GI-Edition

Lecture Notes in Informatics

Jens Gallenbacher (Hrsg.)

Informatik allgemeinbildend begreifen

INFOS 2015
16. GI-Fachtagung Informatik und Schule

20.–23. September 2015

Proceedings

249

Jens Gallenbacher (Hrsg) Informatik allgemeinbildend begreifen – INFOS 2015

Jens Gallenbacher (Hrsg.)

16. GI-Fachtagung Informatik und Schule – INFOS
Informatik allgemeinbildend begreifen

20.-23. September 2015
Darmstadt, Deutschland

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-249

ISBN 978-3-88579-643-5
ISSN 1617-5468

Volume Editors
Dr. Jens Gallenbacher
Technische Universität Darmstadt
Fachbereich Informatik
64289 Darmstadt, Germany
E-Mail: jg@di.tu-darmstadt.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort zur 10000. Infos
Informatik allgemeinbildend begreifen

Die beiden hinteren Worte des diesjährigen Mottos – Informatik allgemeinbilden begreifen – kann man beim Lesen unterschiedlich betonen, es werden gleichzeitig zwei wichtige Grundsätze für Informatik in der Schule beschrieben.

Selbstverständlich ist Informatik ein allgemeinbildendes Schulfach und muss als solches begriffen werden. Daher gehört es von der Grundschule an bis zur Oberstufe in den Pflichtbereich der Stundentafeln und Curricula. Informatik als allgemeinbildendes Fach ist daher auch mit einer besonderen Verantwortung beim Unterrichten verbunden: Nicht technische Werkzeuge und deren Bedienung, sondern das Verständnis unserer technischen Umwelt sowie die kreative und verantwortungsvolle Gestaltung unserer Zukunft stehen hier im Mittelpunkt. Das wörtliche „begreifen" unterstützt dabei den Lernprozess besonders wirkungsvoll und nachhaltig.

Informatik *allgemeinbildend* begreifen

Unsere Arbeits- und Lebenswelt ist geprägt von Informationstechnologie und Informatik ist der Schlüssel, um diese zu verstehen und um sie aktiv mitzugestalten. Dabei „versteckt" sich die Informatik bei der reinen Benutzung von IT-Systemen zunehmend, der alltägliche Umgang ist daher eher implizit. Die Beschäftigung mit Informatik in der Schule ist daher auch im engen Sinne unverzichtbar und sowohl lebensvorbereitend als auch weltorientierend.

Informatik ist die einzige Ingenieurwissenschaft im schulischen Kontext und ist aufgrund ihrer inhärent interdisziplinären Ansätze wie kein anderes Fach geeignet, Brücken zwischen den fachlichen und außerfachlichen Kulturen zu bauen und damit kulturelle Kohärenz herzustellen sowie Verständigung und Kooperation zu fördern.

Informationstechnik und informatisches Verständnis dafür spielt eine entscheidende und noch weiter zunehmende Rolle in Bezug auf kritischen Vernunftgebrauch sowie bei der Entfaltung von Verantwortungsbereitschaft. Die Hintergründe von Aspekten wie Cybermobbing und der NSA-Affäre sind ohne Zweifel nicht informatischer Natur, die Wirkung wird aber über die Möglichkeiten der Informationstechnik auf eine neue Stufe gebracht, weshalb ein grundlegendes Verständnis relevant ist, um staatsbürgerlicher Verantwortung nachzukommen und staatsbürgerliches Handeln zu ermöglichen.

Nicht zuletzt verbindet Informatik ein hohes gestalterisches Potential mit zumeist sehr schnell sichtbaren Resultaten und Erfolgserlebnissen. Selbst sehr komplex und schwierig erscheinende Aufgaben können oft mit informatischen Methoden elegant und eigenständig von Schülerinnen und Schülern gelöst werden, was einen wichtigen Beitrag zur Stärkung des Schüler-Ich leistet.

