spaß (+0,73) eine hohe positive Korrelation (C ≥ +0,5). Wer mehr Spaß an einer Tätigkeit hat, investiert mehr Zeit dafür (+0,50), verbessert dadurch wiederum seine Fähigkeiten und seine Kenntnisse darin (+0,52), wodurch auch mehr Erfolgserlebnisse dabei auftreten können (+0,76), und dies erhöht wiederum den Spaß an der Tätigkeit (+0,68), wodurch sich der Kreis schließt. Genau diese paarweisen Zusammenhänge finden wir auch statistisch über hohe positive lineare Korrelationen wieder (siehe Klammern). Dadurch kann eine förderliche Positivspirale in Gang gesetzt werden (siehe Abbildung 2).

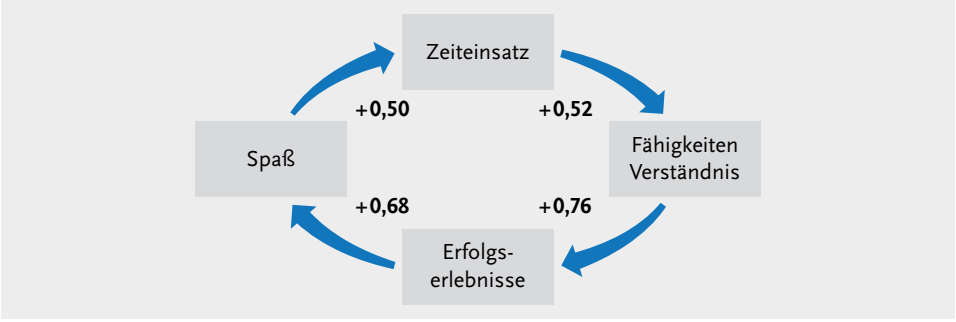

Abbildung 2: Positivspirale beim Erlernen einer neuen Fähigkeit (hier der Programmierung); die Zahlen geben den hier ermittelten paarweisen linearen Korrelationskoeffizienten an

5 Szenarien in der Lehre

Durch die Analysen sind mehrere Einflussfaktoren identifiziert worden, die sich untereinander und in Bezug zur Lernerfahrung statistisch positiv auswirken (siehe Abbildung 2) sowie stark mit dem Klausurprüfungsergebnis korrelieren. Jeder der Einflussfaktoren stellt damit für die Lehrperson einen Ansatzpunkt dar, um konstruktiv

beeinflussend durch Maßnahmen in die Lehre eingreifen zu können. Aufgrund der mathematisch ermittelten positiven Korrelation ist zu erwarten, dass sich durch positive Änderung eines Einflussfaktors die damit korrelierten anderen Einflussfaktoren auch positiv verändern. Hierbei ergeben sich vier Szenarien, je nachdem ob bei A) Zeiteinsatz, B) Fähigkeiten und Verständnis, C) Erfolgserlebnissen oder D) Spaß angesetzt wird. Zunächst werden im Folgenden einige beeinflussende Maßnahmen vorgestellt und danach zu jedem Einflussfaktor ein mögliches Szenario, das potenziell durch Erhöhung eines Einflussfaktors angeregt werden kann.

Den A) Zeiteinsatz könnten erhöhen
- prüfungsrelevante Aufgaben in den Übungsaufgaben besprechen
- Prüfungsvorleistungen in den Praktika abfragen
- Anwesenheitspflichten (Land NRW, 2019, § 64, 2a)
- Übungsaufgaben nur in den Praktika verteilen
- in den Lehrveranstaltungen den Studierenden für sie interessante Inhalte vermitteln

Die B) Fähigkeiten und Verständnis könnten erhöhen
- individuelle Betreuung
- Mentorenprogramme oder Lernwegbegleitung (Eller-Studzinsky, 2021)
- Erhöhung des Betreuungsschlüssels
- Fragen zulassen und beantworten
- Inhalte aus verschiedenen Blickwinkeln darstellen

