Informatik für Ingenieure an der Technischen Universität Hamburg

Bernhard J. Berger[1] und Goerschwin Fey[2]

Abstract: Diese Veröffentlichung stellt die Grundlagenveranstaltung *Informatik für Ingenieure* der Technischen Universität Hamburg vor, die für mehrere Ingenieurstudiengänge als Einführung in die Informatik und das Programmieren dient. Dabei ist es wichtig, in welcher Tiefe und Zusammenstellung die einzelnen Themeninhalte vermittelt werden. Die Veranstaltung befindet sich – wie viele *Hilfsveranstaltungen*– im Zielkonflikt zwischen Vermittlung zum tiefgehenden Verständnis weniger Inhalte und breit angelegtem Überblick vieler Inhalte. Somit gibt es zwei Ziele: (1) Studierende sollen mit Informatik-Fachbegriffen umgehen können, (2) Studierende sollen grundlegende Programmierkenntnisse erwerben. Der finale Leistungsnachweis mit theoretischen und praktischen Inhalten wird mithilfe einer Online-Klausur mit automatisierter Auswertung absolviert. Semesterbegleitende Tutorien und Zwischenprüfung bereiten die Studierenden auf die abschließende Klausur vor. Durch die Herausforderungen für die Veranstaltung ist diese in einem ständigen Wandel. Dieses Papier gibt einen Überblick über die Konzeption der Veranstaltung, die in der Diversität der beteiligten Studiengänge begründeten Herausforderungen und stellt eine automatisierte nicht-klausurspezifische Klausurauswertung zur Verfügung.

Keywords: Informatik, Ingenieure, Programmieren, Klausur, statische Auswertung

1 Einleitung

In vielen Ingenieurberufen ist es heutzutage selbstverständlich, ein grundlegendes Maß an Programmierkenntnissen vorauszusetzen. Vergleichbares lässt sich über ein prinzipielles Verständnis der Informatik und ihrer Teilgebiete sagen. Dies liegt in unterschiedlichen Veränderungen der letzten Jahrzehnte begründet: Zum einen ist die Etablierung des Prinzips der Industrie 4.0 [Mo22] ein entscheidender Grund, zum anderen aber auch der erhöhte Einsatz von eingebetteten Systemen, sowie das allgegenwärtige Paradigma Cyber-Physischer Systeme. Ein weiterer Aspekt ist die immer stärker verbreitete Anwendung von Machine Learning in allen Bereichen unseres alltäglichen Lebens, was somit auch diejenigen betrifft, die die Technik, die wir alltäglich nutzen, konzipieren und realisieren. Je nach Ausprägung des Ingenieurberufes ist das fundierte Wissen in unterschiedlichen Teilgebieten der Informatik von Vorteil oder sogar essenziell, um die Möglichkeiten eines Systementwurfes einzuschätzen. Diese Entwicklung hat dazu geführt, dass in den meisten ingenieurbezogenen Studiengängen mindestens ein Modul mit Informatikbezug und der Vermittlung von

[1] Technische Universität Hamburg, Institut für Eingebettete Systeme, Schwarzenberg-Campus 3, 21073 Hamburg, Germany, bernhard.berger@tuhh.de, https://orcid.org/0000-0001-6093-9229

[2] Technische Universität Hamburg, Institut für Eingebettete Systeme, Schwarzenberg-Campus 3, 21073 Hamburg, Germany, goerschwin.fey@tuhh.de, https://orcid.org/0000-0001-6433-6265

Programmierkenntnissen in den jeweiligen Studienverlaufsplänen vorhanden ist, um die Studierenden möglichst umfassend auf ihre späteren Berufe vorzubereiten.

Hier setzt auch das Modul *Informatik für Ingenieure – Einführung & Überblick* der Technischen Universität Hamburg an. Dieses Modul ist eine Pflichtveranstaltung für Studierende unterschiedlicher Studiengänge im Bereich der Ingenieurwissenschaften, die in dieser Form seit dem Wintersemester 2021/22 besteht.

Ursprünglich existierten in unterschiedlichen ingenieurbezogenen Studiengängen unterschiedliche Informatik-Veranstaltungen, welche jedoch zum Wintersemester 2021/22 zusammengefasst wurden, um einheitliche Lerninhalte der an der Technischen Universität Hamburg in Bezug auf die Informatik sicherzustellen. Die Veranstaltung in ihrer aktuellen Form ist für die Studiengänge Maschinenbau, Schiffbau, Logistik und Mobilität, Elektrotechnik, Allgemeine Ingenieurwissenschaften und General Engineering Science im Bachelorstudium verpflichtend. Da manche dieser Studiengänge an der Technischen Universität Hamburg als englischsprachige Studiengänge angeboten werden, wird das Modul *Informatik für Ingenieure* sowohl auf Deutsch als auch auf Englisch angeboten. Dies bezieht sich sowohl auf die Vorlesung als solche, als auch auf jegliches Übungsmaterial und die Klausur. Die Modulbeschreibung ist online verfügbar [Te21]. Ein optionales teilweise konsekutives Modul *Informatik für Ingenieure – Programmierkonzepte und Data Handling* wird in einigen Studiengängen genutzt, aber im Rahmen der vorliegenden Arbeit nicht diskutiert.

