# Zahlen, Beobachtungen und Fragen zur Programmierlehre

Axel W. Schmolitzky, HAW Hamburg

schmolitzky@acm.org

## Zusammenfassung

Dieser Artikel stellt Informationen zur Diskussion, die 2013 in einer Umfrage zur Programmierlehre erhoben wurden, und ergänzt diese um einige Zahlen, die der Autor in seiner eigenen Programmierlehre an zwei deutschen Hochschulen (einer Uni und einer Fachhochschule) gesammelt hat. Auf Basis dieser Zahlen sowie eigener Beobachtungen ergeben sich einige Fragen zum Thema Programmierlehre.

## 1 Einleitung

*Programmieren ist gewiss nicht alles, aber ohne Programmieren ist alles nichts. (frei nach Schopenhauer)*

Programmieren ist das Handwerkszeug der Softwaretechnik (Ludewig, 2010), die Programmierlehre bildet somit das Rückgrat jedes Informatikstudiums. Trotz dieses Stellenwertes wird dem Thema Programmieren traditionell auf Konferenzen zum Thema Software Engineering wenig Aufmerksamkeit geschenkt. Auf den SEUH-Tagungen der letzten 10 Jahre hat sich dies etwas verbessert, indem zunehmend auch Programmierthemen diskutiert wurden (Heuer et al., 2011; Langhoff et al., 2015; Schmedding et al., 2015). Dennoch haftet der grundständigen Programmierlehre häufig der Ruch des hemdsärmeligen, wenig wissenschaftlichen und teilweise sogar trivialen Anteils des Informatik-Studiums an. Um dem bewusst entgegen zu wirken, wird in diesem Beitrag nicht mehr der Begriff Programmier*ausbildung* benutzt, wie in früheren Artikeln (Schmolitzky, Züllighoven, 2007; Schmolitzky, 2013), sondern der passendere Begriff Programmier*lehre*.

Das Thema Programmierlehre befindet sich stärker im Fluss, als dies von vielen vermutet wird. Programmieren ist eine Kunst, entsprechend ist auch die Programmierlehre eine Kunst, die wir bei weitem noch nicht so gut beherrschen, wie es der Wichtigkeit der Disziplin angemessen wäre. Nach wie vor wissen wir zu wenig darüber, wie eine angemessene Programmierlehre aussehen sollte.

Der Autor hat als ersten Schritt einer Klärung nach der SEUH 2013 die Teilnehmer der Tagung zur Teilnahme an einer Erhebung aufgefordert, in der der aktuelle Stand der Programmierlehre an deutschsprachigen Hochschulen ermittelt werden sollte. Die Randbedingungen und Ergebnisse dieser Umfrage werden im nächsten Abschnitt präsentiert.

Abschnitt 3 stellt anhand konkreter Zahlen eigene Erfahrungswerte in der Programmierlehre des Autors zur Diskussion und Abschnitt 4 diskutiert weitere Aspekte, die dem Autor in Bezug auf Programmierlehre wichtig erscheinen.

## 2 Umfrage post SEUH 2013

Im Anschluss an die SEUH 2013 in Aachen hat der Autor eine E-Mail-Umfrage zur einführenden Programmierlehre unter den Teilnehmern der Tagung durchgeführt. Der Rücklauf umfasste 22 Antworten von Lehrenden an deutschsprachigen Hochschulen zu ca. 50 Bachelor-Studiengängen. Zum Vergleich: laut den Webseiten der GI ist ein Studium für einen Bachelor-Abschluss mit Informatik-Anteil allein in Deutschland an ca. 150 Hochschulen möglich, in fast 800 Studiengängen.

Eine erste Auswertung der Antworten präsentierte der Autor informell in einem Vortrag auf der SEUH 2015 in Dresden. Im Folgenden werden die Ergebnisse erstmals schriftlich zusammengefasst und bewertet.

### Gegenstand der Umfrage

Die Umfrage zielte darauf, in Bachelorstudiengängen mit hohem Informatik-Anteil den *Umfang*, die *Schwerpunkte* und die *Probleme* in der Programmierlehre zu erfassen. Konkret wurde nach *Pflichtmodulen*, vermittelten *Programmierparadigmen* und verwendeten *Programmiersprachen* gefragt (siehe Anhang A). Die Motivation des Autors war unter anderem, seinen subjektiven Eindruck, dass die Programmierlehre an deutschen Hochschulen oft stiefmütterlich behandelt und/oder kritisch von den „Abnehmern“ (Lehrende höherer Semester, Arbeitgeber) beurteilt wird, durch systematisch erhobene Hinweise zu bestätigen.

### Probleme der Umfrage

Die Auswertung der Antworten zeigte einige Probleme der Umfrage selbst auf. So war die Frage nach

„Modulen zur Programmierung“ offensichtlich nicht scharf genug gestellt, denn etliche Antworten nannten alle Module, in denen *auch* programmiert wird (wie Betriebssysteme oder Datenbanken), nicht nur solche, in denen die Programmierung selbst explizit *vermittelt* wird.

