# Informatik ist nicht nur Programmieren – aber ohne Programmieren ist nichts Informatik

Oliver Radfelder, Karin Vosseberg, Ulrike Erb, Henrik Lipskoch
{oradfelder,kvosseberg,uerb,hlipskoch}@hs-bremerhaven.de
Hochschule Bremerhaven

## Zusammenfassung

Ziel der Studieneingangsphase in den Bachelorstudiengängen Informatik und Wirtschaftsinformatik der Hochschule Bremerhaven ist, die Studierenden in eine Fachkultur einzuführen, in der Informatik nicht nur Programmieren ist - aber ohne Programmieren nichts Informatik ist. In kleinen Schritten werden die Studierenden an ihr grundlegendes Handwerkszeug und eine einfache Linux-basierte Infrastruktur für die Automatisierung von Prozessen herangeführt sowie das Arbeiten in Teams in einem ersten Projekt erprobt. Mit regelmäßigen kleinen Übungsaufgaben werden die verschiedenen Grundlagenfächer miteinander verzahnt. Über den Projektkontext wird ein Anwendungsbezug hergestellt. In dem vorliegenden Beitrag wird das Konzept der überarbeiteten Studieneingangsphase vorgestellt und anhand von kleinen Beispielen die Verzahnung von Modulen demonstriert. Im weiteren wird ein Beispiel für eine Lerneinheit aus den Workshops der Studieneingangsphase beschrieben.

## Abstract

The introduction phase of the bachelor programmes Informatics and Business Informatics at Bremerhaven University of Applied Sciences aims at familiarising students with a faculty culture which understands informatics not only as programming, but follows also the idea that without programming nothing is informatics. In small steps, students learn to handle their basic tools. They are introduced to a simple Linux-based infrastructure for the automation of processes, and experience collaborating in teams while working on a first project. Through continuous small exercises they get an understanding of basic topics of the first semester modules and finally apply their knowledge in the project context. This article presents the concept of the introduction phase, describes the interlocking of modules based on small examples, and shows an example of a workshop unit of the introduction phase.

## Einführung

Die zunehmende Heterogenität der Studierenden im ersten Semester erfordert ein Umdenken in der Gestaltung der Studieneingangsphase. Nicht zuletzt gefördert durch den Qualitätspakt Lehre wurden in den letzen Jahren an vielen Hochschulen Projekte initiiert, die Voraussetzungen an die Fachkompetenzen, die Studierfähigkeit aber auch die soziale Integration der Studienanfänger*innen in den Blick nehmen (Key u. Hill, 2018). Die verschiedenen Projekte setzen auf sehr unterschiedlichen Ebenen an. Allen Projekten gemeinsam ist aber eine enge Begleitung der Studierenden in der Übergangsphase zum Studium.

Mit der letzten Reakkreditierung wurde an der Hochschule Bremerhaven in den Bachelorstudiengängen Informatik und Wirtschaftsinformatik 2013 eine fachspezifische Studieneingangsphase (STEP) im Curriculum verankert. Ziel der Studieneingangsphase ist, das Berufsbild der Informatik und Wirtschaftsinformatik bei den Studierenden zu schärfen und das Zusammenspiel der Grundlagenveranstaltungen als Basis für die Informatik- und Wirtschaftsinformatikausbildung zu verdeutlichen. Die Idee, das Studium mit einem Projekt zu starten, gibt dabei den notwendigen Kontext für die vielfältigen Aufgaben in der Gestaltung und dem Einsatz von Softwaresystemen und lässt genügend Raum für Diskussionen über den vorhandenen Gestaltungsspielraum (Vosseberg, 2015). Mit den Projekten können die Studierenden erste Erfahrungen des eigenständigen, forschenden Lernens in Teams sammeln in einem eng begleiteten Lernkontext. Die Projekte fördern insbesondere die soziale Integration und den Austausch der Studierenden mit ihren vielfältigen Kompetenzen, die sie mit in ihr Studium einbringen. Um an dem Erfahrungshintergrund der Studierenden anzuknüpfen wurden in den ersten STEP-Jahren Analyseprojekte initiiert, da nicht davon ausgegangen werden kann, dass die Studierenden bereits umfangreiche Programmiererfahrungen mitbringen (Vosseberg u. a., 2015). Ähnliche Ansätze mit Projekten zum Studienstart werden auch an anderen Hochschulen verfolgt insbesondere mit dem

Fokus auf Studierbarkeit und das Fördern von sozialen Kompetenzen (vgl. (Dennert-Möller u. Garmann, 2016)).

Die Evaluation der ersten vier Durchläufe der Studieneingangsphase haben ergeben, dass die Startprojekte und die enge Begleitung der Projektteams durch studentische Tutor*innen und einem Coaching von Lehrenden die soziale Integration in beiden Studiengängen sehr gefördert haben. Die Verzahnung zu den Grundlagenfächern und die Angleichung von grundlegenden Fachkompetenzen war bislang jedoch nur bedingt gelungen. Gerade die Module Mathematik und Programmierung wurden nach wie vor als getrennt von der Studieneingangsphase wahrgenommen und die kontinuierliche Arbeitsweise aus den STEP-Projekten nicht übertragen. Ähnliche Effekte wie sie in der Auswertung der Umfrage zur Programmierausbildung von Axel Schmolitzky (Schmolitzky, 2017) beschrieben wurden, waren auch nach Einführung der Studieneingangsphase weiter zu beobachten. Nach wie vor sind die beiden Module Mathematik und Programmierung angstbesetzt, und die Modulprüfungen werden von einer überwiegenden Anzahl von Studierenden in spätere Semester geschoben.

Diese Erfahrungen haben uns dazu bewogen, das Lernsetting der Studieneingangsphase mit dem Wintersemester 2017/18 zu verändern. Gemäß dem Motto ”Informatik ist nicht nur Programmierung aber ohne Programmierung ist nichts Informatik” erarbeiten sich die Studierenden in einer Workshop-ähnlichen Lernumgebung erste Fertigkeiten in der Automatisierung von Abläufen und schleifen diese mit kleinen Übungen regelmäßig ein. Unterstützt werden die ca. 100 Studierenden durch Lehrende, die als Coaches zur Seite stehen, sowie durch studentische Tutorinnen und Tutoren. Das Einüben der Fertigkeiten bereitet sie auf die gestellte Projektaufgabe vor, die in dem überarbeiteten Konzept der Studieneingangsphase einen wesentlichen Anteil an Programmierung enthält. Im Folgenden wird das Lernsetting der Studieneingangsphase und insbesondere die Verzahnung mit den Modulen Mathematik und Software Engineering (SWE I) näher beschrieben.

## Studieneingangsphase - Rahmenbedingungen

Die Studieneingangsphase ist an der Hochschule Bremerhaven durch den Modul Einführung in die Informatik bzw. Einführung in die Wirtschaftsinformatik im Curriculum der beiden Bachelorstudiengänge Informatik und Wirtschaftsinformatik verankert. Damit sind sowohl auf studentischer Seite ein Workload von 5 CP vorgesehen, als auch auf Seiten der Lehrenden 8 SWS Lehrleistung eingeplant. Im Rahmen eines Teamteachings betreuen 3-4 Lehrende die Studieneingangsphase, um damit auch die Verzahnung zu anderen Modulen im ersten Semester realisieren zu können. Zusätzlich wird den Studierenden über die enge Verzahnung mit den Aufgaben im Modul Software Engineering I weitere Zeit für das Einüben der für die STEP-Projekte notwendigen grundlegenden Fertigkeiten eingeräumt.