Informatik allgemeinbildend *begreifen*

Ausrichterin der INFOS 2015 ist die Didaktik der Informatik an der Technischen Universität Darmstadt. Das wörtliche Begreifen wird hier seit langem als wichtiger Baustein für das übergeordnete Begreifen gesehen. So werden etwa in der Studieneingangsphase

von allen Studierenden zusammen fachübergreifend interdisziplinäre Projekte durchgeführt.

In anderen klassischen Schulfächern – den Naturwissenschaften, Mathematik, aber selbstverständlich auch etwa in Kunst, Musik und Sport – ist das wörtliche Begreifen schon sehr lange wichtiges Thema und hat einen festen Platz in der fachdidaktischen Forschung. Die Informatik war lange Zeit sehr hardwareorientiert und war insofern ohne das vorherige „begreifen" kaum möglich – die Ausstattung einer Schule beruhte oft genug darauf, dass engagierte Lehrerinnen und Lehrer mit ihren Klassen tage- und nächtelang bestückt und gelötet hatten.

Die Informationstechnik hat seitdem eine rasante Entwicklung durchlaufen – die Informatik ebenfalls. Das Begreifen bezieht sich daher heute eher auf die wichtigen Kompetenzen der Informatik: Modellbildung und Problemlösefertigkeiten. Diese werden im Unterricht anhand moderner Computersysteme und Entwicklungssoftware vermittelt, aber eben auch vermehrt durch anschauliche Experimente ganz ohne Computer, die einerseits sehr starke didaktische Reduktion ermöglichen, andererseits in den Vordergrund stellen, was die Hauptkomponente der Informatik ist: Das menschliche Denken und die menschliche Kreativität.

INFOS 2015

In diesem Band finden Sie sehr viele Impulse, die Informatik allgemeinbildend zu begreifen – wissenschaftliche Untersuchungen wie Praxisberichte und die Ausarbeitungen von Workshops. Die 10000. INFOS[1] steht ganz im Zeichen der Arbeit für die Schule und in der Schule.

Das Programmkomitee hat die hier veröffentlichten 33 Arbeiten aus insgesamt 77 Einreichungen ausgewählt. Ich danke allen Beteiligten für den sehr konstruktiven Review-Prozess.

Ich freue mich auf anregende Erkenntnisse und Diskussionen (nicht nur) im Rahmen der INFOS.

Fachdidaktik der Informatik ist keine Einzelleistung, alle Lehrerinnen und Lehrer erweitern zusammen mit den Wissenschaftlerinnen und Wissenschaftlern an den Universitäten durch Unterrichtserfahrung, „best practice" sowie gezielte Forschung unser Verständnis für die allgemeinbildende, begreifbare Informatik.

Darmstadt, im September 2015

Jens Gallenbacher

[1] Es sei legitim, das in der Informatik meistgebrauchte Zahlensystem zur Bestimmung „glatter" Jubiläen zu nutzen…

Programmkomitee

Jens Gallenbacher, Darmstadt (Vorsitz)
Gerhard Röhner, Dieburg (stellv. Vorsitz)
Ira Diethelm, Oldenburg
Leonore Dietrich, Heidelberg
Michael Fothe, Jena
Steffen Friedrich, Dresden
Lutz Hellmig, Rostock
Tino Hempel, Ribnitz-Damgarten
Henry Herper, Magdeburg
Peter Hubwieser, München
Alexander Hug, Koblenz
Maria Knobelsdorf, Berlin
Johannes Magenheim, Paderborn
Peter Micheuz, Klagenfurt
Jürgen Poloczek, Frankfurt
Ralf Romeike, Erlangen
Nicole Schweikardt, Berlin
Monika Seiffert, Hamburg
Kerstin Strecker, Göttingen
Jan Vahrenhold, Münster
Helmut Witten, Berlin

Organisationsteam

Dominik Heun, Darmstadt
Wiebke Kothe, Darmstadt

Inhaltsverzeichnis

Der besondere Charme der INFOS liegt im Spannungsfeld zwischen fachdidaktischer Forschung und schulischer Praxis. Beide Aspekte sind essentiell und profizieren maximal voneinander. Dieser Tagungsband reflektiert das durch die gleichberechtigte, alphabetische Reihung nach den Nachnamen der Erstautoren. Die Forschungsbeiträge, Praxisbeiträge und Ausarbeitungen zu Workshops sind dabei mit (F), (P) und (W) gekennzeichnet.