Das Modul besteht aus einer Vorlesung mit Übungsbetrieb. Die Vorlesung findet im Jahresturnus jedes Wintersemester statt, da sie in den meisten Studiengängen im Studienverlaufsplan als Modul für das erste Studiensemester ausgewiesen wird. Dies führt zu einer durchschnittlichen Teilnehmendenzahl von 600–700 Studierenden in jedem akademischen Jahr. Um eine gerechte und vergleichbar nachvollziehbare Bewertung der Semesterendprüfung bei dieser Größenordnung der Teilnehmendenzahl zu gewährleisten, wurde die konzeptionelle Entscheidung für eine automatisierte Auswertung getroffen. Die Klausur prüft unterschiedliche Bereiche des Modulinhalts in unterschiedlichen Komplexitätsstufen ab. Da pro Gesamtprüfung mehrere Klausuren geschrieben werden (sowohl mit unterschiedlichen Aufgaben, als auch als Wiederholung der gleichen Klausur mit unterschiedlichen Teilnehmenden), wird neben der konventionellen Auswertung der Klausuren ein statistischer Vergleich der jeweiligen Resultatverteilungen durchgeführt, um ein vergleichbares Anforderungsniveau sicherzustellen. Gleichzeitig liefert die innere Analyse der Klausurergebnisse Anregungen für mögliche Verbesserungen der Veranstaltung, sowie einen Vergleich über die Jahre hinweg (zeitliche Analyse). Die gesammelte statistische Auswertung führen wir über die Statistik-Software *R* durch und bereiten die Resultate graphisch auf. Partiell verwenden wir diese Darstellungen auch innerhalb von Evaluationsgesprächen mit den Studierenden.

Die vorliegende Veröffentlichung umfasst die folgenden Beiträge:

1. Beschreibung der Lehr- und Lerninhalte der Veranstaltung *Informatik für Ingenieure*,

2. Beschreibung einer lehrveranstaltungsunabhängigen und quelloffenen statistischen Klausurauswertung zur besseren Evaluation und besserem Vergleich von Klausuren,
3. Identifikation von aktuellen Herauforderungen der Veranstaltung und Diskussion der Entwicklung der Veranstaltung zur Begegnung dieser Herausforderungen.

2 Organisation der Lehrveranstaltung

Die Vorlesung zum Modul *Informatik für Ingenieure* findet in jedem Wintersemester mit begleitdendem Übungsbetrieb statt. Es ist aber gemäß Regelungen an der TUHH in jedem Semester möglich, eine Prüfung abzulegen. Daher wird im Sommersemester ein Repetitorium angeboten, das sich aber in der Struktur auf Grund unserer Erfahrung von der eigentlichen Vorlesung unterscheidet. Außerdem bieten wir eine eintägige Veranstaltung im Rahmen des Brückenkurses zum Studieneinstieg an.

In Rahmen der eintägigen Veranstaltung vor Beginn der Vorlesung bieten wir eine erste Einführung in das Programmieren an [Te]. Ziel des Brückenkurses ist es, bei den Teilnehmenden die Entwicklungsumgebung, aktuell verwenden wir Visual Studio Code mit Microsofts C/C++-Erweiterung und die *GNU Compiler Collection*, zu installieren und ein erstes einfaches Programm zu programmieren. Der Hauptteil des Brückenkurses besteht aus praktischer Arbeit in Kleingruppen, die durch drei plenare Zusammenkünfte eingerahmt werden. Nach einer generellen Vorstellung und einer anekdotischen Einführung in die Programmierung im Plenum werden die Teilnehmenden in jeweils von einem/einer Tutor*in betreute Gruppen aufgeteilt, um den Rest des ersten Blocks in Kleingruppen die Installation der eingangs erwähnten Software abzuschließen. Der bessere Betreuungsschlüssel erlaubt hier eine bessere Konzentration auf die individuellen Probleme der einzelnen Studierenden. Nach einer Mittagspause gibt es im Plenum erneut die Möglichkeit, aufgetretene Probleme anzusprechen und kleinere Aufgaben zu motivieren, welche dann in den Kleingruppen am Nachmittag bearbeitet werden. Den Abschluss bildet wieder eine gemeinsame Plenumsveranstaltung.

Die Veranstaltung im Wintersemester besteht aus einer Vorlesung im Umfang von 3 SWS und Tutorien im Umfang von 2 SWS in Kleingruppen. Hier ist das Ziel, dass jedes Tutorium maximal 20 Teilnehmende umfasst, was bei 600 bis 700 Teilnehmenden ungefähr 30 angestrebten Tutorien entspricht. Die Vorlesung konzentriert sich auf die theoretischen Aspekte der Veranstaltung und ergänzt diese mit praktischen Anteilen und Programmierbeispielen. Die Tutorien vertiefen auf Basis von Übungsblättern die theoretischen Aspekte nochmals, beschäftigen sich aber auch ausführlich mit den Programmierinhalten der Veranstaltung. Um die aktive Teilnahme zu fördern, gibt es semesterbegleitend drei Zwischenprüfungen, die die Teilnehmenden online ablegen können. Diese Zwischenprüfungen werden in der abschießenden Klausur als Bonuspunkte mit bis zu 10 % der erreichbaren Gesamtpunkte beim Bestehen der Klausur angerechnet. Die Zwischenprüfungen finden im gleichen System wie die eigentliche Klausur statt und bieten den Studierenden damit die Option sich mit