Die Projektorientierung ist nach wie vor eine zentrale Eigenschaft der Studieneingangsphase. Am ersten Studientag werden die Informatik- und Wirtschaftsinformatikstudierenden in 12 gemischte Teams mit jeweils 8- 10 Studierenden eingeteilt. Jedes Team wird durch eine*n Tutor*in über das ganze erste Semester begleitet und erhält einen Coach als erste Ansprechperson an die Seite. Zur Zeit betreut jeder der drei Coaches vier Teams, während die vier studentischen Tutor*innen je drei Teams betreuen und für diese ein STEP-Tutorium anbieten. Zusätzlich betreuen ältere Studierende die technische Infrastruktur der STEP-Teams (siehe unten). Außerdem wird für jedes Team zur Teamorganisation eine Gruppe im Rahmen des Lernmanagementsystems Ilias eingerichtet.

In den ersten Wochen werden neben Einzelaufgaben auch Teamaufgaben gestellt, um sukzessiv auf die Projektaufgabe aber auch auf das gesamte Studium vorzubereiten. Um eine Workshop-ähnliche Lernsituation zu erzeugen werden in einem großen Veranstaltungsraum 12 Gruppentische aufgebaut, an denen die Teams gemeinsam ihre Aufgaben bearbeiten und sich gegenseitig in den Einzelaufgaben unterstützen können. Der Ablauf der 3-4-stündigen Workshops ist geprägt durch einem Wechsel zwischen Inputs der Coaches, z.B. in Form einer kurzen Einführung in ein Thema oder von kleinen Live-Coding-Einheiten, dem selbständigen Üben von Fertigkeiten und der gemeinsamen Bearbeitung der Projektaufgabe im Team. Die Sitzungen sind geprägt durch eine sehr arbeitsintensive Atmosphäre, in der alle Studierenden konzentriert mitarbeiten. Daneben werden sie mit kleinen Wochenaufgaben angehalten, während der Woche regelmäßig - am besten täglich - sich mit der technischen Infrastruktur auseinanderzusetzen, um die einfachen Fertigkeiten in der Automatisierung einzuschleifen.

Im Rahmen einer Portfolio-Prüfung für den Modul Einführung in die Informatik bzw. Einführung in die Wirtschaftsinformatik müssen die Studierenden die gestellten Aufgaben bearbeiten und jede Woche einen Eintrag in ihrem Reflektionsblog in das Lernmanagementsystem einstellen. Am Ende des Semesters stellen die Projektteams ihre Projektergebnisse am Tag der Informatik mit einem Plakat vor. Über die regelmäßige Bearbeitung der gestellten Einzel- und Teamaufgaben insbesondere aber über die wöchentlichen Blogeinträge erhalten die Coaches einen guten Einblick über den Lernfortschritt der Studierenden und können bei Problemen sofort gegensteuern. Nach anfänglichen Schwierigkeiten werden die Blogeinträge regelmäßig geführt und sind damit eine wertvolle Informationsquelle für die Evaluation der Studieneingangsphase.

Um eine organisatorische Verzahnung zwischen den verschiedenen Modulen im ersten Semester zu unter-

stützen, werden die Teams als Lerngruppen in allen Modulen genutzt und in der Gruppenaufteilung im Stundenplan berücksichtigt. Zusätzlich übernehmen Lehrende aus dem ersten Semester eine Gruppe (3 Teams) im Rahmen des Programmierlabors, damit die Verzahnung zum Programmierenlernen sichtbarer wird. Die Programmierlabore werden mit zusätzlichen Tutor*innen unterstützt. Somit haben wir eine sehr enge Begleitung der Erstsemesterstudierenden und sie lernen sehr frühzeitig viele Kolleg*innen aus dem Informatikbereich mit ihrer eigenen Vielfalt kennen.

## Inhalte der Studieneingangsphase

Im Zentrum der Studieneingangsphase stehen die Projekte, die einen Anwendungskontext bieten und zum Erlernen von grundlegendem Handwerkzeug der Informatik motivieren, das die Studierenden für die Realisierung der Projekte benötigen. Dieses Handwerkszeug wird sie auch durch ihr gesamtes Studium begleiten. Um die Studieneingangsphase abzurunden und einen praktischen Einblick in Berufsbilder der Informatik und Wirtschaftsinformatik zu erhalten, besuchen alle Teams an einem Tag jeweils ein Unternehmen aus der Region.

Für den Unternehmensbesuch werden 12 ganz unterschiedliche Unternehmen ausgewählt, von klassischen Softwareentwicklungshäusern, spezialisierten Unternehmen beispielsweise aus der Logistik- oder Lebensmittelbranche bis hin zu Beratungsunternehmen oder IT-Abteilungen von Konzernen. Die Teams müssen für den Unternehmensbesuch Fragen vorbereiten, die insbesondere an ihren bisherigen Erfahrungen mit der neu erlernten Infrastruktur und den Arbeitsweisen anknüpfen. Nach dem Unternehmensbesuch werden die Erfahrungen in Form eines World-Cafes mit allen Studierenden geteilt. Damit wird die Vielfalt der beruflichen Möglichkeiten sichtbar. Außerdem wird diskutiert, welche Grundlagen die Studierenden in ihrem Studium brauchen, um in Zukunft solche oder ähnliche Jobs bewältigen zu können.

### Projekte 2017/2018 und 2018/2019

Die Projektaufgabe der Studieneingangsphase 2017/2018 bestand darin, Modellboote, die jeweils mit einem Raspberry Pi ausgestattet und steuerbar sind, über einen Web-Browser per WLAN zu lenken und zu beschleunigen bzw. zu verlangsamen. Zur Vorbereitung dieser Projekte hatten ältere Studierende die Modellboote nach einer Konstruktionszeichnung des Deutschen Schifffahrtsmuseums Bremerhaven (DSM) für den Druck per 3D-Drucker aufbereitet. Für jedes der 12 STEP-Teams haben sie ein Boot gedruckt und mit Motor, Steuerruder und Raspberry Pi versehen (siehe Abbildung 1).

Eine Besonderheit bei dieser Aufgabe bestand darin, dass der Original-Schlepper, der als Vorlage für die Modellboote diente, seinen Liegeplatz direkt vor

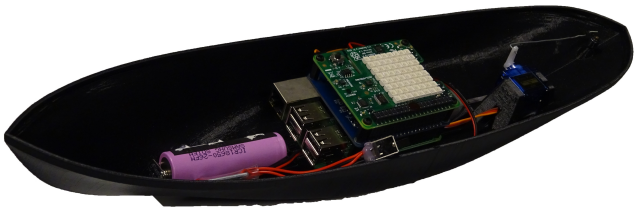


Abbildung 1: Modellboot mit Raspberry Pi

dem STEP-Veranstaltungsraum, dem ehemaligen Fährhaus, hat und von dort aus oft zu sehen war (siehe Abbildung 2).