E-Learning vor Präsenzveranstaltung – eine „Flipped-Vorlesung“ in der Lehrerausbildung der TU Dresden

Peter Arnold[1], Michael Rudolph[2] und Holger Rohland[2]

Abstract: Durch die Methode des Flipped Classroom wird die Phase des Dozentenvortrages aus dem Vorlesungssaal verlegt und stattdessen die aktive und tiefgreifende Auseinandersetzung mit den Vorlesungsinhalten in den Mittelpunkt gerückt. Der Beitrag thematisiert die aktive Anwendung des Konzepts auf eine bestehende Vorlesung der Lehrerausbildung an der TU Dresden und zieht ein Fazit. Es soll aufgezeigt werden, wie sich Lehrveranstaltungen von einer eher fachlich und formal orientierten Vermittlung von Lerninhalten hin zu handlungs- und kompetenzorientierten Szenarien entwickeln können, um neuen Anforderungen eines Hochschulstudiums gerecht zu werden.

Keywords: Flipped Classroom, E-Learning, Blended Learning, Vorlesung, Hochschuldidaktik, Informatik, Lehrerausbildung, Lehramt

1 Einleitung und Motivation

Durch die Zunahme der allgemeinen Mobilität und den Bologna-Prozess steigt die Heterogenität der Studierenden in den Lehrveranstaltungen. Es sind neue Konzepte gefragt. Eine Möglichkeit ist die Methode des Flipped Classroom. Dabei wird die Phase des Dozentenvortrages aus dem Vorlesungssaal verlegt und stattdessen die aktive und tiefgreifende Auseinandersetzung mit dem Vorlesungsgegenstand in den Mittelpunkt gerückt. Unterwegs oder Zuhause: An PC, Tablet oder Smartphone sehen sich die Studenten eine zusammenfassende Videoaufzeichnung des Dozentenvortrages an. In der Zeit der Vorlesung erhalten sie Aufgaben, die zu diesem Vortrag passen. Es werden Vorlesung und Nacharbeit organisatorisch vertauscht. Auf diese Weise können Fragen geklärt, Diskussionen angestoßen und kooperative Methoden besser genutzt werden, als es in einer „normalen“ Vorlesung der Fall wäre. Moderne Lernportale bzw. Lernmanagementsysteme besitzen einmalige Möglichkeiten, jedoch gilt dies auch für traditionelle Wege der Wissensvermittlung im Hochschulbereich, wie die Vorlesung. Die Methode Flipped Classroom kombiniert moderne Formen von E-Learning und Präsenzveranstaltungen zu einem sinnvollen Lernarrangement. Durch den Flipped Classroom erhält der Dozent mehr Zeit, um während der Präsenzphase individuell auf Studenten einzugehen, denn die starre Phase der Präsentation der Lerninhalte ist

[1] TU Dresden, Zentrum für Lehrerbildung, Schul- und Berufsbildungsforschung, Zellescher Weg 20, 01217 Dresden, peter.arnold@mailbox.tu-dresden.de
[2] TU Dresden, AG Didaktik der Informatik/Lehrerbildung, Nöthnitzer Straße 46, 01187 Dresden, mail@mrudolph.com / holger.rohland@tu-dresden.de

vorgelagert. Die Studenten erhalten dagegen die Möglichkeit, individuell und selbstbestimmt ihrem Lerntyp und Lerntempo entsprechend zu arbeiten [La14].