Ein weiterer Schwachpunkt war die Frage nach dem Umfang in SWS. Diese Kenngröße wird an deutschen Hochschulen ausgesprochen uneinheitlich verwendet und ist wenig aussagekräftig, insbesondere seit der Bologna-Reform. Für die Studierenden wurde die Währung SWS durch die neue Währung Leistungspunkte (LP) bzw. Credit Points (CP) ersetzt, die explizit die Aufwände der Studierenden bemessen soll. Ein generelles Problem (nicht nur dieser Umfrage) ist, dass die Umrechnung von SWS in LP mit hohem kreativen Potenzial auf sehr unterschiedliche Weise erfolgt. Vergleiche werden damit noch mehr erschwert.

Insgesamt kann aus den folgenden Ergebnissen nur ein erster Eindruck gewonnen werden.

## Ergebnisse der Umfrage

Die Umfrage lieferte Daten zu 48 Studiengängen, 18 davon an Universitäten und 30 an Fachhochschulen. Die häufigsten Studiengangsbezeichnungen sind *Informatik* mit 12 Nennungen und *Wirtschaftsinformatik* mit neun Nennungen. Alle anderen Studiengänge tragen „Bindestrichnamen“, von *Technischer Informatik* und *Medieninformatik* (jeweils drei Nennungen) bis hin zu *Informationslogistik* oder *Multimedia Marketing*. Bemerkenswert ist der hohe Anteil an „Insidern“ unter den antwortenden Lehrenden: 21 von 22 sind selbst in der Programmierlehre aktiv.

### *Umfang der einführenden Module*

Trotz stark heterogener Strukturen und Bezeichnungen bieten alle Studiengänge mindestens ein, meist zwei klar identifizierbare *einführende Pflichtmodule* zur Programmierung an. Der Umfang in Semesterwochenstunden (SWS) reicht dabei von 1+1 (Vorlesung + Übung bzw. Praktikum) bis zu 4+4, von hohem Vorlesungsanteil mit wenigen Übungsstunden bis hin zu kurzen Vorlesungen mit starkem Übungsanteil. Die fünf häufigsten Nennungen für Umfänge des ersten Moduls (M1) sind Abbildung 1 zu entnehmen, diese decken 42 der 48 Studiengänge ab.

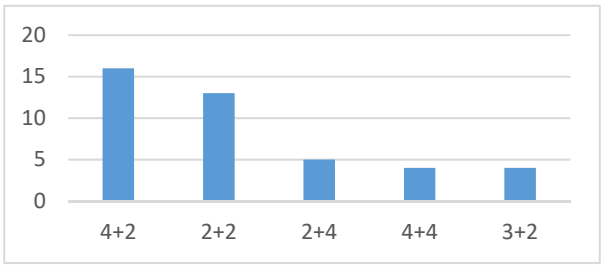


*Abb. 1: Umfang M1 (in SWS), häufigste Nennungen*

Das zweite Modul (M2) ist oft kleiner im Umfang, die fünf häufigsten Nennungen führt Abbildung 2 auf.

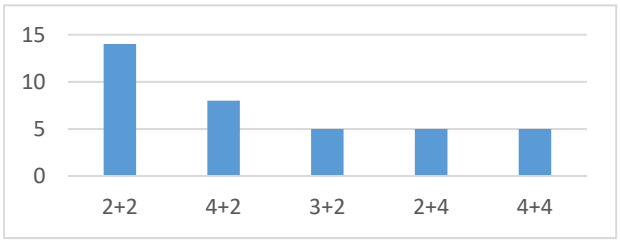


*Abb. 2: Umfang M2 (in SWS), häufigste Nennungen*

Erwartungskonform ist der hohe Übungs- bzw. Praktikumsanteil, nur sehr wenige Antworten (ca. 6%) beschrieben Module mit weniger als einem +2-Anteil.

Die *Anzahl* der Pflichtmodule zur Programmierung reicht von zwei bis zu sieben. Bei den reinen Informatik-Studiengängen gibt es nur einen mit lediglich zwei Pflichtmodulen, meist sind es drei oder vier. Bei den Wirtschaftsinformatik-Studiengängen hingegen sind zwei Pflichtmodule der Normalfall, nur zwei der neun Studiengänge haben drei Pflichtmodule.

### *Programmierparadigmen und -sprachen*

In allen 48 Studiengängen gehören *imperative und objektorientierte Programmierung* zur Pflicht. *Funktionale Programmierung* (FP) gehört nur in 13 Studiengängen zum Pflichtprogramm (9 davon an Unis) und ist in 8 Studiengängen optional, in 27 Studiengängen wird sie nicht angeboten, also in mehr als der Hälfte. Ein ähnliches Bild zeigt sich bei *logischer Programmierung* (LP): Sie ist Pflicht in 12 Studiengängen (8 davon an Unis) und optional in 8, nicht angeboten wird sie in 28 Studiengängen (Abbildung 3).