Abbildung 2: Der Schlepper Tide

Als Hochschule am Meer haben wir auch in der Studieneingangsphase 2018/2019 ein maritimes Thema gewählt. Dieses Mal wurde ein Raspberry Pi mit einem DVB-T-Empfänger verbunden und oben auf dem Fährhaus stationiert. Über diesen Rechner werden permanent AIS-Signale aller Schiffe in einem Umkreis von bis zu 50 km empfangen, dekodiert und in einem bestimmten Format als Datenstrom in unserer Informatik-Infrastruktur zur Verfügung gestellt. Das AIS ist das Automatische Schiffsidentifizierungssystem (engl. Automatic Identification System) und dient dem Austausch von Navigations- und Schiffsdaten zur Verbesserung der Sicherheit im Schiffsverkehr (WSV, 2018).

Die Projektaufgabe besteht darin, aus diesem Datenstrom Angaben zu ausgewählten Schiffen zu filtern und auf einer Website per HTML und SVG zu veranschaulichen. Zum Beispiel können die Positionen und Bewegungen von Schiffen relativ zum Fährhaus und auch auf einer Karte angezeigt werden. Dazu können Informationen wie Schiffsname oder Zielort gegeben werden. Die Entscheidung über die Darstellungsart der Daten und die Anzeige zusätzlicher Informationen bleibt den STEP-Teams überlassen. Daraus ergeben sich Webseiten, bei denen eher Informationen über die Schiffe im Vordergrund stehen, und solche, die die

Möglichkeiten dynamischer Webseiten in der gegebenen Infrastruktur ausloten.

Anhand der beschriebenen Aufgabenstellungen werden grundlegende Kenntnisse von Bash-Shell-Skripten und HTML eingeübt sowie ein Verständnis für webbasierende Client-Server-Architekturen vermittelt.

### Handwerkszeug der Informatik und Wirtschaftsinformatik

An der Hochschule Bremerhaven wurde innerhalb der vergangenen zwei Jahre wieder eine klassische Linux-basierte Infrastruktur aufgebaut, wie sie in den 80er und 90er Jahren an vielen Universitäten und Hochschulen in der Informatik zum Standard gehörte. Damit wird der aktuellen Entwicklung Rechnung getragen, in der Webtechnolgien und insgesamt Server-orientierte Anwendungen einen massiven Aufschwung erleben.

So besteht die gemeinhin als *Cloud* bezeichnete technolgische Basis moderner Anwendungen letztlich aus einer Menge an Servern mit Rechen- und Speicherkapazität. Anwendungen bestehen dort aus Programmen, die typischerweise unter Linux laufen. Selbst in der Microsoft-eigenen Azure-Cloud laufen zunehmend Anwendungen unter Linux:

> *Today, Scott Guthrie, Microsoft's executive vice president of the cloud and enterprise group, said in an interview, "it's about half now, but it varies on the day because a lot of these workloads are elastic, but sometimes slightly over half of Azure VMs are Linux."*
>
> (Steven J. Vaughan-Nichols)

Zudem ist das *InternetOfThings* und große Teile dessen, was unter *Industrie 4.0* verstanden wird, auf unixoiden System – speziell Linux – aufgebaut.

Folglich müssen heute Studierende wieder darauf vorbereitet werden, sich in einer solchen Umgebung sicher bewegen zu können.

Unsere Umgebung besteht aus mittlerweile mehreren Arbeitsservern, auf denen sich die Studierenden innerhalb der Hochschule ebenso wie von außerhalb per *ssh* einloggen können. Sie finden dort ein Home-Verzeichnis für Ihre Arbeiten vor und lernen von Beginn an, mit dem Editor *vim* Ihre Aufgaben zu erledigen. Zudem haben sie dort ein Webverzeichnis, in dem sie ihre persönliche Webseite mit HTML und CSS gestalten können und sollen. Die Infrastruktur bietet jedem Studierenden und jedem Lehrenden außerdem Zugang zu einer eigenen Datenbank, zu PHP, Java und einem Git-Server.

Da wir neben Java und der Bash im ersten Semester nicht noch eine weitere Programmiersprache mit PHP einführen wollten, trotzdem aber mit dynamisch erzeugen Webseiten bereits Automatisierung in das Zentrum der Vorbereitung auf die kommenden Semester stellen wollten, haben wir für das Wintersemester 2018/2019 eine Docker-basierte Umgebung aufgesetzt (Merkel, 2014). Darin erhält jeder und jede Studierende einen eigenen gut ausgestatteten Container mit ssh- und Webserver, auf dem sie sich wiederum einloggen und mit cgi-Skripten dynamisch Seiten erzeugen können. Java ist dort ebenfalls installiert, so dass sie ihre beginnenden Java-Kenntnisse frühzeitig anwenden können, ohne mit der Komplexität von Enterprise-Java (Servlets etc.) oder GUI-Programmierung (AWT/Swing) schon konfrontiert zu werden.

Auf den Arbeitsservern sowie in den Containern ist eine durchaus typische Server- und Arbeitsumgebung installiert. Aufgrund der Gegebenheit, von Beginn an nur in der Kommandozeile zu arbeiten, um alle Arbeitsschritte wiederholbar und automatisierbar durchführen zu können, trotzdem aber die Studierenden nicht auf die für sie zunächst unmodern und unhandlich wirkende reine Textausgabe zu beschränken, wurde ein spezifischer Arbeitsfluss etabliert: Wo immer sie etwas mit einem selbst geschriebenen Programm erzeugen, nutzen sie das Webverzeichnis, um sich das Ergebnis im Browser anzeigen lassen zu können und gegebenenfalls auch Freunden und Familie frühzeitig zu zeigen, was sie im Studium tun. So lernen sie früh das Werkzeug *gnuplot* kennen, um Graphen visuell ansprechend als PDF oder SVG zu erzeugen und sie lernen LaTeX, um Dokumente so zu generieren, dass sie zum einen heutigen ästhetischen Ansprüchen genügen als auch die sorgfältige Trennung von Struktur und Darstellung verdeutlichen. Da sie frühzeitig HTML von Hand und ohne all zu mächtige Hilfsmittel zu schreiben angehalten werden, können sie systematisch Listen und Tabellen mit Skripten oder mit Java erzeugen.

Der grundsätzliche Arbeitsrhythmus ist also: *Erstelle ein Programm, das etwas produziert, das im Web angezeigt werden kann - sei es ein HTML-Dokument, eine PDF-Datei, ein Bild (PNG) oder eine Grafik (SVG/PDF)*. Wie in dem Abschnitt zum *Workshop Videoerstellung* dargestellt, gehen wir dabei soweit, dass aus einer Menge von generierten Grafiken automatisiert eine Menge von Bildern und daraus dann ebenfalls automatisiert ein Video erstellt werden kann. Die Werkzeuge dafür (*inkscape* und *ffmpeg*) gehören daher auch zu unserer Standardumgebung.