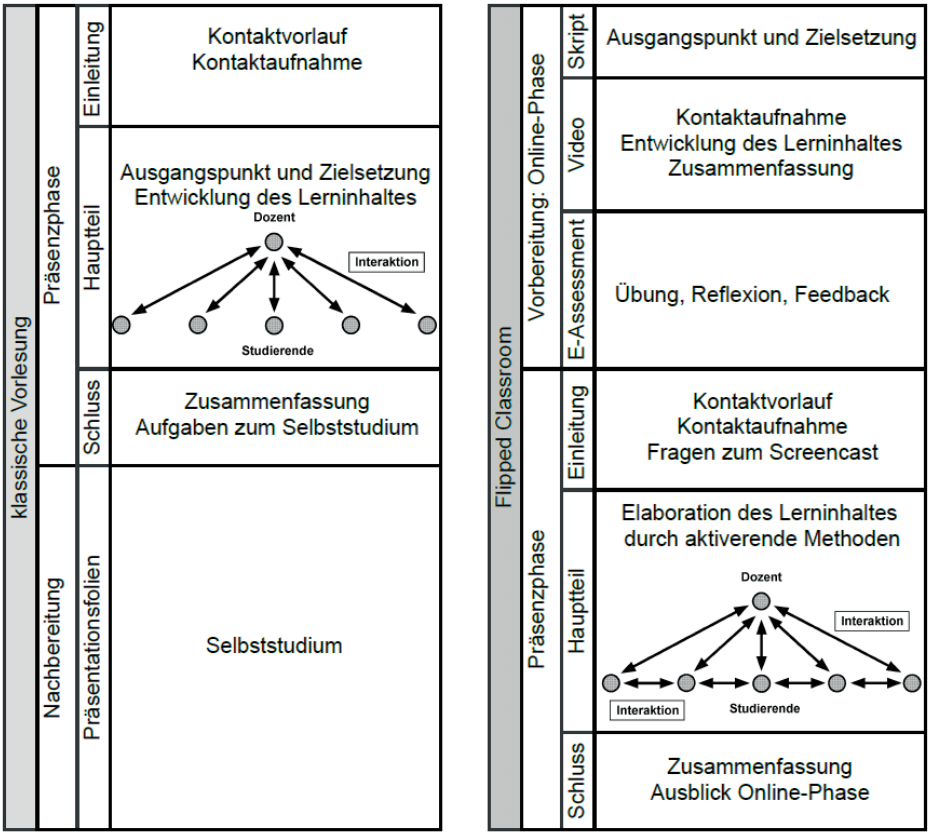


Abb. 1: Vergleich einer klassischen Vorlesung mit einem Flipped Classroom [Ru14], auf Grundlage von Berendt u.a. [Be13]

2 Überblick über das Modul „Anwendersysteme“[3]

2.1 Verortung im Studienablauf und Vorwissen

Die Vorlesung Anwendersysteme umfasst laut Modulbeschreibung eine Semesterwochenstunde Vorlesung und zwei Semesterwochenstunden Übung. Die Vorlesung hat grundlegenden Charakter und steht somit am Beginn (1. Fachsemester)

[3] basierend auf der Modulbeschreibung [Mo07]

der Lehrerausbildung für Lehramtsstudierende des Faches Informatik. Von den Studierenden werden grundlegende Kenntnisse zur Bedienung von Computern und deren Standardanwendungen, zur Dateiverwaltung und zur Benutzung einfacher Netzwerkdienste erwartet. Da das Schulfach Informatik in einigen Bundesländern, Schulen bzw. Schularten immer noch keinen verpflichteten Charakter besitzt, hat die Erfahrung gezeigt, dass Studierende eine starke Heterogenität in ihrem Vorwissen und ihren bis dato erworbenen entsprechenden Kompetenzen aufweisen.