dem Klausursystem vertraut zu machen. Für die persönliche Vorbereitung der Studierenden stellen wir zudem zwei bis drei Wochen vor der Klausur Altklausuren online, die von den Teilnehmenden gerechnet werden können und alle zwei Tage ausgewertet werden. Teilnehmende erhalten die Ergebnisse per E-Mail in einem Bericht. Im Rahmen der Veranstaltung unterstützen wir auch empirische Studie, die untersuchen, wo für Programmieranfänger die Hürden beim Erlernen des Programmieren liegen [ETK22]. Hierdurch unterstützen wir die Forschung im Bereich der Lehrdidaktik und könne unsere eigene Veranstaltung verbessern.

Aus den Erfahrungen der letzten Klausuren heraus fokussiert sich das Repetitorium verstärkt auf den Programmierteil der Veranstaltung. Auswertungen der letzten Jahre haben gezeigt, dass Aufgaben aus diesem Bereich im Schnitt schlechter abschneiden als Aufgaben aus den anderen Bereichen. Daher gibt es im Repetitorium keine einfache Wiederholung der Vorlesung, sondern eine Hörsaalübung im Umfang von 2 SWS und Tutorien in Kleingruppen im Umfang von 2 SWS. Im Rahmen des Repetitoriums gibt es eine fortlaufende Liste von Programmieraufgaben mit steigender Komplexität. Hierbei wird bewusst darauf verzichtet, diese den Veranstaltungswochen zuzuordnen, um allen Teilnehmenden ein individuelles Tempo zu ermöglichen. Gegen Ende des Semesters werden nochmals Altklausuren gerechnet, um nochmal die Themen der anderen Bereiche aufzufrischen. Zur Klausurvorbereitung gibt es diese nochmals im Klausursystem als Onlineklausur.

3 Lehrinhalte

Die Veranstaltung *Informatik für Ingenieure* hat vier wesentliche Komponenten im Bereich der zu vermittelnden Fachkompetenzen, die in Abbildung 1 dargestellt sind. Erstens sollen die Studierenden einen Überblick über die Grundlagen der Rechnerarchitektur erhalten. Zweitens sollen sie die (theoretischen) Grundlagen der Informatik erlernen und drittens einen Einblick die Themen der Softwaretechnik erhalten. Dies alles soll dazu dienen, die grundlegenden Programmierfähigkeiten, die sie erlernen sollen, mit einem theoretischen Verständnis zu versehen. Dabei ist zu differenzieren zwischen wissensbasierten Kompetenzen, die die Studierenden sich aneignen sollen und praktischen Kompetenzen. Die drei Basissäulen tragen hauptsächlich zum Erwerb der wissensbasierten Kompetenzen bei, während die grundlegenden Programmierfähigkeiten den Großteil der praktischen Kompetenzen ausmachen. Die folgenden Abschnitte beschreiben diese vier grundlegenden Komponenten genauer.

In der ersten Säule der Veranstaltung wird die Funktionsweise heutiger Rechner systematisch eingeführt. Zunächst werden Binärzahlen sowie Zahlen- und Zeichencodierungen eingeführt. Diese Basis wird genutzt, um mit Hilfe des Konzepts einer booleschen Algebra digitale Schaltungen einzuführen. Hierzu gehören einfache Schaltungen wie Multiplexer, Addierer und Multiplizierer. Diese Basisbausteine werden genutzt, um einen einfachen Prozessor und dessen zugehörigen Steuerungsautomaten herzuleiten. Daran kann schließlich die von-Neumann-Architektur erklärt werden. Hierbei wird das Konzept von deterministischen und nicht-deterministischen endlichen Automaten eingeführt, um den Steuerungsautomat auch

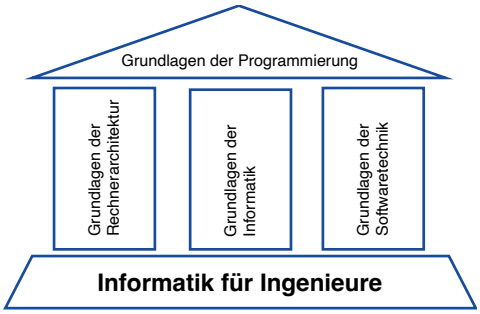

Abb. 1: Übersicht über die Modulinhalte der Veranstaltung *Informatik für Ingenieure*

auf konzeptioneller Ebene erläutern zu können. Die Grundlagen der Hardwareebene werden genutzt, um Maschinencode als unterste Ebene zur Programmierung von Rechnern einzuführen. Hierfür wird ein einfacher reduzierter Befehlssatz genutzt, um Speicherorganisation und indirekte Speicherzugriffe zu erklären.