## Verzahnung von Modulen

Um Studierenden einen Zugang und ein entsprechendes Verständnis für grundlegende Konzepte der Informatik zu ermöglichen, verfolgen wir in der Studieneingangsphase den Ansatz, verschiedene Fachmodule derart miteinander zu verzahnen, dass spezifische Themen aus jeweils einem der Module auch in anderen Modulen aufgegriffen und behandelt werden. Eine solche Verzahnung ermöglicht es, Themen aus verschiedenen Perspektiven und anhand von verschiedenen Beispielen zu betrachten. Die Hoffnung dabei ist, dass mehr Studierende zeitnah beim Stoff der Vorlesun-

gen mitgenommen werden und verschiedene Lerntypen die Chance haben, den Stoff zu erfassen. Für die Verzahnung ist es wichtig, dass sich die beteiligten Lehrenden der jeweiligen Module (mindestens der Module STEP, Programmieren, Mathematik, Software Engineering) abstimmen, in welchen Wochen voraussichtlich welche Themen behandelt werden. Es geht dann nicht darum, auf dem Vorwissen aus den jeweils anderen Veranstaltungen aufzusetzen, sondern das Thema mit dem Ansatz und aus der Perspektive der jeweiligen Veranstaltung einzuführen und dabei ggf. Beziehungen zu den anderen Veranstaltungen herzustellen. Dieses Konzept trägt auch den pädagogischen Erkenntnissen Rechnung, dass neues Wissen nachhaltiger erworben werden kann, wenn es in einen breiten Kontext vorhandenen Wissens eingeordnet werden kann:

> *"As just explored, a wide range of converging evidence stresses the fundamental importance of the learning edge, the boundaries of existing knowledge which form the context into which newly learned concepts (information, understanding and skills) must be integrated. This fact is widely understood and accepted, and the basis of many sound pedagogical practices."*
>
> (Robins, 2010, S.66)

Unser Curriculum sieht insgesamt drei Module für Software Engineering (SWE) vor: Im ersten Semester geht es um Modellierung im Allgemeinen und UML im Besonderen. Im zweiten Semester wenden die Studierenden die gelernten Methoden in einem konkreten Kundenprojekt ihrer Wahl an, während im dritten Semester anhand von Fallbeispielen ein besonderer Fokus auf Software-Architekturen und Qualitätssicherung gelegt wird. Die enge Verzahnung mit Inhalten des STEP-Projektes bietet in SWE I die Chance, Modellierungsbeispiele direkt auf die sehr praxisbezogenen Aufgabenstellungen der STEP-Projekte zu beziehen. Insbesondere die Erstellung von UML-Aktivitätsdiagrammen zu konkreten bash-Skripten aus den STEP-Projekten führt einerseits zu einem besseren Verständnis für den Sinn der Modellierung. Andererseits erhalten die Studierenden dadurch einen anderen, visuellen Blick auf den Quellcode, der vielen einen weiteren Zugang zum Verstehen der Programme und Programmierkonzepte ermöglicht.

Im Folgenden skizzieren wir einige Beispiele zur Verzahnung von Modulen aus der Studieneingangsphase unserer Bachelorstudiengänge Informatik und Wirtschaftsinformatik in den Wintersemestern 2017/2018 und 2018/2019.

### Modellbildung

Die Aufgabenstellung des STEP-Projektes 2017/2018 eignete sich besonders gut zur Erläuterung einiger Grundprinzipien der Modellierung im Allgemeinen, die in SWE I vermittelt werden. Zum Original, dem Schlepper Tide (siehe Abbildung 2), der seinen Liegeplatz direkt vor dem Fährhaus hat, gibt es mehrere Modelle: Zum einen diente die Konstruktionszeichnung (siehe Abbildung 3) als präskriptives Modell sowohl für den Original-Schlepper als auch für den 3D-Druck der Modellboote.

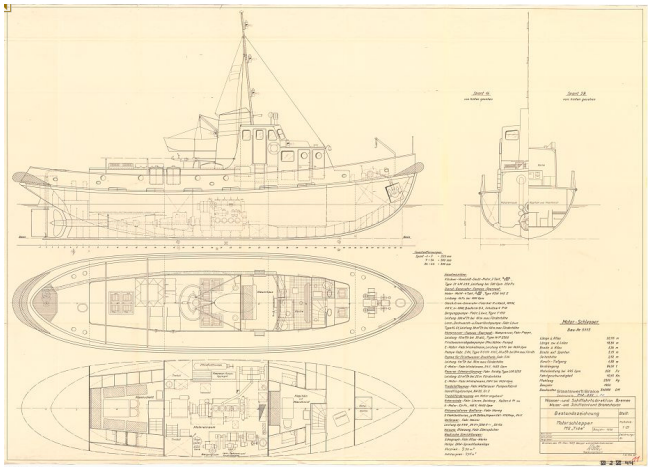


Abbildung 3: Konstruktionszeichnung der Tide (DigiPEER)

Zum anderen ist das ausgedruckte STEP-Boot (siehe Abbildung 1) ein Modell, das einige wenige Merkmale des Originals abbildet. Abbildung, Verkürzung und Pragmatismus, die von Stachowiak (Stachowiak, 1973) benannten typischen Merkmale von Modellen, werden an diesem Beispiel greifbar.

Neben solchen allgemeinen Aspekten der Modellierung erfahren die Studierenden ganz konkret den Nutzen der Modellierung beim Software Engineering, wenn sie in SWE I die Aufgabe erhalten, das Konzept für die im STEP-Projekt zu entwickelnde Anwendung zunächst mittels UML-Anwendungsfalldiagramm zu entwerfen und später einzelne Abläufe per UML-Aktivitätsdiagramm zu konkretisieren. In den STEP-Projekten werden keine festen Vorgaben zur erwarteten Funktionalität gemacht, sondern vor allem ein Rahmen abgesteckt. Die Idee für das Endprodukt wird von den Teams jeweils selbst entwickelt. Anwendungsfalldiagramme sind dann eine gute Methode für die Verständigung im Team über das zu entwickelnde System.

### Kontrollstrukturen, Aktivitätsdiagramm und mathematische Berechnungen

In der STEP-Veranstaltung werden stets auch Programmierprinzipien aus der Java-Programmierveranstaltung aufgegriffen und deren Umsetzung in der Bash gezeigt. Auf diese Weise wird die Programmiersprachen-übergreifende Bedeutung von Konstrukten wie bedingten Anweisungen und Schleifen unterstrichen. In SWE erhalten die Studierenden durch die Modellierung dieser Konstrukte in Aktivitätsdiagrammen auch einen visuellen Zugang. Ein anschauliches Beispiel ist die Modellierung der

Erhöhung der Geschwindigkeit der STEP-Boote aus dem STEP-Projekt 2017/2018. Die Boote konnten per Browser durch den Aufruf entsprechender CGI-Skripte gesteuert werden, die auf einem Webserver, dem Raspberry Pi der Boote, ausgeführt wurden. Ein Skript diente z.B. dazu, die Boote zu beschleunigen, bis sie sich zu sehr in Schieflage neigten. In SWE I sollte dies mittels UML-Aktivitätsdiagrammen mit 'Schwimmbahnen' modelliert werden. Eine mögliche Lösung dazu findet sich in Abbildung 4. Der zyklische Verlauf einer Schleife kann im Aktivitätsdiagramm gut veranschaulicht werden.