2.2 Inhalte des Moduls

Inhaltlich reicht die Lehrveranstaltung „Anwendersysteme“ von der objektorientierten Betrachtung von Anwendersoftware, über eine Einführung in HTML und CSS bis hin zur relationalen Konzeption und Implementierung von Datenbanken. Einen Einstieg geben Standardanwendungen der Textverarbeitung und Tabellenkalkulation. Hierbei werden praktische Beispiele verknüpft mit informatischen Modellen und Vorgehensweisen. Die Objektorientierung bietet dabei einerseits die Grundlage zur Erschließung vorhandener Software und liefert andererseits einen Einstieg zur Modellierung und Problemlösung. Eine Einführung in HTML, CSS und JavaScript gibt einen ersten Einblick in den Umgang mit standardisierter Syntax, semantischer Gliederung von Information und ersten Programmierbeispielen. Im weiteren Verlauf der Lehrveranstaltung werden Datenbanksysteme genauer betrachtet. Die Studierenden planen Datenbanken in Entity-Relationship-Modell, überführen dieses in das Relationenmodell und implementieren dies dann in einem Datenbank-managementsystem. Hierzu werden theoretische Grundlagen vermittelt, wie Kenntnisse in ER- und UML-Modellierung, Transformation in das Relationenmodell durch Anwendung von Transformationsregeln und Beachtung der Normalformen, Überblick über Datentypen und die Bestimmung von Kardinalitäten von Beziehungen. Abschließend wird auf Abfragen, Formulare und Berichte im Rahmen des genutzten Datenbankmanagementsystems eingegangen.

2.3 Bisheriger Ablauf

Der bisherige Ablauf orientierte sich an der organisatorischen Rahmenvorgabe 2 SWS[4] Übung und 1 SWS Vorlesung. Die Vorlesung fand so 14-täglich und die Übung wöchentlich statt. Seit 2008 werden dabei Skripte und Übungsaufgaben über das Lernmanagementsystem OPAL[5] angeboten. Organisatorische Belange, wie die Einschreibung in Übungsgruppen oder die Abstimmung von Terminen wurden genauso mit dem Portal umgesetzt, wie Abgaben studentischer Lösungen, die an zentralen Stellen gefordert und kommentiert zurückgegeben wurden.

[4] Semesterwochenstunde(n)
[5] Online Plattform für akademisches Lernen der BPS Bildungsportal Sachsen GmbH

Vorlesung	Übung	OPAL
V1: Gegenstand der Informatik	Textverarbeitung	Organisation und Dokumente
	Tabellenkalkulation	Abgabe
V2: Modellierung	Klassen, Objekte in Anwend.[6]	
	Kooperation von Anwend.	
V3: Web-Präsentationen	HTML I, Strukturierung	Abgabe
	HTML II, Verweise	
V4: Dynamik in Web-Präs.	HTML III, CSS	
	HTML IV, Formulare	
V5: Datenbankentwurf I	Komplexaufgaben (bis V4)	
	DB Entwurf	
V6: Datenbankentwurf II	Entwurf und Implementierung	
	Modifikation und Analyse	Abgabe
V7: Abfragen und Berichte	Abfragen und Berichte erstellen	
	Abfragen und Formulare	

Tab. 1: Ablauf der Lehrveranstaltung im WS 2013/2014[7]

Seit 2008 wurde darauf geachtet, ab dem ersten Fachsemester das Lernmanagementsystem in den Studienalltag der Studenten zu integrieren. Das Verschicken von Lösungen per E-Mail oder das Einschreiben in aushängende Listen der Fakultät wurde obsolet. Da die Lehrveranstaltung einen einführenden Charakter hat, ist die Nutzung des Portals durchaus für weiterführende Veranstaltungen grundlegend.

3 Die Flipped Vorlesung

Zur Integration der Methode Flipped Classroom fand eine stufenweise Anhebung der Aktivitäten in zwei Phasen statt, da die Studierenden in ihrem ersten Fachsemester das genutzte Lernmanagementsystem erst kennenlernen mussten. Dabei gleicht die erste Phase organisatorisch den ersten 4 Vorlesungen und 8 Übungen, die zweite Phase überspannt dann den restlichen Teil, inhaltlich also den Bereich der Datenbanken.