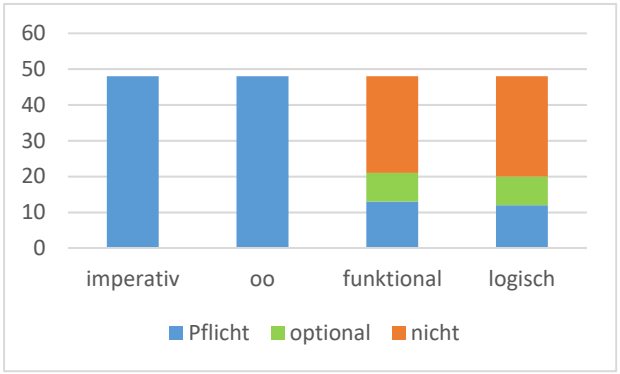


*Abb. 3: Angebotene Programmierparadigmen*

Die mit Abstand am häufigsten eingesetzte Programmiersprache ist *Java*. In M1 wird sie in 32 der 48 Studiengänge eingesetzt, als Alternative für den imperativen Einstieg wird 14-mal *C* genannt. *C++* für

den *Einstieg* in die Programmierung wird nur einmal genannt. Bei einem Einstieg mit funktionaler Programmierung (ausschließlich oder gemeinsam mit anderen Paradigmen: 4 Studiengänge) werden *Haskell* oder *Scheme* verwendet.

In M2 wird in 31 von 48 Studiengängen Java verwendet, alternativ kommen hier C, C++, Haskell, Scheme, *Prolog* und *CHR* zum Einsatz. In allen Studiengängen wird Java aber mindestens entweder im ersten oder im zweiten Semester eingesetzt, so dass jeder Student in seinem Studium Java kennenlernt.

***Zufriedenheit/Probleme***

Die überwiegende Mehrheit der Antwortenden zeigte sich zufrieden mit der an der eigenen Hochschule angebotenen Programmierlehre: 17 von 22 antworteten mit „klares ja", „ja" oder „eher ja" (Abbildung 4).

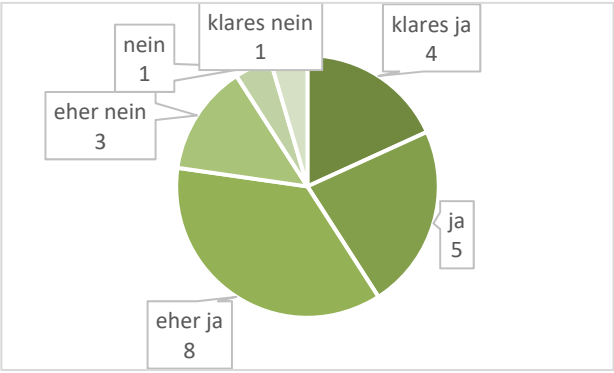


*Abb. 4: Zufrieden mit lokalem Angebot?*

Probleme mit der Programmierlehre konnten über Freitexte mitgeteilt werden. Die im Folgenden vollständig aufgeführten Rückmeldungen können grob unterschieden werden in solche zu Problemen der Studierenden (Motivation, Zeiteinsatz) und zu Problemen auf Seiten der Lehrenden (Organisation, Fokus, Praxisnähe).

**Probleme auf Seiten der Studierenden:**

*„[…] Das größere Problem ist für mich aber, dass die Studis zu wenig Präsenzübung haben (ist nun in XXX verstärkt geplant) und oft nicht genügend üben."*

*„Wir haben eine zu hohe Durchfallquote (40-50%). Studierende halten Programmierung für „einfach", sie fangen zu spät an, ernsthaft zu arbeiten."*

*„Teilweise fehlende Motivation der Studierenden (Programmierung gilt zu Beginn des Studiums vielfach neben Mathe als Angstfach), mangelnde Selbstständigkeit (zu Beginn des Studiums verbreitet wenig Motivation zur selbständigen Arbeit und zum Experimentieren)"*

*„Eingangskompetenzen der Studierenden gar nicht vorhanden bzw. unzulänglich hinsichtlich:*
*- Abstraktionsvermögen*
*- Algorithmischem Denken*
*- Ersten Programmiererfahrungen*
*Geringe Motivation für Informatik*
*Informatik ist sehr abstrakt für sehr viele Studierende*
*Informatik steht im Schatten anderer Module*
*Folge: Informatik ist wenig attraktiv"*

**Abstimmung zwischen den Lehrenden (fünf Antworten):**

*„Die Vorlesung Programmierung wird semesterweise wechselnd von verschiedenen Dozenten gehalten, die zum Teil deutlich unterschiedliche Schwerpunkte legen. Dementsprechend ist die Vorlesung mal sehr auf Algorithmen und Datenstrukturen konzentriert auf Kosten wichtiger Konzepte der Programmiersprache Java und mal sehr auf Java und seine vollständige Kenntnis konzentriert auf Kosten konzeptioneller Überlegungen. […]"*

*„Kein einheitliches Konzept unter den Dozenten vereinbar."*

*„Die Programmierausbildung ist über viele Dozenten verteilt und wird nicht gemeinschaftlich geplant. Das Resultat ist ein Flickenteppich aus nicht aufeinander abgestimmten Kursen, die vieles doppelt, widersprüchlich und/oder gar nicht behandeln."*

*„Wir haben im Wechsel zwei sehr unterschiedliche Lehrende, die sich in fast nichts einigen können. So steht IP gegen OOP, main-Methodenprogrammierung gegen Design Patterns, Blockpraktikum gegen schriftliche Klausur usw."*

*„Programmiertechniken werden losgelöst von Softwaretechnik betrachtet und sind oft nicht mehr auf dem aktuellen Stand der Forschung. Die Programmierausbildung (vlt. Lehre allgemein) wird auch nicht als wichtiges Betätigungsfeld, sondern eher als lästige Pflicht oder als Möglichkeit zur Nachwuchsrekrutierung gesehen."*