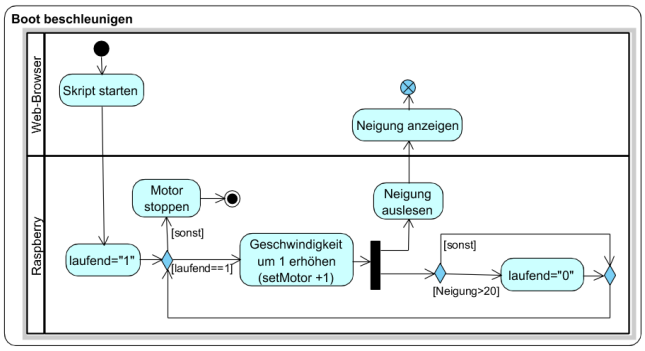



Abbildung 4: Aktivitätsdiagramm Boot beschleunigen

Die Nutzung von Schleifen wurde auch zum Beispiel anhand der Dezimal-Binär-Umwandlung mit Modulo-Berechnung verdeutlicht. Mit diesem Beispiel wird sowohl ein Bogen zur Mathematik-Veranstaltung geschlagen, als auch das bash-scripting einer Schleife gezeigt (siehe Abbildung 5). Zudem kommen dafür in den Aktivitätsdiagrammen Notationselemente wie Verzweigung und Zusammenführung zum Einsatz.

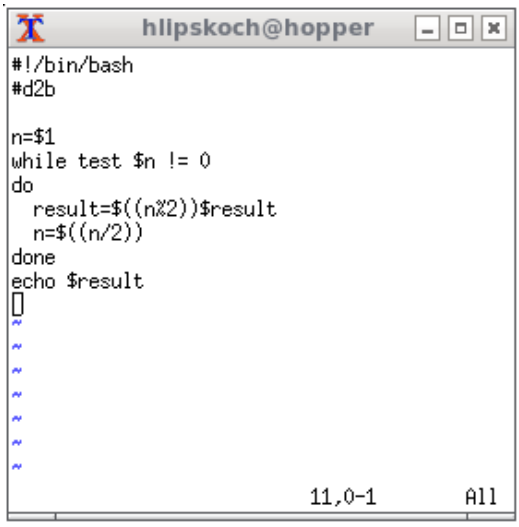



Abbildung 5: Bash-Skript Dezimal-Binär-Umwandlung

### Verzahnung mit der Mathematik am Beispiel RSA

Im Modul Mathematik werden die Begriffe Binärzahlen, Dividieren mit Rest und Kombinatorik mit der Fakultät in jeweils eigenen Veranstaltungseinheiten eingeführt. Binärzahlen als Grundlage der digitalen Computertechnik und das Dividieren mit Rest, welches ein Beispiel für die Endlichkeit mathematischer Berechnungen im Computer darstellt, erfahren durch die Verzahnung mit STEP direkte praktische Übungen. Diese Übungen wären sonst im Modul Mathematik aufgrund der erforderlichen Themenbreite für Unix, Shell-Skripte etc. nicht zu leisten.

Insbesondere kann so das RSA-Verschlüsselungsverfahren (Rivest u. a., 1977) praktisch ausprobiert werden: In der Mathematik lernen die Studierenden den Algorithmus und die mathematische Grundlage. Die Basis bilden zwei sehr große Primzahlen und dann werden nacheinander mehrere Zahlen daraus berechnet. Dabei wird auch das sogenannte Multiplikative Inverse benötigt. Hierzu lernen die Studierenden ebenfalls in der Mathematik den Euklidischen Algorithmus (Knuth, 1998) kennen und anwenden. Für die korrespondierende STEP-Aufgabe ist ein Programm, welches den Euklidischen Algorithmus ausführt und die einzelnen Schritte auf der Konsole darstellt, gegeben.

In Abbildung 6 ist die Ausgabe der Berechnung zu sehen. In der Zeile für $i = 5$ ist der Wert von $r$ gerade 1, was bedeutet, dass die Zahlen 310 und 617 keine gemeinsamen Teiler haben. Deswegen gibt es das Multiplikative Inverse bezüglich modulo 617. Es steht in der gleichen Zeile in der Alpha-Spalte. Da die Zahl negativ ist, müssen die Studierenden zusätzlich 617 addieren:

$$(310 \cdot (-205 + 617)) \mod 617 = 1$$

Diese Teilaufgabe ist nicht ohne die mathematischen Kenntnisse zu lösen.

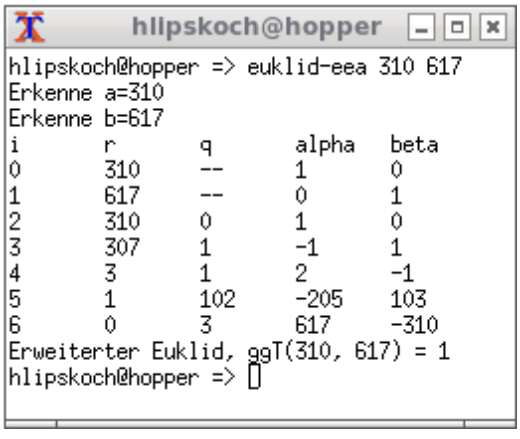



Abbildung 6: Erweiterter Euklidischer Algorithmus berechnet für die Zahlen 310 und 617

Die gesamte STEP-Aufgabe besteht daraus, dass die Studierenden zwei große Primzahlen finden und die verschiedenen weiteren benötigten Zahlen für RSA vermittels einfacher mathematischer Formeln (Multiplikationen) berechnen. Wie sie die Primzahlen finden, ist ihnen überlassen. Durch Shell-Skript-Aufruf

bestimmen sie das Multiplikative Inverse, in dem das gegebene Programm aufgerufen und aus der Ausgabe der entsprechende Wert herausgefiltert wird.

Danach besteht die Aufgabe daraus, mit den hergestellten Zahlen, das RSA-Verfahren anzuwenden: Das bedeutet wieder ein Skript zu schreiben, welches in der Lage ist Potenzen und Modulo zu berechnen. Am Schluss haben die Studierenden alle nötigen Werkzeuge hergestellt, um sich gegenseitig verschlüsselte Nachrichten zuzusenden und zu entschlüsseln. Die vier Hauptaufgaben eines Verschlüsselungsverfahrens finden Anwendung: Verschlüsselung, Entschlüsselung, Signierung, Verifizierung.

## Workshop Videoerstellung

Die Vermittlung der Kompetenz, selbst Visualierungen zu generieren, ist ein weiteres Thema im Rahmen der Step-Veranstaltung, auf das wir im Folgenden eingehen.

Videos werden von heutigen Studierenden gern und viel zum Lernen genutzt. Jedoch sind fremdgeschaffene Lernvideos zunächst einmal ein passives Medium und bieten Ihrer Natur nach eben keine Möglichkeit, den dargestellten Sachverhalt mit eigenen Parametern zu variieren oder mit eigenen Ideen anzureichern.

Das hier vorgestellte Instrument *selbsterstellte Videos zur Visualisierung komplexer Vorgänge* knüpft an die konstruktionistischen Ideen von Seymore Papert an:

> *Slowly I began to formulate what I still consider the fundamental fact about learning: Anything is easy if you can assimilate it to your collection of models. If you can't, anything can be painfully difficult.*
>
> (Papert, 1980, S. iiv)

Große Teile der Ideen von Papert sind explizit auf den Umgang von Schüler*innen mit Computern ausgerichtet, um sowohl das Programmieren als solches zu erlernen als auch die Lust am mentalen Modellieren und Prüfen der Modelle zu fördern. Demzufolge sind auch in dieser Tradition fortgeführte Programmierumgebungen (von Logo bis Scratch) nicht auf generische Programmiersprachen ausgerichtet und bisweilen für Studierende zu verspielt.