Die C) Erfolgserlebnisse könnten erhöhen
- mit besonders einfachen Aufgaben beginnen
- schnelle Rückkopplung zu den Ergebnissen von Aufgaben geben
- Gamification-Elemente (Detering et al., 2001)
- Instant Gratification (Barnes et al., 2007)
- Preise verleihen für besonders gute Resultate
- sprachliche Anerkennung gelungener Ergebnisse
- Wissensabfrage-Wettbewerbe durch Online-Umfragetools

Den D) Spaß könnten erhöhen
- Rätsel und Zaubertricks mit Bezug zum Thema
- Witze mit Bezug zum Thema
- Witze mit Bezug zum Thema von den Studierenden abfragen und prämieren
- als Lehrperson selbst mit Begeisterung das Thema vermitteln, was die Zuhörer auch begeistert
- Neugierde der Teilnehmer wecken über Unbekanntes wie beispielsweise das Resultat eines Quelltextes
- Zugang zur Lebenswirklichkeit und zu den Lebenszielen der Teilnehmer herstellen

Beispielsweise könnten die folgenden Ansätze entsprechende Szenarien auslösen (siehe Abbildung 3).

Zu A) Werden Übungsaufgaben mit direktem Prüfungsbezug nur in den Lehrveranstaltungen ausgeteilt und besprochen, könnte dadurch der Anreiz für Studierende größer sein, an Lehrveranstaltungen teilzunehmen, was den Zeiteinsatz erhöht. Damit würden sich die Studierenden mehr mit diesen prüfungsrelevanten Studieninhalten auseinandersetzen, was das Verständnis erhöht. Mit häufigerer und intensiverer Auseinandersetzung mit den Studieninhalten treten auch häufiger Erfolgserlebnisse auf, was den Spaß der Studierenden erhöhen kann. Geeignet scheint dieses Szenario, da die Wirksamkeit der Kommunikation von und über prüfungsrelevante Inhalte für den langfristigen Lernerfolg bereits gezeigt wurde (Schulz, 2010). Eine Schwierigkeit bei diesem Szenario kann darin bestehen, dass aufgrund des kurzen zeitlichen Umfangs einer Prüfungsleistung darin nicht alle relevanten Themen der Lehrveranstaltung abgefragt werden können: Folglich bestünde die Gefahr, dass die Übungsaufgaben nicht alle relevanten Themen abdecken.

Zu B) Wenn Lernwegbegleitungen (Eller-Studzinsky et al., 2021) eingeführt werden zwischen Studierenden unterschiedlicher Semester, können Studierende höherer Semester ihre Kenntnisse vertiefen und das inhaltliche Verständnis ihres/ihrer zugeteilten Studierenden in Einzelgesprächen erhöhen. Zudem können Lernhürden, wie die Angst vor dem Fragenstellen, reduziert werden. Die/Der Lernwegbegleitende kann auch schnellere und direktere Rückmeldung geben. Auch kann sie/er durch Lob für Arbeitsergebnisse Erfolgserlebnisse schaffen. Gemeinsam können beide auf diese Weise Spaß am Lernen haben. In einer anderen Studie (Metzger et al., 2012) wurde angegeben, dass nur ein Bruchteil aller Studierenden unter „herkömmlichen Bedingungen" den Lernalltag selbstbestimmt erfolgreich gestaltet. Lernwegbegleitungen könnten bei diesem Aspekt zusätzlich Abhilfe schaffen, indem der/die begleitete Studierende von den Erfahrungen des/der erfahrenen Studierenden für den Lernalltag und die Art des Wissenserwerbs profitiert. Dieses Modell erfordert zusätzlichen Zeiteinsatz, sowohl organisatorisch wie auch für die Studierenden höherer Semester.