[6] Anwendungen
[7] eine Zelle entspricht einer 90-minütigen Lerneinheit, in den Rubriken Vorlesung/Übung

3.1 Erste Phase: kumulative Nutzung von Screencasts und Online-Tests in den Übungen

Die erste Phase hatte einen einführenden Charakter, um die Studierenden an die Arbeit mit dem Lernmanagementsystem OPAL heranzuführen und eine permanente Kommunikation sowie Kollaboration über dieses System herzustellen. Vor allem die Übung erfuhr permanente Unterstützung durch Screencasts[8], um zentrale Bedienkompetenzen zu entwickeln und wiederkehrende Konzepte zu verdeutlichen. Den Studierenden standen so insgesamt 20 verschiedene Screencasts zur Verfügung, um plattformübergreifend Software zu ergründen und Prinzipien der Objektorientierung zu verinnerlichen:

- Screencasts zur Textverarbeitung[9]: Absatzeigenschaften, Formatvorlagen, Abschnitte/Seitenvorlagen, Seriendruck
- Screencasts zur Tabellenkalkulation: Datenformate, Funktionen, Zellbezüge, Diagramme
- Screencasts zur Web-Präsentation: Tags, Grundgerüst, Formatierung ohne und mittels CSS

Die online einzureichenden Abgabeaufgaben wurden in ihrer Form und Durchführung beibehalten. Hinzu kamen noch zwei komplexere Online-Tests, welche in Form von Multiple-Choice, Single-Choice, Lückentexten sowie Zuordnungsaufgaben bearbeitet werden konnten [On14] und über OPAL bereitgestellt wurden. Der erste dieser Tests prüfte inhaltlich Wissen zum Thema Objektorientierung in Standardsoftware, der zweite die Kenntnisse in Bezug auf HTML und CSS. Übungsbegleitende Lernerfolgskontrollen nahmen damit zu und wurden dichter auf die erste Phase verteilt: Nach der Hälfte der Übungen stand also entweder eine Abgabeaufgabe oder ein Online-Test.

3.2 Zweite Phase: Die „Flipped“ Vorlesung

In der zweiten Phase, welche sich inhaltlich mit Datenbanken befasste, wurde die Inhaltsvermittlung – welche für die traditionelle Vorlesung vorgesehen war – in Screencasts verlegt. Dazu wurden entsprechend der drei Vorlesungen auch drei Videos mit zugehörigen Onlinetests produziert und vor den zugehörigen Vorlesungszeiten, welche nun „Präsenzphasen“ genannt wurden, bereitgestellt. Die Lehrveranstaltung hatte nun eine wiederkehrende Gliederung in 4 Arbeitsschritte im 2-Wochenrhythmus, welche jedoch nicht mehr nach Vorlesung und Übung zu trennen waren.

[8] „Der Bildschirminhalt des Produzenten wird über eine entsprechende Software mitgeschnitten und als Video ausgegeben. Standardmäßig ist die Aufnahme eines Audiosignals parallel zur Bildaufnahme möglich.“ [Be12]

[9] Den Studierenden stand es frei, mit Microsoft- oder OpenOffice-Lösungen zu arbeiten, deshalb wurden für beide Lösungen Screencasts erstellt.

1. **Onlinephase I – Dozentenvortrag:** Die Präsentation des Dozenten ist im Video zu sehen, dazu spricht der Dozent.

2. **Onlinephase II – Test:** Ein Test bestehend aus verschiedenen Fragetypen, soll dem Studierenden den aktuellen Wissensstand aufzeigen und den erwarteten Kenntnisstand vor dem Besuch der Präsenzveranstaltung transparent machen.

3. **Präsenzphase I – Elaboration vormals „Vorlesung“:** Im Zeitrahmen der vorherigen Vorlesung wurden nun Kapazitäten für den Austausch zwischen Dozent und Studierenden sowie unter Studierenden frei. Inhaltliche Aspekte aus den Screencasts, aber auch Übungsaufgaben konnten nun diskutiert werden, dabei fanden aktivierende Methoden wie das Think-Pair-Share[10] oder Gruppendiskussionen statt. Die Inhalte der Screencasts wurden in der Präsenzphase nicht wiederholt sondern durch gute Mitarbeit der Studierenden aktiv elaboriert.