**Zu geringe Praxisnähe (vier Antworten):**

*„Es besteht noch Verbesserungspotential bei der Unterstützung der Programmieranfänger ohne Vorkenntnissen."*

*„Die Programmierausbildung könnte praxisnäher sein."*

*„Ich erachte vor allem die sehr geringe Programmierpraxis, die in der Vergangenheit in XXX vorgesehen war und den großen Sprung zu YYY (3 Semester später), als ein signifikantes Problem. Von Studierenden, die kaum programmieren konnten, wurde erwartet, dass sie direkt mit objektorientiertem Software-Entwurf beginnen."*

*„Trotz SoPra haben viele Studierende die grundlegenden Prinzipien nicht verstanden. Meiner Meinung nach liegt ein großes Problem darin, dass den Studierenden zwar das „Handwerkszeug" mitgegeben wird, allerdings werden die Studierenden häufig alleine gelassen, wenn es um die Anwendung der erlernten Prinzipien geht. Insbesondere das Vorgehen um von einer Problemstellung auf eine konkrete Lösung, welche verschiedene einzelne „Techni-*

*ken" der Programmierparadigmen kombiniert, zu kommen, bleibt unklar. Die separierte, rein „technische" Vermittlung der Programmierkonstrukte trägt sicherlich dazu bei, die Zusammenhänge aus dem Auge zu verlieren."*

**Zu einseitige Lehre (zwei Antworten):**

*„Mit FP und LP kommen unsere Studierenden ausschließlich im Master in Berührung, wobei FP ausschließlich im Anwendungskontext gelehrt wird und nicht als grundsätzliches Paradigma. Da unsere beiden Studiengänge dem Namen nach anwendungsorientiert sind, ist dies auch so gewünscht und sinnvoll - schade ist es trotzdem, da so wichtige Grundlagen zum Blick über den objektorientierten Tellerrand nicht gelegt werden."*

*„Im Moment läuft es auf eine sehr Java-lastige Programmierausbildung hinaus. Es ist noch nicht so klar, wo die Studenten Erfahrungen mit anderen Sprachen sammeln."*

### Bewertung der Umfrage

Dass die Mehrheit der antwortenden Lehrenden mit der angebotenen Programmierlehre zufrieden ist, überrascht nicht: fast alle Antwortenden sind selbst in der Programmierlehre aktiv. Offensichtlich haben nur Personen, die eine starke Bindung zum Thema Programmierlehre haben, die Mühe auf sich genommen, die detaillierten Fragen (siehe Anhang 1) zu beantworten. Die vom Autor in einzelnen Gesprächen aufgefangene Unzufriedenheit mit den Programmierfähigkeiten der Teilnehmer fortführender Veranstaltungen konnte so nicht systematisch erfasst werden. Andererseits ist es für den Autor ein gutes Zeichen, wenn auf einer Software-Engineering-Tagung eine starke Bindung zum Programmieren deutlich wird.

## 3 Studiengänge im Vergleich

Im Rahmen seiner eigenen Lehrtätigkeit hat der Autor Erfahrungen mit mehr als zehn verschiedenen Studiengängen gesammelt, von denen er einen selbst mitgestaltet hat. Unter den Studiengängen waren Diplom-, Bachelor- und Master-Studiengänge in zwei deutschen Bundesländern sowie einer in Australien.

An der Uni Hamburg hat der Autor vor allem in den Jahren 2005 bis 2013 die einführenden Programmierveranstaltungen der neuen Bachelor-Studiengänge mit aufgebaut und durchgeführt (Schmolitzky, Züllighoven, 2007; Schmolitzky, 2013).

Nach seinem Wechsel an die HAW Hamburg im Jahr 2014 hat der Autor dort drei Einführungszyklen durchgeführt (bestehend aus *Programmieren 1* und *Programmieren 2* in den ersten beiden Semestern), in allen drei an der HAW Hamburg angebotenen Informatik-Bachelor-Studiengängen: in der *Angewandten Informatik* (AI), in der *Technischen Informatik* (TI) und zuletzt in der *Wirtschaftsinformatik* (WI).

In diesem Abschnitt werden drei Vergleiche vorgenommen: Als erstes zwischen den Teilnehmern an der Abschlussprüfung einer Programmierveranstaltung des ersten Semesters an der Uni Hamburg, gruppiert nach ihren Studiengängen; die hohen Teilnehmerzahlen lassen hier verallgemeinernde Rückschlüsse zu. Der zweite Vergleich basiert auf einer Programmierprüfung, die der Autor sowohl an der Uni Hamburg als auch an der HAW Hamburg durchgeführt hat. Als drittes werden die genannten Programmierveranstaltungen an der HAW Hamburg zwischen den erwähnten Studiengängen verglichen.