In der zweiten Säule der Veranstaltung werden grundlegende (theoretische) Konzepte aus der Informatik eingeführt. Zu Beginn werden grundlegende Konzepte aus der Komplexitätstheorie zusammen mit der Landau-Notation vorgestellt, um den Studierenden ein Werkzeug zur Betrachtung von Algorithmen an die Hand zu geben. Diese werden anhand konkreter Beispiele in Bezug zur Rechnerarchitektur gesetzt. Anschließend werden dynamische Datenstrukturen anhand von Listen vorgestellt und deren Implementierungskonzepte präsentiert. Die einzelnen Operationen werden auf ihre Komplexität hin untersucht, um den Studierenden die Bedeutung der Wahl geeigneter Algorithmen aufzuzeigen. Auch das Sortieren von und Suchen in Feldern wird in diesem Kontext diskutiert.

In der letzten Säule wird auf die Grundlagen der objektorientierten Modellierung eingegangen. Hierbei werden die Unterschiede zwischen Daten und Operationen herausgearbeitet. Um den Studierenden ein Werkzeug zur Beschreibung objektorientierter Modellierung mitzugeben, werden in diesem Rahmen UML-Klassendiagramme eingeführt. Als wichtige Konzepte für die Modellierung werden Abstraktion, Modularisierung und Kapselung vorgestellt. Außerdem gibt es eine kurze Einführung in Softwareentwicklungsprozesse und in diesem Rahmen auch eine Einführung in die Konzepte des systematischen Testens wie Unit-Tests und Integrationstests.

Die Veranstaltung führt das Programmieren mit den Sprachen C und C++ ein, da diese die Möglichkeit bieten, zunächst mit imperativer Programmierung zu starten und die grundlegende Syntax und Semantik der Sprache C einzuführen. Außerdem sind diese Programmiersprachen im Bereich der Hardwaresteuerung noch immer weit verbreitet und für Studierende von ingenieurbezogenen Studiengängen mit hoher Wahrscheinlichkeit später relevant. Die Studierenden erhalten eine Einführung in den Umgang mit den Programmen Compiler, Debugger und Profiler, um Fehler selbständig analysieren und beheben zu

können. Zu Beginn der Veranstaltung werden zunächst die elementaren Datentypen, Arrays, Zeiger und die dynamische Speicherverwaltung eingeführt (auf die Grundlagen der zweiten Säule zurückgreifend). Später werden dann auf der Basis von C++ Klassen und die objektorientierten Konzepte aufgegriffen, was den Bogen zur dritten Säule schlägt.

In einer Klausur können Aufgaben im Allgemeinen nach drei Kriterien eingeordnet werden: Aufgabenform, Aufgabenthematik und Aufgabenniveau. Die Aufgabenthematik beschreibt hierbei den thematischen Bereich, aus dem die Aufgabe stammt. Im Rahmen der hier vorgestellten Veranstaltung werden Aufgaben thematisch zu den vier oben genannten Bereichen zugeordnet. Das Aufgabenniveau beschreibt das Anforderungsfeld der Aufgabe zwischen Reproduktion, Analyse und Transfer. In einer Klausur streben wir eine Verteilung von 60-30-10 bei einer Bestehensgrenze von 50% an. Die Aufgabenform beschreibt die Art der Aufgabenstellung. Durch die elektronische Form der Klausur und die automatisierte Auswertung bieten sich nur bedingt viele Möglichkeiten der Aufgabenstellung an. Im Rahmen dieser Veranstaltung wurden bisher Rechenaufgaben, Multiple-Choice Aufgaben, Drag-and-Drop Aufgaben und Programmieraufgaben verwendet. Hierfür verwenden wir das YAPS Prüfungssystem [BF21], ein erweiterbares digitales Prüfungssystem, das die genannten Aufgabentypen im Web-Browser ermöglicht. Für die Programmieraufgaben wird hierfür die verfügbare Serverinfrastruktur verwendet, um die Lösung der Studierenden zu bauen und auszuführen. Die Bearbeitungszeit der Klausur beträgt 90 Minuten. Die Klausur wird auf mehrere Durchführungen aufgeteilt, da die Kapazität, um die notwendige Anzahl an Klausuren gleichzeitig zu schreiben, nicht vorhanden ist. Das Klausurergebnis erhalten Teilnehmende in einem übersichtlichen Bericht per Email über den die Bewertung übersichtlich transparent gemacht wird. Eine Klausureinsicht vor Ort entfällt dementsprechend, außerdem wird die Schwelle für die Klausureinsicht drastisch gesenkt. Bei etwaigen Einsprüchen werden gegebenenfalls alle Klausurbewertungen angepasst, somit werden gleiche Bedingungen für alle Studierenden erreicht.