4. **Präsenzphase II – Projektarbeit vormals „Übung“:** Durch Bearbeitung und Diskussion von Übungsaufgaben in der Phase der Elaboration konnte im Zeitrahmen der Übung nun projektorientiert gearbeitet werden. Die Studierenden wählten dabei aus drei Szenarien, welche mit einer Datenbank abgebildet werden sollten, beispielsweise eine Krankenhausverwaltung.

Vereinzelte Probleme lösten die Studenten durch wiederholtes Rezipieren der Screencasts oder eigene Recherche. Gab es gehäufte Probleme, so wurden Theorieschübe (maximal 5 Minuten lang) durch den Übungsleiter oder Vorlesenden eingelegt. Die Projektarbeiten wurden in einer abschließenden Veranstaltung vorgestellt und diskutiert.

Online-Phase		Präsensphase	
OPAL		Elaboration	Übung
Video	Test	Aufgaben: Datenbankentwurf I	Planung im ER-Modell
			Projektarbeit I
Video	Test	Aufgaben: Datenbankentwurf II	Relationen-Modell, Überführung
			Projektarbeit II
Video	Test	Aufgaben: Abfragen und Berichte	Projektarbeit III
			Präsentation der Projekte

Tab. 2: Ablauf der Lehrveranstaltung im WS 2014/2015[11]

[10] Einzelarbeit – Teilen und Überdenken – Präsentieren
[11] eine Zelle entspricht einer 90-minütigen Lerneinheit, in den Präsenzphasen

4 Evaluation

4.1 Wahrnehmung und Kritik der Dozenten

Für die beteiligten Dozenten und Übungsleiter wurde schnell klar, dass die Zeit in den Präsenzveranstaltungen effektiver und individueller genutzt werden kann, da das „Vorlesen" der Inhalte entfiel und die Studenten inhaltlich gut vorbereitet zu den Lehrveranstaltungen erschienen. So war es möglich, sich während der Arbeitsphasen mit einzelnen Studierenden zu unterhalten und diese individuell zu unterstützen und zu fördern. Die Gruppendiskussionen als Diskurs zu den Themen der Screencasts und Übungsaufgaben wurden durchweg als fruchtbar eingeschätzt, freiwillige Meldungen sowie Einwände bereicherten die Präsenzphase für alle Teilnehmenden. Methodisch konnte so wesentlich freier als zuvor gearbeitet werden. Gegenüber dem Vorlesungskonzept wurden differenzierte didaktische Überlegungen möglich und notwendig.

4.2 Online-Befragung der Teilnehmer

In einer abschließenden Befragung der Lehrveranstaltungsteilnehmer[12] zeigte sich, dass der Großteil der Studierenden gut mit der Methode umgehen und lernen konnte. In der Befragung wählten die Studierenden je aus den Ausprägungen „*trifft nicht zu*", „*trifft eher nicht zu*", „*trifft eher zu*" und „*trifft zu*". Durch die Hinführung in der ersten Phase, in welcher bereits Screencasts und Tests verwendet wurden, gab es nur wenige Probleme unter den Studierenden.

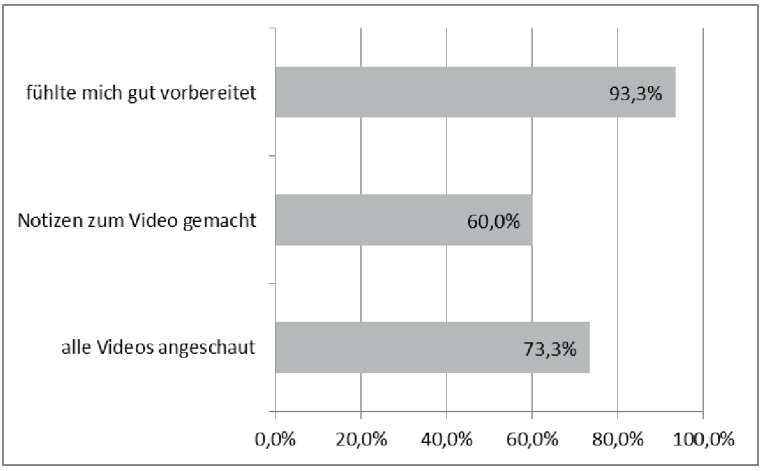


Abb. 2: Teilnehmerbefragung zur Online-Phase, Summe „*trifft eher zu*" und „*trifft zu*"