### Vergleich zwischen Uni-Studiengängen

An der Uni Hamburg besteht die einführende Programmierlehre aus den Modulen *Softwareentwicklung 1 und 2* (SE1 und SE2); an diesen Modulen war der Autor von 2005 bis 2013 maßgeblich beteiligt. In SE1 und SE2 gibt es Teilnehmer aus inzwischen fünf Bachelor-Studiengängen: *Informatik* (Inf), *Wirtschaftsinformatik* (WInf), *Software-System-Entwicklung* (SSE), *Mensch-Computer-Interaktion* (MCI) sowie *Computing in Science* (CiS). Hinzu kommen die *Lehramt*-Studierenden (LA) sowie als weitere große Zielgruppen *Mathematik*- (MA) und *Physik*-Studierende (PH) mit Nebenfach Informatik. In den Jahren 2009 (Start der Studiengänge SSE, MCI und CiS) bis 2013 hat der Autor bei der ersten Prüfung die Klausurergebnisse nach Studiengängen differenziert, so dass ein Ranking der Studiengänge nach der Durchschnittsnote ihrer Kohorte möglich wurde. Die Anzahl der Klausurteilnehmer lag zwischen 290 (im Jahrgang 2009/2010) und 376 (2013/2014). Abbildung 5 zeigt die ermittelten Durchschnittsnoten in der Übersicht.

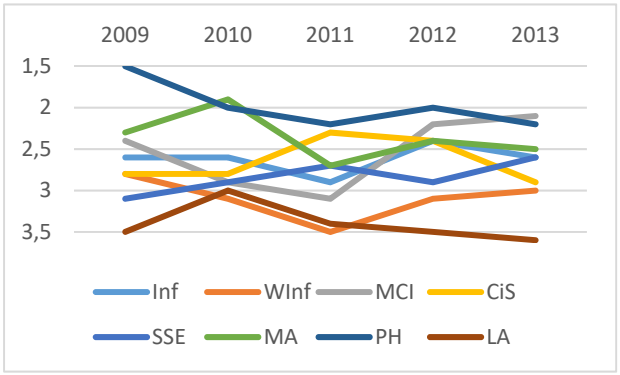


*Abb. 5: Kohortenschnitte SE1 (Uni HH)*

Neben der Idee, beim Blick auf diese Zahlen spontan einen Informatik-Studiengang mit einem Titel wie „Physik-Informatik" gründen zu wollen, um talentierte Programmierer unmittelbar anzusprechen, fällt insbesondere auf, dass Wirtschaftsinformatik

und Informatik auf Lehramt reproduzierbar die Schlusslichter in der Programmierlehre bilden. Die Zahlen sprechen hier eine sehr deutliche Sprache, eine Erklärung liefern sie nicht.

## Punktueller Vergleich Uni und HAW

Die einführende Veranstaltung *Programmieren 1* (PR1) in den Informatik-Studiengängen an der HAW Hamburg hat ein ungewöhnliches Format: 6V+2P, also einen hohen Vorlesungsanteil und einen vergleichsweise geringen Praxisanteil. Formal besteht sie aus zwei Modulen (*Programmiermethodik* und *Programmiertechnik*) und hat einen Umfang von 12 CP. In dieser Hinsicht ist sie somit doppelt so „groß" wie die Veranstaltung SE1 der Uni Hamburg, die lediglich 6 CP umfasst (bei 2V+2Ü). Da der Praxisanteil jedoch gleich groß ist, hat der Autor in seinem ersten PR1-Durchgang an der HAW Hamburg die gleichen Inhalte vermittelt wie in SE1 an der Uni, diese jedoch sehr viel ausführlicher in seminaristischer Form mit den Studierenden im Vorlesungsanteil diskutiert.

Ein weiterer Unterschied zur Uni besteht darin, dass in PR1 für beide Module je eine Prüfung stattfindet. Von diesen wird eine als klassische Klausurprüfung durchgeführt, während die andere traditionell als *Laborprüfung* erfolgt (mit hohem technischen und organisatorischen Aufwand, der u.a. von den wissenschaftlichen Mitarbeitern des Departments Informatik erbracht wird). In der Laborprüfung müssen die Teilnehmer am Rechner unter kontrollierten Bedingungen (kein Internet-Zugang) Programmieraufgaben unter Zeitdruck lösen. Für den Autor war diese Form der Prüfung neu.

Unter den bisher genannten Bedingungen hat der Autor bewusst ein Experiment durchgeführt: Da die vermittelten Inhalte vergleichbar waren, hat er die *exakt gleiche* Klausurprüfung am Ende von PR1 durchgeführt, die er auch an der Uni im letzten SE1-Durchgang verwendet hatte; es handelte sich tatsächlich sogar um *dasselbe* PDF-Dokument.

Auf diese Weise konnte unter vergleichbaren Bedingungen (gleicher Dozent, gleiche Inhalte) ein direkter Vergleich zwischen Uni- und HAW-Studierenden erfolgen. Die erreichten Noten unterschieden sich deutlich: Der Schnitt an der Uni lag bei 2,7, an der HAW bei 3,3.

Man könnte versucht sein, aus diesem Ergebnis auf grundsätzliche Leistungsunterschiede zwischen Uni- und HAW-Studierenden zu schließen. Der Autor sieht dies aber kritisch: An der Uni war das Format der Klausurprüfung seit vielen Jahren bekannt, an der HAW war es neu; weiterhin wurden den Studierenden an der Uni Online-Selbsttests zur Vorbereitung auf die Klausur angeboten, die ebenfalls das gleiche Format aufwiesen; und bei einem neu berufenen Professor laufen typischerweise viele Wiederholer auf, die mit den bisherigen Veranstaltern bzw. ihren Inhalten Probleme hatten und somit den Schnitt verfälschen können. Andererseits muss beachtet werden, dass der Umfang von PR1 zwei Fünfteln des ersten Semesters entspricht, während SE1 lediglich ein Fünftel des ersten Semesters an der Uni ausmacht.