In unserem Ansatz verfolgen wir das Ziel, dass die Studierenden eine Grundmenge an Elementen an die Hand bekommen, mit der sie sich aus jeder Programmiersprache heraus selbst Visualisierungen höherer und abstrakterer Konzepte erstellen können.

### Scalable Vector Graphics

Das Format SVG (Scalable Vector Graphics) hat sich in den vergangenen Jahren zu einem Standardformat für Vektorgrafiken entwickelt. Jeder einigermaßen moderne Browser kann SVG-Grafiken nativ (also ohne Plugins) darstellen. SVG ist ein reines, auf XML basierendes Textformat - grafische Primitive wie Kreis, Rechteck, Linienzug oder Text können direkt im Editor eingegeben werden:

```
<svg width='400' height='200'
  xmlns='http://www.w3.org/2000/svg'>
  <rect x='0' y='0'
        width='400' height='200'
        stroke-width='1'
        stroke='black' fill='whitesmoke'/>
  <circle cx='100' cy='100' r='50'
          stroke='black' fill='gray' />
  <text x='10' y='30'>
    Ohne Programmieren ist nichts
    Informatik</text>
</svg>
```

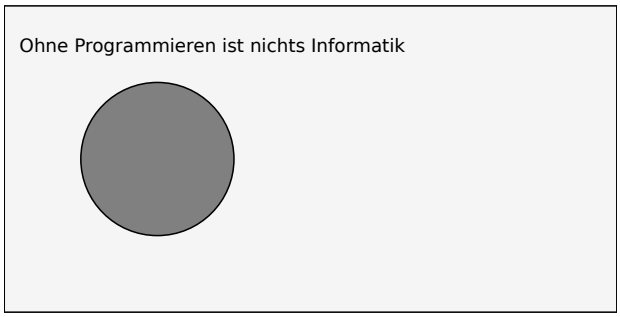



Abbildung 7: SVG Ausgabe

Es genügt zum Erstellen einer Grafik hier also zunächst die Fertigkeit zur Bedienung eines Editors und die Beherrschung einiger weniger Unix-Kommandos wie *cp* für das Kopieren der Datei in das Webverzeichnis. Beim Editieren ist immer auch ein Browser geöffnet, in dem das Ergebnis sehr schnell geprüft werden kann. Da die Studierenden zu dem Zeitpunkt, da wir SVG eingeführt haben, bereits HTML-Kenntnisse besitzen, ist die Syntax von SVG nicht mehr all zu fremd - die Tatsache, dass Baumstrukturen mit ineinander geschachtelten Tags in der speziellen Notation mit spitzen Klammern konstruiert werden, ist folglich keine große Hürde mehr. Das schnell wechselnde Arbeiten zwischen Editor und Browser ist ebenfalls eingeübt und stellt keine Probleme mehr dar.

### Einheitskreis

Als bedeutend größere Schwierigkeit stellt sich heraus, dass die Studierenden zwar mit den Worten Sinus, Cosinus, Pythagoras und Einheitskreis vertraut waren, jedoch die Zusammenhänge mit Koordinatensystemen ihnen zu einem erheblichen Teil so fremd waren, dass sie nicht - oft auch kaum nach ausführlicher Hilfe in Tutorien - in der Lage waren, einen Kreis aus 18 Kreisen zu konstruieren. Der Stoff aus dem Mathematikunterricht spätestens der 10. Klasse war nicht präsent - geschweige denn zu einem Teil der inneren Modellbildung geworden. Für einen Studiengang, der heutzutage so stark auf Visualisierungen und Grafik im Allgemeinen aufbaut, stellt das eine

ernstzunehmende Problematik dar, der zu begegnen ist; zumal nicht wenige der Studierenden selbst sehr visuell geprägt sind und das Programmieren von Computerspielen recht weit oben auf der Wunschliste der eigenen Befähigungen steht. In Zeiten, in denen 3D-Visualisierungen und -Animationen für jeden programmierwilligen Menschen erschwinglich und machbar ohne Spezialausrüstung sind, wird man die Studierenden gerade zu diesem so zukunftsträchtigen und spannenden Feld so nicht hinführen können: Kein API oder Framework nimmt einem die Notwendigkeit ab, mit drei ausgestreckten Fingern vor sich Rotationen im dreidimensionalen Raum in Cosinus und Sinus denken zu müssen.

## Grundannahme

Wer

- bereits einfache Skripte schreiben kann, die Texte ausgeben,
- Ausgaben auf der Kommandozeile ganz problemlos in eine Textdatei umleiten kann,
- Schleifen und Bedingungen ausdrücken kann,
- versteht, dass Programmaufrufe unter Unix immer sowohl in der Shell ausführbar sind als auch genau so in einem Skript mit Kontrollstrukturen zur Automatisierung stehen können,
- ein Kommandozeilenprogramm zur Verfügung hat, das SVG-Grafiken in PNG-Bilder konvertieren kann (inkscape) und
- ein Kommandozeilenprogramm zur Verfügung hat, das eine Menge von PNG-Bildern in ein digitales Video konvertiert (ffmpeg),

kann sich mit wenig Aufwand ein eigenes Video programmieren, in dem der Zusammenhang von Sinus, Cosinus und Einheitskreis dargestellt wird.

Im folgenden Abschnitt zeigen wir die konkrete Lerneinheit, die in Form eines Workshops an einem Vormittag den Studierenden etwa in der Mitte des Semesters gegeben wurde, nachdem in der Vorwoche klar wurde, dass Trigonometrie noch einmal vertieft werden könnte.

## Ein Workshop zum Einheitskreis

1. *Schreiben Sie ein Skript, das eine SVG-Datei der Breite 700 Einheiten und der Höhe 300 Einheiten ausgibt, in dem ein Rechteck derselben Größe an dem Punkt (0,0) positioniert wird.*

2. *Fügen Sie einen Kreis mit dem Radius 100 Einheiten an der Position (150,150) ein und zeichnen Sie ein Koordinatenkreuz ein, das durch eben diesen Mittelpunkt des Kreises geht. Für die folgenden Überlegungen gilt dieser Punkt als Nullpunkt in dem kartesischen Koordinatensystem und 100 graphische Einheiten entsprechen einer Einheit in diesem System.*

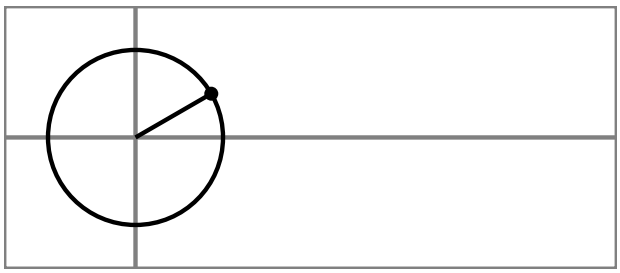


Abbildung 8: SVG Ausgabe für die Längen im Einheitskreis bei 30 Grad

3. *Variieren Sie Ihr Skript so, dass es als Argument eine Zahl in Graden von 0 bis 360 entgegennimmt und zeichnen Sie eine Linie von dem Mittelpunkt des Kreises bis zum Rand des Kreises bei der übergebenen Gradzahl. Fügen Sie dort noch einen Kreis mit dem Radius 8 ein. Berechnen Sie die Koordinaten mit bc.*