Durch den Schwerpunkt der Veranstaltung auf der Kompetenz Programmieren umfassen Programmieraufgaben stets 50% der erreichbaren Klausurpunkte (allerdings in unterschiedlichen Anforderungsniveaus). Die verbleibenden 50% der erreichbaren Klausurpunkte verteilen sich auf die anderen inhaltlichen Themen. Bei diesen Aufgaben werden unterschiedliche Aufgabentypen verwendet und es ist das Ziel, den Anteil der Multiple-Choice Aufgaben möglichst gering zu halten. Hierfür eignen sich Aufgaben, wie zum Beispiel das Umrechnen zwischen Zahlensystemen und das Ausführen eines Moore- oder Mealey-Automaten als Aufgaben mit dynamisch generierten Werten pro Studierendem. Mit Drag-und-Drop-Aufgaben stellen wir Aufgaben zu Besuchsreihenfolgen in Graphen oder lassen aus einer beschränkten Anzahl an vorgegebenen Code-Snippets in der Vorlesung vorgestellte Algorithmen zusammenbauen. Als letztes gibt es in dieser Kategorie noch Multiple-Choice Aufgaben, um Fragen im reproduktiven Aufgabenbereich zu stellen. Der Anteil der reinen Multiple-Choice Fragen ist pro Klausur auf circa 25% der erreichbaren Gesamtpunkte beschränkt. In der Klausur gibt es zwei Programmieraufgaben mit unterschiedlichem Schwierigkeitsgrad, die aktuell zu gleichem Teil die Gesamtpunkte des Programmierteils ergeben. Die erste

Programmieraufgabe beschränkt sich auf imperative Programmierkonzepte, also den Umfang der Sprache C einschließlich dynamischer Speicherverwaltung. In dieser Aufgabe sind mehrere Funktionsrümpfe vorgegeben, die die Studierenden zu füllen haben, aber die einen sehr geringen Anteil an selbst zu entwickelnder Algorithmik enthalten. Hier müssen zum Beispiel Arrays auf dem Heap angelegt und initialisiert, Werte in Arrays quadriert oder die Summe aller Elemente eines Arrays berechnet werden. Für eine Funktion gibt es in dieser Aufgabe einen vorgegebenen natürlichsprachigen Pseudo-Code, den die Studierenden in C-Code übersetzen müssen. Daher fallen die Funktionen dieser Aufgabe in die Reproduktions- bzw. Analysekategorie. In der zweiten Programmieraufgabe nutzen wir zusätzlich noch Klassen und die STL von C++. Hierbei erhalten die Studierenden aber eine Dokumentation aller eingesetzten STL-Typen. Diese Aufgabe fällt in die Analyse- und Transferkategorie, da sie deutlich mehr selbstständiges Denken erfordert, um diese erfolgreich lösen zu können.

4 Statistische Auswertung der Klausuren

Im Folgenden wird die statistische Auswertung der Klausuren beschrieben und ihre allgemein anwendbare Struktur erläutert, sowie die Interpretation ihrer Visualisierungen dargelegt. Bei der Auswertung einer Klausur sind prinzipiell drei unterschiedliche Bereiche von Interesse: (1) Wie ist ein einzelner Durchlauf einer einzelnen Klausur zu bewerten? (2) Wie ist ein gesamter Durchlauf eines Semesters mit potentiell mehreren Klausuren und Wiederholungen einzelner Klausuren zu bewerten? (3) Wie ist die Entwicklung der Klausurergebnisse im zeitlichen Kontext zu bewerten? Die einzelnen Fragen werden in den jeweiligen Abschnitten diskutiert.

4.1 Auswertung eines einzelnen Klausurdurchlaufs

Hier wird das Ergebnis einer zeitgleich geschriebenen Klausur zwischen allen Teilnehmenden analysiert.

Ein schlichter Ausgangspunkt für die Analyse ist ein Histogramm [Wa06] oder Boxplot [Tu77] der Punkteverteilung. Ob die Punkte in diesem Fall relativ oder absolut dargestellt werden, ist nicht von Bedeutung. Ein Boxplot visualisiert gleichzeitig den Median (das Lagemaß), Minimum und Maximum (Spannweite) und unteres/oberes Quartil (Inter-Quartils-Abstand). Es ist möglich, auch noch den Mittelwert mit einzuzeichnen. Die Analyse kann für die einzelnen Aufgaben wiederholt werden, sodass aufgabenspezifische Effekte in aufgabenspezifischen Histogrammen und Boxplots identifiziert werden können.

Um die Güte bzw. Eignung einzelner Aufgaben in Bezug auf das Gesamtklausurergebnis zu beschreiben, können die einzelnen Klausurergebnisse entsprechend ihrer Gesamtpunktzahl gruppiert werden und ihr Resultat innerhalb einer spezifischen Aufgabe als Boxplot

repräsentiert werden. Eine Aufgabe, die exakt die Ergebnisse der Gesamtklausur repräsentiert, erhielte so eine Boxplotkurve, die der kumulativen Normalverteilung entspricht. Aufgaben, die durchschnittlich schlechtere Ergebnisse lieferten, haben eine nach rechts verschobene Kurve, Aufgaben, die durchschnittlich bessere Ergebnisse liefern, haben sie eine nach links verschobene Kurve. Abbildung 2 zeigt zwei Aufgaben in der erläuterten Darstellung.