Insgesamt gilt es also, auch diese Zahlen mit Vorsicht zu genießen.

## HAW-Studiengänge im Vergleich

Der Autor hat die Folge PR1 und PR2 (letzteres ist mit 3V+1P und 6 CP deutlich kleiner als PR1) in allen drei Bachelor-Studiengängen der Informatik an der HAW durchgeführt und dabei das didaktische Konzept sowie die Inhalte beibehalten. Dabei traten überraschend deutliche Unterschiede zwischen Studierenden der AI und TI einerseits und Studierenden der WI andererseits zutage.

| Studiengang | Teilnehmer | Schnitt | nicht bestanden |
|---|---|---|---|
| AI | 60 | 3,3 | 20% |
| TI | 49 | 3,0 | 20% |
| WI | 49 | 3,3 | 22% |

*Tabelle 1: PR1-Klausuren (HAW)*

Während die Klausurergebnisse sich nicht wesentlich unterschieden (siehe Tabelle 1), war die Nichtbestehensquote in der Laborprüfung bei den Wirtschaftsinformatikern im Vergleich überraschend hoch (siehe Tabelle 2).

| Studiengang | Teilnehmer | Schnitt | nicht bestanden |
|---|---|---|---|
| AI | 56 | 3,2 | 38% |
| TI | 52 | 3,4 | 44% |
| WI | 42 | 3,8 | **64%** |

*Tabelle 2: PR1-Laborprüfungen (HAW)*

Dies war umso erstaunlicher, als den Studierenden der WI-Laborprüfung die Prüfungen der beiden vorherigen Durchgänge zur Verfügung standen, sie das Format der Prüfung (das in allen drei Prüfungen sehr ähnlich war) also besser vorhersehen konnten als ihre Kommilitonen.

Irritiert durch dieses schlechte Abschneiden, suchte der Autor nach weiteren Hinweisen, dass in der WI andere Bedingungen herrschen, und wurde in den Lehreevaluationen fündig.

Während sich diese zwischen den Studiengängen bei vielen Punkten nur unwesentlich unterschieden, gab es bei der Einschätzung der Aussage *Die*

*Stoffmenge ist dem zeitlichen Rahmen angemessen* klare Unterschiede: Die AI-Teilnehmer bewerteten dies mit einem Mittelwert von 1,76 auf der Lickert-Skala, die TI sogar mit 1,48; die WI-Angaben hingegen lieferten einen Mittelwert von 2,36. Auch bei allen weiteren Angaben zur investierten Zeit, sowohl für die Vor- und Nachbereitung der Vorlesung als auch des Praktikums, lagen die Angaben bei den Wirtschaftsinformatikern deutlich höher.

Offensichtlich hängt somit der Erfolg beim Programmieren-Lernen nicht nur von der investierten Zeit ab, sondern auch von der persönlichen Motivation; WI-Studierende haben häufig die Haltung, dass (eigenes) Programmieren für ihre spätere Berufspraxis eine geringere Rolle spielt; dies zeigten etliche persönliche (nicht repräsentative) Gespräche, die der Autor mit Studierenden geführt hat.

Dies wurde auch deutlich in einer anonymen Erhebung, die der Autor jeweils im letzten Vorlesungstermin von PR1 in allen drei Studiengängen durchgeführt hat. Die letzte zu bewertende Aussage der Erhebung lautet: *Dieses Modul hat meine Lust auf Informatik gesteigert*. Die Studierenden konnten darauf mit *Ja*, *Eher ja*, *Eher nein* oder *Nein* antworten. Während in der AI (97% ) und der TI (100%) fast alle mit *Ja* oder *Eher ja* antworteten, waren es in der WI lediglich 75%; mehr als 20% antworteten sogar mit einem klaren *Nein*.

### Diskussion

In seiner Zeit an der Uni Hamburg hat sich der Autor vergleichsweise wenig mit dem (klar ersichtlichen) „Fremdeln" der Wirtschaftsinformatiker mit dem Programmieren beschäftigt. Es schien ihm klar, dass der Erfolg beim Programmierenlernen direkt in Abhängigkeit zur investierten Zeit steht. Erst durch die intensive Erfahrung, an der HAW Hamburg ein Jahr lang ausschließlich Wirtschaftsinformatiker zu unterrichten, wurde dem Autor vor Augen geführt, dass ein entscheidendes Kriterium neben der investierten Zeit auch die Motivation ist.

Studierende der Wirtschaftsinformatik empfinden Programmieren häufig, zumindest für sich selbst, als nicht so wichtig. Dies hat einen messbaren Einfluss auf ihren Erfolg in entsprechenden Veranstaltungen.