Zu C) Wenn die Übungsaufgaben zu Beginn der Lehrveranstaltung einfach gehalten sind, so können sie leichter von den Studierenden gelöst werden. Damit haben diese direkt zu Beginn Erfolgserlebnisse, was den Spaß bei der Tätigkeit fördert. Zusätzlich kann eine positive Rückmeldung vom Dozenten bezüglich der gut gelösten (einfachen) Übungsaufgabe dem Studierenden zusätzliche Anerkennung ermöglichen. Durch diese Faktoren könnte der/die Studierende den Zeiteinsatz erhöhen, um Erfolgserlebnisse und Spaß wiederholen zu können. Während dieses Prozesses steigt sein/ihr Verständnis an. Durch solche persönlichen Rückmeldungen der/des Dozierenden nach dem studierendenzentrierten Ansatz kann auch die Lernumgebung positiv beeinflusst werden (Wright, 2011), was zur positiven Atmosphäre und zum Spaß beim Lernen beitragen kann und andere Lernhürden reduziert. Bei diesem Szenario ist auf einen angemessenen Niveauanstieg der Übungsaufgaben zu achten, damit auf der einen Seite die Studierenden Erfolgserlebnisse sammeln können und

auf der anderen Seite alle relevanten Inhalte der Lehrveranstaltung mit angemessenem Schwierigkeitsgrad vermittelt werden können.

Zu D) Werden in den Lehrveranstaltungen fachlich relevante Rätsel oder Zaubertricks vorgestellt, so kann die Neugierde der Studierenden geweckt werden, was den Spaß an der Materie fördert. Ein selbst erfolgreich erprobtes Beispiel mit Bezug zur Informatik ist das Erraten eines von Studierenden geheim ausgesuchten Bildes: Gewählt wird dieses aus 16 möglichen Bildern. Danach erhalte ich als Dozent von den Studierenden nur 4 Ja-/Nein-Informationen, ob sich das Bild auf speziell präparierten Karten befindet. Da ich somit 4 Bit an Information erhalten habe, kann ich 16 verschiedene Möglichkeiten differenzieren und das Bild damit eindeutig bestimmen. Viele Studierende sind dadurch überrascht. Es wird eine Auseinandersetzung mit der Thematik angeregt, wodurch die Studierenden Zeit mit der Überlegung darüber verbringen, wie der Zaubertrick funktioniert. Insbesondere auch ein thematischer Bezug der Rätsel zur Alltagswirklichkeit oder mit Praxis- und Anwendungsbezug kann diesen Effekt verstärken. Damit können die Studierenden über die Rätsel ein Verständnis für die Thematik entwickeln und mit diesem Verständnis bei anderen Übungsaufgaben Erfolgserlebnisse erfahren. Nach dem Verständnis der Gamification (Detering et al., 2001) wird durch solche Rätsel eine spielerische Komponente in die Lehre eingeführt, die die Hürde für weiteren Zeiteinsatz der Studierenden senken kann. Zum einen können in der Regel nicht alle Lehrinhalte über Rätsel vermittelt werden, zum anderen kann durch zu viele Rätsel das Interesse der Studierenden einer Langeweile