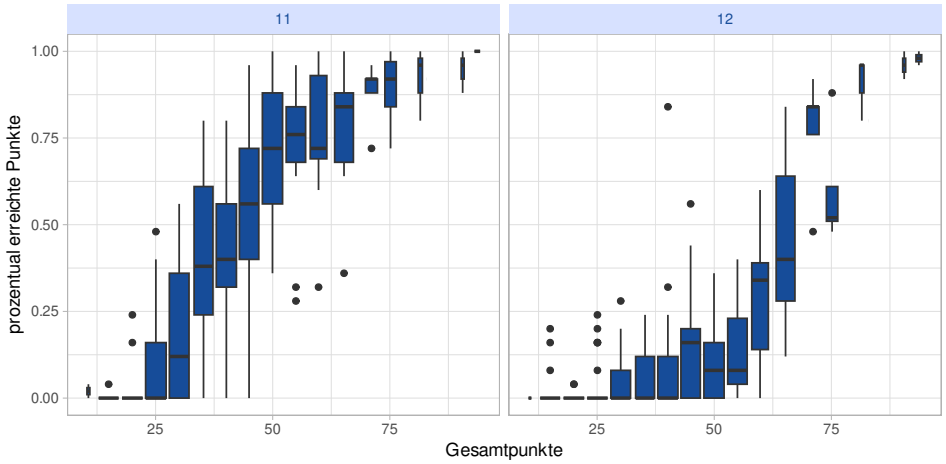

Abb. 2: Zwei Beispielprogrammieraufgaben als Box-Plots, die die Gesamtpunkte in der Klausur gegen die Punkte der jeweiligen Aufgabe abbildet. Die erste Aufgabe stammt aus dem Reproduktions- und Analysekategorie. Die zweite Aufgabe ist die schwierigere Programmieraufgabe aus der Analyse- und Transferkategorie. Somit ensprechen die Darstellung im wesentlichen den Erwartungen.

Die Trennschärfe [Fi04] einer Aufgabe bestimmt die Fähigkeit dieser Aufgabe, das Gesamtklausurergebnis vorherzusagen. Eine hohe Trennschärfe zeigt an, dass das Ergebnis in dieser Aufgabe das Gesamtergebnis gut vorhersagt, eine niedrige Trennschärfe sagt, dass diese Aufgabe kaum Einfluss auf das Gesamtergebnis hat. Eine negative Trennschärfe bedeutet, dass gute Ergebnisse bei dieser Aufgabe auf ein schlechtes Gesamtergebniss hindeutet, was eigentlich eher unerwartet ist.

Zunächst werden die Resultate für themen- und niveaugleiche Aufgaben addiert, falls mehrere vorhanden sind. Diese kumulativen Ergebnisse werden dann für die entsprechenden Themen- und Niveaustufen als Boxplots (oder Histogramme) dargestellt. Mit dieser Grafik kann zum Einen untersucht werden, ob unterschiedliche Themen unterschiedlich gut verstanden wurden, und zum Anderen, ob die Einordnung der Aufgaben hinsichtlich ihres Niveaus mit den Resultaten der Studierenden in der Klausur übereinstimmt.

In dieser Veranstaltung werden innerhalb des Semesters Testate geschrieben, in denen es den Studierenden möglich ist, Bonuspunkte für ihre finale Modulnote zu sammeln. Ein Punktdiagramm zwischen erreichten Bonuspunkten und erreichten Punkten in der Klausur kann Aufschluss darüber geben, ob eine (erfolgreiche) Teilnahme (oder Nicht-Teilnahme) an

den Testaten, die nicht verpflichtend sind, einen Einfluss auf die Leistung in der Klausur hat. Damit wird die Korrelation zwischen erzielter Bonuspunkte und Klausurpunkte dargestellt.

4.2 Auswertung eines Gesamtdurchlaufs

Für den Gesamtüberblick über alle aktiven Läufe wiederholen wir alle obigen Diagramme zur Gesamtlage (also Boxplot und Histogramme) mit den Gesamtdaten. Um Unterschiede zu visualisieren, werden die unterschiedlichen Klausurdurchläufe mit unterschiedlichen Farben symbolisiert.

Um einen Vergleich der Resultate von Klausurwiederholungen sichtbar zu machen vergleichen wir die Verteilungen – entweder nicht-parametrisch (wenn die Voraussetzungen nicht erfüllt sind) oder parametrisch. Dieser kann sowohl auf Aufgabenebene als auch auf Klausurebene durchgeführt werden.

Für den Vergleich zwischen unterschiedlichen Klausurvarianten verwenden wir einen nicht-parametrische Test. Um diese auf Aufgabenebene durchführen zu können, ist allerdings die Klassifizierung anhand der oben genannten Kriterien wichtig, da sich die Aufgaben in den unterschiedlichen Klausurvarianten unterscheiden.

4.3 Auswertung der Entwicklung der Veranstaltung

Eine einfache Möglichkeit, die Entwicklung der Klausurergebnisse (bei gleichen Verteilungen der Aufgaben in Bezug auf die oben genannten Kriterien) über die Zeit zu vergleichen, sind die Boxplots über die Gesamtergebnisse eines Durchlaufs für die einzelnen Jahre darzustellen. Anhand der Visualisierung kann dann beobachtet werden, ob die Streuung oder Lage sich über die Zeit verändert.