Wirtschaftsinformatiker sind somit *im Schnitt* schlechtere Programmierer. Dies liegt nicht nur daran, dass sie insgesamt einen kleineren Informatikanteil in ihrem Studium haben, sondern vermutlich auch an der mehrfach beobachteten Grundhaltung, dass sie in ihrem späteren Berufsbild eher keine eigene Programmiertätigkeit sehen. Inwieweit dies der Realität der Berufspraxis entspricht, kann hier nicht beurteilt werden. Möglicherweise sind Wirtschaftsinformatiker mit einem gesunden Halbwissen der Programmierung zumindest gute Vermittler zwischen den fachlichen Anforderungen im IT-Umfeld und den technischen Fähigkeiten gut ausgebildeter Vollinformatiker.

Zumindest in der Informatik an der HAW Hamburg wurde intern mehrfach die Frage gestellt, ob auf die hier beschriebenen Gegebenheiten mit einem speziellen Angebot für Wirtschaftsinformatiker eingegangen werden sollte. Bisher wurde dies stets verneint, mit dem Hinweis, dass alle Informatik-Studierenden die gleiche grundlegende Programmierlehre erhalten sollten. Dass die Wirtschaftsinformatiker hierbei schlechter abschneiden, wird als verkraftbar angesehen. Diese Haltung wird gestützt durch die Ergebnisse der Umfrage aus Abschnitt 2: Acht der neun WI-Studiengänge verwenden für den Einstieg in die Programmierung die gleichen beiden Module, die auch in der Kern-Informatik zum Einsatz kommen; lediglich in einem der Studiengänge wird ein abgespecktes WI-Paket angeboten (in dem aber lediglich die Vorlesungsanteile halbiert sind).

## 4 Fragen zur Entwicklung

Nach Eindruck des Autors befindet sich die einführende Programmierlehre in einem Spannungsfeld. Einerseits haben sich die *Studierenden* über die letzten Jahre stark verändert:

- Sie sind aufgrund der teilweise um ein Jahr verkürzten Schulzeit und des Wegfalls von Wehr- bzw. Zivildienst deutlich jünger als früher; teilweise sitzen 17-jährige in den Erstsemesterveranstaltungen, in denen früher kaum jemand jünger als 20 Jahre war.
- Eine heutige Erstsemesterkohorte setzt sich anders zusammen als früher: Vor den Zeiten des Internet-Booms studierten vor allem die Personen Informatik, die intrinsisch an ihren Themen interessiert waren; heute gibt es viele Studierende, die vor allem die wirtschaftlichen Vorteile eines Informatik-Studiums sehen.

Diese beiden Einflüsse senken die durchschnittliche *Studierfähigkeit* der Studierenden.

Andererseits steigt der Anspruch an die *Ergebnisse* der Programmierveranstaltungen: Kollegen in höheren Semestern klagen häufig, dass die Teilnehmer in ihren Veranstaltungen wie *Verteilte Systeme* oder *Software Engineering* nicht gut genug programmieren können. Damit einhergehend steigt auch der Anspruch an die abgedeckten *Inhalte*:

- *Funktionale Programmierung* wanderte mit der Version 8 in Java hinein, entsprechend sollten auch die passenden Sprachkonzepte thematisiert werden.
- Quelltextverwaltungssysteme wie *Git* werden immer wichtiger, entsprechend sollte auch der Umgang mit Git früh thematisiert werden.

- die Java-Bibliotheken können teilweise nur mit guten Kenntnissen von *Entwurfsmustern* kompetent benutzt werden, entsprechend müssen Entwurfsmuster thematisiert werden.
- In Zeiten von Mehrkernsystemen wird *nebenläufige Programmierung* wieder wichtiger – diese war schon immer in den Genen von Java, spielt aber in den meisten einführenden Modulen nur eine geringe Rolle.

Darüber hinaus stellt sich die Frage: Wo sollen Programmierinhalte *jenseits von Java* diskutiert werden? Können grundlegende Sprachkonzepte, die Java nicht anbietet (z.B. andere Konzepte zur Parameterübergabe, Wertsemantik bei benutzerdefinierten Typen, explizite Speicherverwaltung), überhaupt noch im Pflichtprogramm vermittelt werden? Vor allem angesichts der Tendenz, immer mehr Inhalte in der Programmierlehre in immer kleinere Anteile des Curriculums zu pressen, weil andere Inhalte des Studiums zunehmend als wichtiger empfunden werden, in letzter Zeit beispielsweise, mit zugegebener Berechtigung, die IT-Sicherheit?

Diese Überlegungen führen zu einer Reihe von Fragen, die bezüglich der Programmierlehre in den nächsten Jahren beantwortet werden sollten:

- Welchen Einfluss hat die starke Verjüngung der Studienanfänger aufgrund des Wegfalls von Wehr- bzw. Zivildienst und der Verkürzung der Schulzeit in vielen Bundesländern?
- Wie problematisch ist der Umstand, dass die Programmierlehre ihren Grundstein im ersten Semester gelegt bekommt, in dem viele Studierende häufig noch nicht studierfähig sind?
- Können wir es uns leisten, die Programmierlehre weiter zu beschneiden? Oder sollte ihr angesichts der gestiegenen inhaltlichen Ansprüche wieder mehr Gewicht gegeben werden?