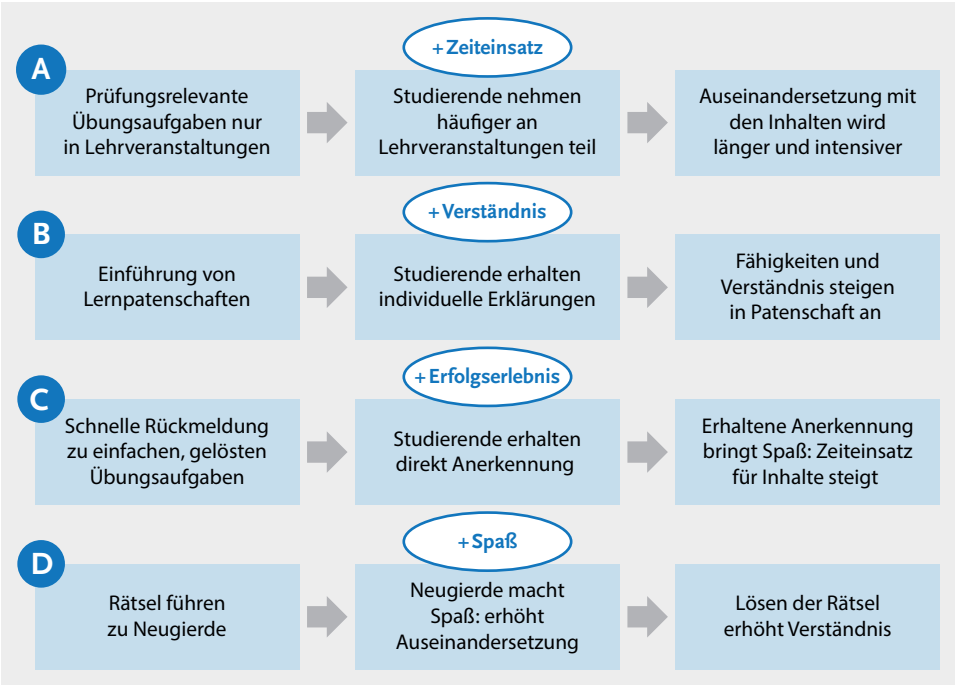

Abbildung 3: Mögliche Positivszenarien, ausgelöst durch eingreifende Maßnahmen der Lehrkraft

weichen: Hier ist ein geeignetes Maß zu finden, sodass die Rätsel gerade genug Neugierde vermitteln, damit die weiteren Lehrinhalte auch als interessant empfunden werden.

Dass alle ermittelten Korrelationen kleiner als 1,0 sind, bedeutet, dass die Positivspirale nicht für alle Teilnehmer*innen und somit auch nicht für jede Gruppe durch die gleichen Maßnahmen in Gang gesetzt werden kann. In diesem Fall kann es helfen, eine andere Maßnahme auszuprobieren.

6 Fazit

Erfreulicherweise gibt über die Hälfte der Studierenden an, insbesondere diejenigen mit Programmiervorerfahrung, viel Spaß an der Programmierung zu haben, was ein positives Zeichen für die Wahl des Studienfaches ist. Damit kann eine Positivspirale in Gang gesetzt werden (siehe Abbildung 2), deren Bestandteile (Zeiteinsatz, Fähigkeiten/Verständnis, Erfolgserlebnisse, Spaß) alle paarweise und auch mit dem Klausurergebnis hohe Korrelationen aufweisen, wie wir statistisch aus den Erhebungsdaten berechnet haben. Wenn dem oder der Studierenden durch Selbsterkenntnis bewusst wird, dass der Mensch auf diese Weise durch Wiederholung und Spaß „funktioniert“ und lernfähig ist, so ist ein großer Schritt ins akademische Leben getan, bei dem durch selbstständige Aneignung Wissen erlangt werden kann.

Wie in der Einleitung angedeutet, wird im Modul „Programmiersprachen 1“ gerade darauf Wert gelegt, und zwar sowohl im Hinblick auf die theoretischen wie auch insbesondere die praktischen Inhalte – Letzteres über die durch Übungsaufgaben angeleitete Programmierung. Tatsächlich zeigt sich (siehe Tabelle 1), dass die Praxis gegenüber der Theorie einen Einfluss gemäß 2 zu 1 auf das Klausurergebnis der Studierenden hat und diese gleichzeitig ihre Zeit gemäß 40 zu 60 darauf aufteilen. Dieser bereits hohe praktische Zeiteinsatz von etwa 40 %, der noch erhöht werden könnte, wird durch ein Praktikum gefördert, in dem die Studierenden schon in der ersten Praktikumswoche und dann laufend herangeführt werden, eigenständig Quelltexte und Programme programmieren zu lernen („HalloWelt“-Textausgabe, Benutzereingaben verarbeiten, einfache mathematische Berechnungen usw.).

Aus den Ergebnissen kann man weiterhin folgern, dass eine Verbesserung der Klausurnote um $\Delta = 1{,}0$ entweder durch 2,5 Jahre Vorerfahrung oder durch einen zusätzlichen Gesamtzeiteinsatz von etwas mehr als 3 h/Woche erreicht werden kann. Folglich können auch mehrere Jahre der Programmiervorerfahrung durch die Studierenden durch entsprechend erhöhten Zeiteinsatz aufgeholt werden – oder wie der Volksmund zusammengefasst sagt: „Ohne Fleiß kein Preis!“ Laut eigener Aussage hat jedoch niemand der Studierenden die im Modul geforderte Arbeitsleistung von 150 h pro Semester erreicht, der Durchschnitt liegt bei weniger als der Hälfte dieser Zeit. Statistisch extrapoliert würde man bei Erbringen der vollen Arbeitsleistung eine Klausurnote von 1,7 erreichen, was unserer Meinung nach ein dafür angemessenes Ergebnis darstellt. Interessanterweise hat nicht der Zeiteinsatz der Studierenden die

höchste Korrelation mit der Verbesserung des Klausurergebnisses (+0,53), sondern der Spaß am Fach (+0,73).