4.4 Implementierung

Der statistische Auswertung der Klausur liegt eine R-Implementierung zugrunde. R ist eine Programmiersprache, die sich auf statistische Analyse von Daten fokussiert [Da15]. Die Auswertung ist unabhängig von der Veranstaltung *Informatik für Ingenieure* und kann daher für die Auswertung beliebiger Klausuren verwendet werden. Die Daten müssen in einem einfachen CSV-Format vorliegen, sodass das Skript automatisch die beschriebenen Informationen berechnen kann. Dieses Skript ist öffentlich zugänglich [BF].

5 Herausforderungen

Lehre ist stets mit Herausforderungen verbunden. Interdisziplinäre Lehre der unterstützenden Form ist allerdings neben den *normalen* Herausforderungen noch mit zusätzlichen Herausforderungen konfrontiert.

Eine Herausforderung besteht in der Konzeption der Veranstaltung, die durch die Modulstruktur vorgegeben ist: Sowohl die theoretischen Grundlagen als auch Programmierfähigkeiten innerhalb eines Semesters zu vermitteln, ist didaktisch fordernd. Obwohl die theoretischen Grundlagen aus Sicht eines erfahrenen Programmierers wichtig sind für die Entwicklung ausreichender Programmierkompetenzen, ist dies für Programmieranfänger oft nicht in Gänze nachzuvollziehen. So entsteht für manche Studierende eine Disparität zwischen den Lerninhalten, die ihren Lernerfolg erschweren.

Der Zeitpunkt der Veranstaltung im Studienverlaufsplan (erstes Semester) ist eine weitere Herausforderung. Zu diesem Zeitpunkt wirken die Lerninhalte für Studierende noch nicht mit ihren späteren beruflichen Aufgaben motiviert. Sie beginnen erst, sich mit ihrer Fachrichtung auseinander zu setzen und sehen noch nicht die Aufgaben, in denen grundlegende informatische Kenntnisse von Vorteil sind und Programmierfähigkeiten notwendig. Dies ist sowohl problematisch im Hinblick auf die Motivation der Studierenden als auch auf die Anwerbung von studentischen Tutor*innen. Der Bedarf an studentischen Tutor*innen ist in der Veranstaltung hoch und es ist nicht immer möglich, die benötigte Menge an Tutor*innen zu finden. Da die Technische Universität Hamburg auch den Studiengang *Computer Science* anbietet, suchen wir auch hier aktiv nach möglichen Tutor*innen.

Ein weiterer Aspekt ist die mangelnde Akzeptanz der Wichtigkeit der vollständigen Bearbeitung von Übungsaufgaben. Wie viele Fächer reicht auch (insbesondere im Programmieren) in dieser Veranstaltung ein rein theoretisches Verständnis nicht aus. Um eine ausreichende Kompetenzstufe im Bereich des Programmierens zu erlangen, ist konstantes und ausdauerndes Üben an unterschiedlichen Programmieraufgaben von Nöten. Daher beinhaltet diese Veranstaltung umfangreiches Übungsmaterial und legt Wert auf Präsenzübungen in den Tutorien. Nichtsdestotrotz führt die Tatsache, dass die Bearbeitung der Übungszettel nicht verpflichtend ist, sondern nur als Angebot vorliegt, zu einer geringeren Anzahl an Studierenden, die diese vollständig bearbeiten und auch bei Herausforderungen sich diesen stellen. Ein größerer Anteil an Studierenden sieht die Übungsaufgaben als optional und erreicht so nicht den nötigen Erfahrungsschatz, der für einen erfolgreichen Erwerb von Kompetenzen im Bereich der Programmierung notwendig wäre.

Es ist durchgängig ein Rückgang an bereits vorhandenen Kompetenzen bei den Studierenden zu beobachten, sowie eine hohe Streuung, was ihre Vorkenntnisse betrifft. Für einen größer werden Anteil an Studierenden ist beispielsweise die Formulierung der Übungsaufgaben oder partiell Klausuraufgaben auf einem Niveau, das nicht ihrer Leseverständniskompetenz entspricht. Ähnliches gilt für grundlegende mathematische Kompetenzen, wie Rechenregeln oder logisches Denken. Gleichzeitig gibt es jedoch einen kleinen Anteil an Studierenden,

die enorme Vorkenntnisse und Kompetenzen mitbringen, die durch eine Unterforderung im Kurs demotiviert werden. Im universitäten Kontext ist jedoch eine binnendifferenzierte Leistungserbringung bisher noch nicht vorgesehen, sowie eine binnendifferenzierte Lehrstruktur (wie sie in Schulen umgesetzt wird) in ihrer Umsetzung sehr herausfordernd. In Bezug auf Programmierkenntnisse wird dies partiell in den Übungsgruppen umgesetzt. So gibt es explizit Übungsgruppen für Studierende mit Vorkenntnissen und Übungsgruppen für Studierende ohne Vorkenntnisse. Ein Nachteil dieser Lösung ist allerdings die dadurch mangelnde Durchmischung unterschiedlicher Gruppen, ein Lernen durch Lehren (in Gruppenstrukturen bspw.) ist so nicht mehr möglich.