   $rad = \frac{degrees \cdot \pi}{180}$

   $x = cx + \cos(rad) \cdot r$

   $y = height - (cy + \sin(rad) \cdot r)$

   ```
   rad=$(echo ''($deg * 4*a(1))/180''|bc -l)
   x=$(echo ''$cx + s($rad)*$r''|bc -l)
   y=$(echo ''$h-($cy + c($rad)*$r)''|bc -l)
   ```

4. *Rufen Sie in einer Schleife mit i von 0 bis 360 Ihr Skript mit i als Argument auf und erzeugen Sie dadurch 361 SVG-Dateien der Form kreis$num.svg, wobei sich $num aus $\$i + 1000$ berechnet. Erzeugen Sie aus den SVG-Dateien ebenfalls in der Schleife jeweils eine PNG-Datei:*

   ```
   inkscape -z --export-png=kreis$num.png \
   kreis$num.svg
   ```

5. *Benutzen Sie ffmpeg nun, um aus den 361 Dateien eine Videodatei zu erzeugen:*

   ```
   ffmpeg -y -i kreis1%03d.png \
  -pix_fmt yuv420p kreis.mp4
   ```

   *Betrachten Sie das Video in Ihrem Browser.*

6. *Verändern Sie nun Ihr Skript so, dass zusätzlich zu der Linie, die vom Mittelpunkt zum Kreisrand gezogen wird, noch die Linie vom Kreismittelpunkt zu der berechneten X-Koordinate aber auf der gleichen Höhe wie der Kreismittelpunkt in blau eingezeichnet wird. Nun zeichnen Sie noch eine Linie von diesem Punkt zu dem berechneten Punkt auf dem Kreisrand in rot. Erzeugen Sie nun wieder ein Video und betrachten Sie den Verlauf des rechtwinkligen Dreiecks.*

7. *Verändern Sie in einem vorerst letzten Schritt noch einmal Ihr Skript so, dass – wieder in einer Schleife – alle Werte von 0 bis zur aktuellen Gradzahl für Cosinus (x-Achse) und Sinus (y-Achse) berechnet und neben dem Kreis als einzelne kleine Kreise mit einem Radius von 5 Einheiten aufgetragen werden.*

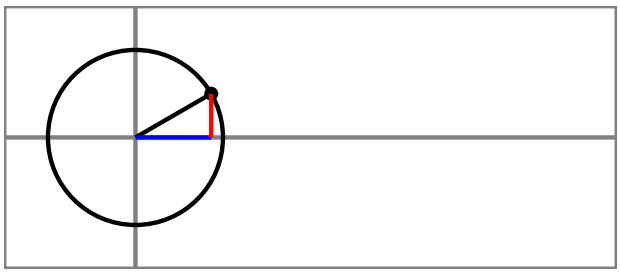


Abbildung 9: SVG Ausgabe für die Längen im Einheitskreis zur Veranschaulichung von Sinus und Cosinus bei 30 Grad

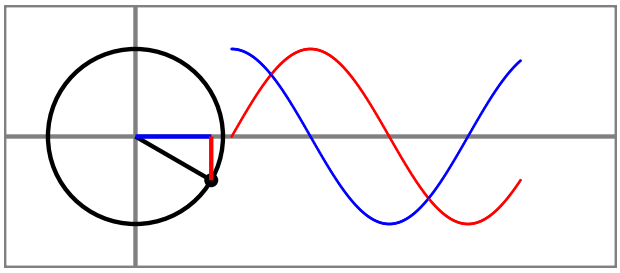


Abbildung 10: SVG Ausgabe für die Längen im Einheitskreis zur Veranschaulichung von Sinus und Cosinus bei 330 Grad mit Sinus- und Cosinus-Kurve

8. *Betten Sie die das Video in Ihre HTML-Datei nun so ein, dass es in einer Endlosschleife abläuft und betrachten Sie Ihr Werk einige Minuten. Versuchen Sie so das Verhältnis von Sinus und Cosinus im Einheitskreis als Längen in einem Koordinatensystem zu begreifen.*

### Kontextualisierung

SVG als Visualisierungswerkzeug schult die Studierenden von Beginn an darin, in Schnittstellen zu denken. Statt spezieller APIs oder sogar Programmiersprachen wird in guter Unix-Tradition als Interface reiner Text genutzt, der über Pipes und Ausgabeumlenkung in Dateien gelenkt wird. Zusammen mit der Möglichkeit, in Schleifen viele, systematisch benannte Dateien zu erzeugen und mit einem einzigen Programmaufruf in ein Video zu konvertieren, lassen sich selbständig komplexe Sachverhalte durchdenken. Das oben beschriebene Konzept führen wir bei uns in der Studieneingangsphase etwa in der Mitte des Semesters ein. Das Dateisystem, Schleifen, Bedingungen und einfache Skripte sind den Studierenden bis dahin ebenso wie HTML und SVG hinreichend nahe gebracht worden. Dass das Beispiel Sinus und Cosinus am Einheitskreis behandelt wird, ist lediglich der konkreten Situation geschuldet, dass klar wurde, dass zumindest in dieser Kohorte diesbezüglich Nachholbedarf bestand. Es könnte sich ebenso gut um einfache Sortieralgorithmen, das Durchlaufen von Arrays oder Algorithmen auf Graphen handeln.

Ebenfalls lässt sich das Konzept auf die Programmierung mit Java anwenden - allerdings nur in Verbindung mit der Shell. Das Öffnen und Schließen und Beschreiben von Dateien in Java erfordert ein komplexes Geflecht aus Objekten und Klassen, die zu Beginn des Programmierenlernens noch zu fremd sind. Mit *System.out.println* lässt sich jedoch für jeden Programmaufruf ein Datenstrom beschreiben - und so wird in einer Shell-Schleife jeweils ein Java-Programm aufgerufen, das wiederum SVG hinausschreibt.

Nach dem Workshop haben wir den Studierenden als Wochenaufgabe gestellt, mit den gleichen Konzepten (SVG, Bash und ffmpeg) *irgendeine* Animation zu erstellen - möglicherweise eine, die etwas aus dem bisherigen Studium veranschaulicht. Dabei sind sehr beeindruckende Ergebnisse entstanden: mehrfach wurde ein Sortieralgorithmus visualisert und die *Türme von Hanoi*, die in der Programmiervorlesung behandelt worden waren, fanden sich ebenfalls mehrfach wieder. Dass in fast allen anderen in irgendeiner Form mit den Kreisfunktionen experimentiert wurde (von Analoguhren bis zu Lissajous-Figuren) war angesichts der oben beschriebenen Ausgangssituation ein besonders erfreuliches Ergebnis.