Eine andere Studie gibt an, dass in Bezug auf den Prüfungserfolg der Zeiteinsatz eine untergeordnete Rolle spiele gegenüber der Zuordnung der Studierenden zu einem bestimmten Lernverhalten (Schulmeister, 2011). In einer Folgestudie kann entsprechend geprüft werden, inwiefern bei einer Zuordnung von Studierenden zu den verschiedenen Lernverhalten die hier genannten Korrelationen unterschiedlich ausfallen. Insbesondere die Korrelationen zum Einflussfaktor Zeiteinsatz sollten sich dann stark je nach Lernverhalten unterscheiden. Weiterhin können die hier vorgestellten Szenarien auf ihre Wirksamkeit hin geprüft und quantifiziert werden.

Das Ziel eines Studiums ist die Bildung von verantwortungsvollen Persönlichkeiten (Technische Hochschule Ostwestfalen-Lippe, 2018), die auch die Fähigkeiten und das Wissen ihres Studienfaches beherrschen. Gerade heute stehen dabei die „Future Skills" im Vordergrund, insbesondere bei den informatisch-technischen Studiengängen (ITG & GI, 2018). In diesem Beitrag wurden, basierend auf einem studierendenzentrierten Ansatz (Wright, 2011), Korrelationen zu messbaren Einflussfaktoren genannt, die entsprechende Ansätze zur konstruktiven Beeinflussung der Lehre der Programmierung ermöglichen. Ein erster Ansatz ist sicherlich das Bewusstmachen der gezeigten Positivspirale (siehe Abbildung 2), um bei den Studierenden einen intrinsischen Anreiz zu schaffen, selbstständig und selbstwirksam auf diese konstruktiv einzuwirken. Anschließend kann man bei jedem der einzelnen Einflussfaktoren ansetzen, indem man versucht, den Zeiteinsatz, die Fähigkeiten, die Erfolgserlebnisse oder den Spaß der Studierenden zu erhöhen. Entsprechende Beispielmaßnahmen wurden dazu genannt und mögliche Szenarien daraus entwickelt.

Danksagung

Wir danken Frau Kieu-Anh To für hilfreiche Anmerkungen zum Artikel.

Literatur

Barnes, K., Marateo, R. C. & Pixy Ferris, S. (2007). Teaching and learning with the net generation. *Innovate: Journal of Online Education*. Nova Southeastern University. https://nsuworks.nova.edu/innovate/vol3/iss4/1/, abgerufen am 08.01.2021

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education, 32*(3), 347–364.

Bloom, B. S., Engelhart, M. D. & Füner, E. (1973). *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz.

Detering, S., Dixon, D., Khaled, R. & Nacke, L. (2001). From game design elements to gamefulness: defining gamification. In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). ACM. https://doi.org/10.1145/2181037.2181040

Eller-Studzinsky, B., Magadi, M. & Thies, K. (2021). „Was machen eigentlich diese Lernscouts?" Lerngruppenarbeit im Selbststudium und in der Präsenzlehre. In T. Schmohl (Hrsg.), *Situiertes Lernen im Studium. Didaktische Konzepte und Fallbeispiele einer erfahrungsbasierten Hochschullehre*. (TeachingXchange, Bd. 5). Bielefeld: wbv media, S. 9–18.