Eine weitere, eher unerwartete Herausforderung ist das fehlende Wissen im grundlegenden Umgang mit Computern. Grundlegend ist hier die Beobachtung, dass moderne Betriebssysteme mittels App-Stores Installationsvorgänge weitgehend vereinfachen und die entsprechenden Apps den tatsächlichen Speicherort von Daten nicht mehr offensichtlich für Nutzende darstellen. Während die Studierenden sich mit der Nutzung von Standardanwendungen sehr gut auskennen, wird es bei Programmen, die manuell installiert werden müssen, wie einem Compiler deutlich schwieriger. Vielen Studierenden ist hierbei zum Beispiel unklar, wo Programme installiert werden, wie installierte Dateien auf dem System gefunden werden können. Ebenso ist der Umgang mit Archiven, die entpackt werden müssen, oder das Auffinden eigener gespeicherter Daten eine Herausforderung.

6 Ausblick

Konstant progressierende Anforderung an die Veranstaltung sowie sich diesen Änderungen anpassende Lehrkonzepte seitens der Veranstalter führen zu einem steten Wandel in der Veranstaltungsform. Eine standardisierte statistische Auswertung soll bei der Evaluierung dieser Änderungen unterstützen und wird im Rahmen dieser Publikation auch für andere Veranstaltungen zur Verfügung gestellt.

Die Veranstaltung lehrt die Grundlagen der Rechnerarchitektur, (theoretischen) Informatik und Softwaretechnik, um den Studierenden das Grundgerüst für den Erwerb von Programmierkompetenzen zu geben, der den Schwerpunkt der Veranstaltung ausmacht. Die richtige Balance zwischen beiden Aspekten zu finden, ist eine Herausforderung, der sich entsprechend der Rückmeldung der Studierenden von Jahr zu Jahr durch neue Konzepte in der zeitlichen Abfolge gestellt wird.

Neben diesen inhaltlichen Herausforderungen sind jedoch auch Unterschiede in der Vorbildung und ein Rückgang im grundlegenden technischen (rechnerbasierten) Verständnis eine Herausforderung. Durch den großen Anteil der Zeit, der in Tutorien durch Vorarbeiten wie Installationsprozesse gebunden wird, ist eine geplante Änderung für den nächsten Durchlauf die Durchführung einer Lerneinheit diese grundlegenden rechnerbasierten Kenntnisse (über z.B. Betriebssysteme) aufgreift.

Unserer Meinung nach ist eine ganzheitliche, grundlegende Bildung im informatischen Sinne für die Ingenieur*innen im 21. Jahrhundert von essentieller Bedeutung. Daher die Weiterentwicklung unseres Lehr- und Lernkonzeptes von dem Grundgedanken geprägt, die thematische Breite in jedem Fall beizubehalten, jedoch durch Zusatzveranstaltungen oder binnendifferenzierte Übungsaufgaben zu Beginn der Veranstaltung die unterschiedlichen Vorbildungsstände der Studierenden aufzufangen, um ihnen einen bildungsgerechten Start in ihr Studium zu ermöglichen sowie eine erfolgreiche berufliche Laufbahn.

Literatur

[BF] Berger, B. J.; Fey, G.: Auswertungsskript für Klasuren, Stand: 31.08.2023.

[BF21] Bahnsen, F. H.; Fey, G.: YAPS - Your Open Examination System for Activating and emPowering Students. In: 2021 16th International Conference on Computer Science & Education (ICCSE). IEEE, 2021, URL: https://doi.org/10.1109/iccse51940.2021.9569549.

[Da15] Daroczi, G.: Mastering Data Analysis with R. Packt Publishing, 2015, ISBN: 1783982020.

[ETK22] Eckert, D.; Timmermann, D.; Kautz, C.: Student Misconceptions about Loops in Introductory Programming Courses and the Influence of Representations. In: 2022 IEEE Frontiers in Education Conference (FIE). IEEE, 2022, URL: https://doi.org/10.1109/fie56618.2022.9962545.

[Fi04] Fisseni, H. J.: Lehrbuch der psychologischen Diagnostik. Hogrefe Verlag, Göttingen, Germany, 2004.

[Mo22] Mohl, E.: Veränderte Kompetenzbereiche durch Industrie 4.0 in der Ausbildung von Techniker∗innen an Höheren Technischen Lehranstalten in Österreich. Springer Fachmedien Wiesbaden, 2022.

[Te] Technische Universität Hamburg: Homepage Brückenkurs, Stand: 31.08.2023.

[Te21] Technische Universität Hamburg: Modulhandbuch – Bachelor of Science (B.Sc.) – Maschinenbau, Techn. Ber., Technische Universität Hamburg, 2021, URL: https://studienplaene.tuhh.de/po/MB/mhb_MBBS_kh_w21_von_20210918_v_0_de.pdf.

[Tu77] Tukey, J. W.: Exploratory Data Analysis. Pearson, Upper Saddle River, NJ, 1977.

[Wa06] Wasserman, L.: All of Nonparametric Statistics. Springer New York, 2006.