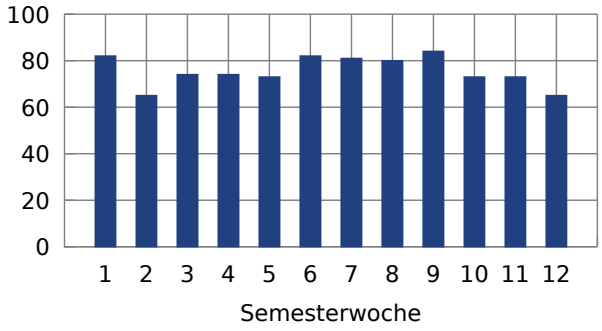
## Fazit

Die mittlerweile sechsjährigen Erfahrungen mit der Studieneingangsphase an der Hochschule Bremerhaven haben gezeigt, dass ein begleiteter, projektorientierter Studienstart die soziale Integration der Studierenden fördert und in den meisten Teams die Zusammenarbeit auch nach anfänglichen Schwierigkeiten sehr gut funktioniert. Die Teams kommen zu guten bis sehr guten Projektergebnissen und die Einzelnen zeigen deutliche Fortschritte in ihrer Lernentwicklung. Mit der Neuausrichtung der Projekte von reinen Analyseaufgaben zu Programmieraufgaben kann eine bessere Verzahnung der Grundlagenfächer hergestellt werden. Somit werden die Zusammenhänge für die Studierenden erfahrbarer.

Ob alle angestrebten Ziele auch tatsächlich erreicht werden, bedarf noch entsprechender Evaluationen. Bisher gründen unsere Aussagen dazu auf eigenen Eindrücken sowie auf den STEP-Blogs der Studierenden. Wie erwähnt wird von den Studierenden erwartet, dass sie in ihrem Teamordner auf der ILIAS-Lernplattform einen wöchentlichen, persönlichen Blogeintrag schreiben, in dem kurz zusammengefasst wird, was in allen Lehrveranstaltungen des ersten Semesters gelernt wurde, welche Schwierigkeiten im Wege standen und was in der kommenden Woche angestrebt wird. In Abbildung 11 ist das vorläufige Ergebnis zu sehen. Von etwa 90 Studierenden, die wir als aktiv studierend betrachten können, schreiben im Schnitt mehr als 70 pünktlich ihren Blogeintrag.

Diese Blogeinträge sind für uns ein wertvolles Evaluationsinstrument, bei dem wir wöchentliches Feedback darüber erhalten, was in den verschiedenen Fächern bei den Studierenden ”hängengeblieben” ist, wo zusätzlicher Erklärungsbedarf besteht oder ob die

Abbildung 11: Blog-Einträge pro Semesterwoche




Arbeitsbelastung zu hoch ist. Ein Wochenblogeintrag wie: ”In Programmieren und SWE haben wir Klassen und Objekte kennengelernt” zeigt zum Beispiel auch, wie gut die Verzahnung der Module funktioniert. Durch die Einträge wird den Studierenden selbst klar, was sie in der vergangenen Woche gelernt haben:

> *"Diese Woche haben wir uns in STEP mit der Erstellung eines Einheitskreises beschäftigt. Mit Hilfe des Programms ffmpeg, haben wir außerdem mehrere Bilder, mit verschiedenen Varianten des Kreises, zu einem Video zusammenfügen können.*
>
> *Die Wochenaufgabe in STEP empfand ich diese Woche wesentlich leichter, als in der vergangenen Woche und konnte sie auch schnell umsetzen. In Programmieren haben wir das Thema Rekursion wiederholt, hinzu kam das Thema Objektorientierte Programmierung in Java, sowie die Verknüpfung von mehreren Klassen.*
>
> *Weiterhin schwer fällt mir Mathe, aber da muss man halt durch. Zum Glück hilft ja bekanntlich das Internet bei Verständnisproblemen. Ich freue mich jedenfalls auf den Unternehmensbesuch in der nächsten Woche."*

Als ein erfreuliches Ergebnis der Studieneingangsphase kann zudem festgehalten werden, dass die Workshop-ähnliche Lernumgebung über das ganze Semester hinweg eine Wissensvermittlung in einer sehr intensiven Lernatmosphäre ermöglicht hat.

## Literatur

[Dennert-Möller u. Garmann 2016] Dennert-Möller, Elisabeth ; Garmann, Robert: Das „Startprojekt“ - Entwicklung überfachlicher Kompetenzen von Anfang an. In: Schwill, Andreas (Hrsg.) ; Lucke, Ulrike (Hrsg.): *HDI 2016 : Hochschuldidaktik der Informatik*, 2016, 11 – 24

[DigiPEER ] DigiPEER: *Digitalisierung großformatiger Pläne und technischer Zeichnungen zur Erfassung und Erschließung des Raums*. – http://www.digipeer.de/index.php Zugriff: 02.12.2018

[Key u. Hill 2018] Key, Olivia ; Hill, Lukasz: Die Studieneingangsphase im Umbruch. Anregungen aus den Hochschulen. In: *nexus impulse für die Praxis* (2018), Nr. 14

[Knuth 1998] Knuth, Donald E.: *The Art of Computer Programming*. Bd. 2, Seminumerical Algorithms. 3. Addison-Wesley Professional (Pearson), 1998

[Merkel 2014] Merkel, Dirk: Docker: Lightweight Linux Containers for Consistent Development and Deployment. In: *Linux J.* 2014 (2014), März, Nr. 239. – ISSN 1075–3583

[Papert 1980] Papert, Seymour: *Mindstorms: children, computers, and powerful ideas*. New York, NY, USA : Basic Books, Inc., 1980. – ISBN 0–465–04627–4

[Rivest u. a. 1977] Rivest, R. L. ; Shamir, A. ; Adleman, L.: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. 1977. – http://people.csail.mit.edu/rivest/Rsapaper.pdf

[Robins 2010] Robins, Anthony V.: Learning edge momentum: a new account of outcomes in CS1. In: *Computer Science Education* 20 (2010), Nr. 1, S. 37–71

[Schmolitzky 2017] Schmolitzky, Axel: Zahlen, Beobachtungen und Fragen zur Programmierlehre. In: Brügge, Bernd (Hrsg.) ; Krusche, Stephan (Hrsg.): *Tagungsband des 15. Workshops Software Engineering im Unterricht der Hochschulen*, 2017, 83–90

[Stachowiak 1973] Stachowiak, Herbert: *Allgemeine Modelltheorie*. Bd. 2, Seminumerical Algorithms. Wien : Addison-Wesley Professional (Pearson), 1973. – ISBN 3–211–81106–0

[Steven J. Vaughan-Nichols ] Steven J. Vaughan-Nichols: *Linux now dominates Azure*. – https://www.zdnet.com/article/linux-now-dominates-azure/ Zugriff: 30.10.2018

[Vosseberg 2015] Vosseberg, Karin: Mit Projekten ins Studium starten. In: Schmolitzky, Axel (Hrsg.) ; Hauptmann, Anna S. (Hrsg.): *Tagungsband des 14. Workshops Software Engineering im Unterricht der Hochschulen*, 2015, 123–124

[Vosseberg u. a. 2015] Vosseberg, Karin ; Czernik, Sofie ; Erb, Ulrike ; Vielhaber, Michael: Projektorientierte Studieneingangsphase. Das Berufsbild der Informatik und Wirtschaftsinformatik schärfen. In: Schubert, Sigrid (Hrsg.) ; Schwill, Andreas (Hrsg.): *HDI 2014 : Gestalten von Übergängen*, 2015, 169 – 177

[WSV 2018] WSV: *Maritime Verkehrstechnik*. 2018. – https://www.gdws.wsv.bund.de/DE/schifffahrt/03_verkehrstechnik/verkehrstechnik-node.html