Technische Hochschule Ostwestfalen-Lippe (2018). *Hochschulstrategie 2018*. https://www.hs-owl.de/fileadmin/downloads/PDFs/Hochschulstrategie_Gemeinsam_die_Zukunft_gestalten_2018_06_21.pdf

Technische Hochschule Ostwestfalen-Lippe (2019). *Modulhandbuch Bachelor Technische Informatik*. https://www.hs-owl.de/fb5/fileadmin/download/pdf/Studiengaenge/Modulhandbuch/Modulhandbuch_Ba-E-TI_4-11.pdf

Hattie, J. (2015). The applicability of Visible Learning to higher education. *Scholarship of Learning and Teaching in Psychology, 1*(1), 79.

ITG & GI (2018). *Curriculum für Bachelor- und Master-Studiengänge Technische Informatik*.

Land NRW (2019). Gesetz über die Hochschulen des Landes Nordrhein-Westfalen (Hochschulgesetz – HG). https://recht.nrw.de/lmi/owa/br_text_anzeigen?v_id=10000000000000000654, abgerufen am 08.01.2021

Loshkareva, E., Luksha, P., Ninenko, I., Smagin, I. & Sudakov, D. (2015). *Skills of the future: How to thrive in the complex new world*. https://futuref.org/futureskills/

Metzger, C., Schulmeister, R. & Martens, T. (2012). Motivation und Lehrorganisation als Elemente von Lernkultur. *Zeitschrift für Hochschulentwicklung, 7*(3), 36–50.

Norman, D. A. (1978). Notes Toward a Theory of Complex Learning. In A. M. Lesgold, J. W. Pellegrino, S. D. Fokkema & R. Glaser (Hrsg.), *Cognitive Psychology and Instruction*. Nato Conference Series. Springer.

Rupp, S., Pawlitzek, R. & Bach, C. (2017). Metriken zur Messung von Lernerfolg im Informatik-Grundlagen-Unterricht. In C. Igel, C. Ullrich & M. Wesner (Hrsg.), *Bildungsräume, DeLFI 2017 – Die 15. e-Learning Fachtagung Informatik, Lecture Notes in Information (LNI)*. Gesellschaft für Informatik.

Schulmeister, R. & Metzger, C. (2011). *Die Workload im Bachelor: Zeitbudget und Studierverhalten*. Waxmann.

Schulz, C. (2010). Entwicklung und Auswertung eines Fragebogens zur Langzeitlernerfolgskontrolle. *Zeitschrift für Hochschulentwicklung* 5(3),128–148.

Sembill, D. & Seifried, J. (2006). Selbstorganisiertes Lernen als didaktische Lehr-Lern-Konzeption zur Verknüpfung von selbstgesteuertem und kooperativem Lernen. In D. Euler, G. Pätzold & M. Lang (Hrsg.), *Selbstgesteuertes Lernen in der beruflichen Bildung* (S. 93–108). Steiner.

Stifterverband (2018). *Stifterverband für die deutsche Wissenschaft. Future Skills*. https://www.stifterverband.org/future-skills/framework, abgerufen am 08.01.2021

Stoyan, R. & Glinz, M. (2005). Methoden und Techniken zum Erreichen didaktischer Ziele in Software-Engineering-Praktika. In K.-P. Löhr & H. Lichter (Hrsg.), *Software Engineering im Unterricht der Hochschulen*. SEUH 9. d.verlag.

TIOBE (2019). *TIOBE-Index*. https://www.tiobe.com/tiobe-index/, abgerufen am 08.01.2021

Uni Konstanz (2014). *Kompetenzorientiert lehren und prüfen*. https://www.uni-konstanz.de/lehren/regularien/handreichungen-fuer-lehrende/kompetenzorientierung/, abgerufen am 08.01.2021.

Wright, G. B. (2011). Student-Centered Learning in higher education. *International Journal of Teaching and Learning in Higher Education, 23*(3), 92–97.

Zuse, H. (1999). *Geschichte der Programmiersprachen*. TU Berlin.

Autoren

Dr. rer. nat. Nils Beckmann
Fachbereich Elektrotechnik und Technische Informatik
nils.beckmann@th-owl.de

Prof. Dr.-Ing. Thomas Korte
Fachgebiet Informationstechnologie
thomas.korte@th-owl.de