Wenn wir die zu Anfang dieses Abschnitts beschriebenen Randbedingungen akzeptieren (müssen), dann sollten wir uns bezüglich unserer einführenden Veranstaltungen weitere Fragen stellen:

- Welche organisatorischen Rahmenbedingungen können den Einstieg in die Programmierung erleichtern?
- Welche didaktischen Möglichkeiten bieten sich uns, die jungen Studierenden für das Programmieren stärker zu begeistern?
- Welche Chancen bieten *interaktive Systeme*, die gerade mit Java gut vermittelt werden können, für eine interessante Programmierlehre?
- Welche *Werkzeuge* fehlen uns noch für eine bessere Programmierlehre?
- Sollten wir, je nach Schwerpunkt des Studiengangs, eher einen Fokus auf Anwendungs- oder auf Systemprogrammierung legen?
- Wie schaffen wir es, Problemlösen mit Computern *und* Sprachkonzepte zu vermitteln?

Nach Ansicht des Autors gibt es hier ausreichend Klärungsbedarf für die nächsten Jahre, sowohl in der Forschung als auch in der Lehre.

## Fazit

Dieser Artikel beleuchtete in drei Abschnitten verschiedene Facetten der Programmierlehre. Abschnitt 2 zeigte als zentrale Ergebnisse einer Umfrage, dass Java-Kenntnisse eine unumstrittene Rolle spielen und dass Lehrende von Programmierveranstaltungen mit ihrer eigenen Lehre überwiegend zufrieden sind. Offen bleibt, ob letzteres auch für die verschiedenen „Abnehmer" (Studierende, Lehrende höherer Semester, Arbeitgeber) gilt. Abschnitt 3 arbeitete heraus, dass insbesondere Wirtschaftsinformatiker, stellvertretend für viele Bindestrich-Informatiker, im Schnitt schlechtere Programmierer sind. Hier bleibt zu klären, ob darauf mit speziellen Lehrangeboten eingegangen werden sollte. In Abschnitt 4 schließlich wurde dargestellt, dass die Programmierlehre aktuell vor spezifischen Herausforderungen steht, auf die wir die richtigen Antworten noch finden müssen.

## Literatur

Heuer, A., et al. (2011): Entwicklung eingebetteter Software in einem Softwarepraktikum mit Lego Mindstorms. Software Engineering im Unterricht der Hochschulen (SEUH), Hannover, dpunkt.verlag, Heidelberg.

Langhoff, A., et al. (2015): Java, LEDs und ein RaspberryPi: Ein Projektversuch mit Erstsemestern. Software Engineering im Unterricht der Hochschulen (SEUH), Dresden, CEUR Workshop Proceedings 1332.

Ludewig, J. (2010): Software-Ingenieur werden. Informatik Spektrum 33(3), S. 288-291.

Schmedding, D., et al. (2015): Clean Code - ein neues Ziel im Software-Praktikum. Software Engineering im Unterricht der Hochschulen (SEUH), Dresden, CEUR Workshop Proceedings 1332.

Schmolitzky, A., Züllighoven, H. (2007): Einführung in die Softwareentwicklung: Softwaretechnik trotz Objektorientierung? Software Engineering im Unterricht der Hochschulen (SEUH). Zeller, A., Deininger, M. Stuttgart, dpunkt.verlag.

Schmolitzky, A. (2013): Eine softwaretechnische Programmierausbildung? Software Engineering im Unterricht der Hochschulen (SEUH), Aachen, CEUR Workshop Proceedings 695.

## Anhang A: Vollständiger Text der E-Mail-Umfrage vom März 2013

Formaler Rahmen:

Wie lautet Ihr vollständiger Name?

Wie heißt Ihre Professur/Ihr Lehrstuhl/Ihre Gruppe?

Sind Sie/Ihre Gruppe selbst in (Teilen) der Programmierausbildung aktiv?

Wie heißt die Hochschule, über die Sie berichten?

Welche Bachelor-Studiengänge (jeweils mit Kürzel) mit hohem Informatik-Anteil gibt es an Ihrer Hochschule, über deren Programmierausbildung Sie berichten können?

Wie viele Pflichtmodule zur Programmierung gibt es in diesen Studiengängen? (nicht alle müssen in allen Studiengängen Pflicht sein)

Wie heißen diese Module (vollständiger Name und evtl. Modulkürzel)?

In welche Studiengängen sind welche Module Pflicht?

Welchen Umfang haben die Module?
Antwortformat: "Kürzel: 0-9 V(orlesung) + 0-9 Ü(bung)+ 0-9 P(raktikum)"

Welche Paradigmen werden in den Pflichtmodulen gelehrt?
Antwortmöglichkeiten: IP (Imperative Programmierung), OOP (Objektorientierte P.), FP (Funktionale P.) LP (Logische P.)

In welchem Modul werden welche Paradigmen vermittelt?

Welche Programmiersprachen werden in welchem Modul vermittelt?

Wenn es weitere Module zur Programmierung im Wahlpflicht- oder Wahlbereich gibt:
Wie heißen diese (mit Kürzel), welche Paradigmen werden vermittelt, welchen Umfang haben sie?

Persönliche Einschätzung:

Sind Sie persönlich mit der angebotenen Programmierausbildung zufrieden?

Antwortmöglichkeiten: klares ja - ja - eher ja - eher nein - nein - klares nein

Falls Sie nicht zufrieden sind: Welche Probleme sehen Sie? (Freitext)

Wünschen Sie eine Anonymisierung Ihrer Antworten?

Antwortmöglichkeiten: ja, alle - lediglich meine persönliche Einschätzung - nein