[12] Die Befragung wurde ebenfalls online mit OPAL durchgeführt; N=13

Im zweiten Teil der Befragung zeigt sich das Potenzial der Methode, die Mehrheit gab an, das „Lernen mit Screencasts“ falle ihnen leichter als „das Lernen in der traditionellen Vorlesung“.

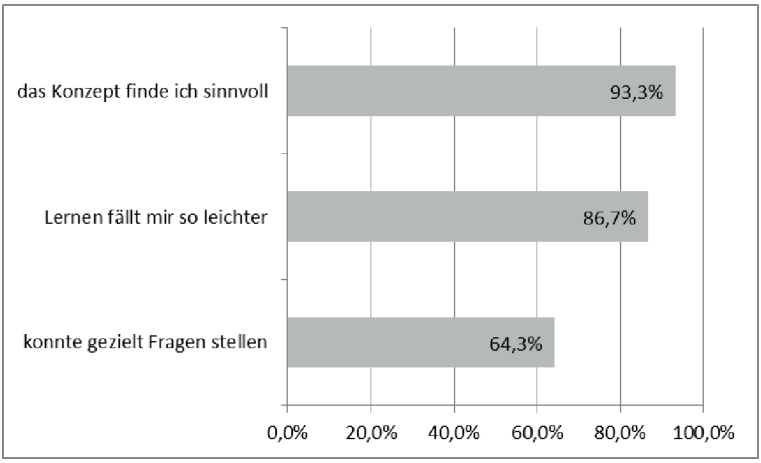


Abb. 3: Teilnehmerbefragung zur Online-Phase, Summe „*trifft eher zu*“ und „*trifft zu*“

Es ist bemerkenswert, dass sich die Lehramtsstudierenden mit der Methode identifizieren können und diese als „sinnvoll“ erachten. Die konsequente Metakommunikation zur Methode an zentralen Stellen war dabei sicherlich sinnvoll, um nicht zuletzt das Methodenrepertoire der Studierenden auch um die Methode Flipped Classroom zu erweitern.

In einer Einschätzung, welche die Studierenden im Freitext abgeben konnten, waren unter anderem folgende Kommentare zu lesen:

- *„Es ist ein echt tolles Konzept im Gegensatz zu dem überholten und langweiligen Konzept der Vorlesungen. Nicht nur in Informatik sollte dieses Konzept Einzug halten, sondern auch in allen anderen Vorlesungen.“*
- *„Das Konzept des Flipped Classroom finde ich sehr interessant und es macht mehr Spaß zu lernen, weil man sich zuhause mit den Inhalten befassen kann.“*
- *„Sonst finde ich das ganze Konzept echt super, obwohl ich zugeben muss, dass ich anfangs sehr skeptisch war.“*

Viele weitere Kommentare setzten sich vor allem mit der Umsetzung des Konzepts auseinander. Das direkte Feedback der Studenten war jedoch in großen Teilen konstruktiv formuliert, unter anderem forderten die Teilnehmer:

- kürzere, äußerlich strukturierte und inhaltlich getrennte Screencasts
- bessere Navigation in/zwischen den Screencasts

- klare Lösungen zu Tests, auch wenn Lösungen aus Vorlesungsmaterial oder Screencasts abgeleitet werden können und adaptive Fehlermeldungen ausgegeben werden
- Verbesserung der Steuerung in der Testumgebung (OPAL), transparentere Punkteverteilung

Sehr wenige Studenten lehnten die Methode gänzlich ab, da diese die Phase des Elaborierens als *„Kontrollen über eine eventuell nicht erledigte Vorbereitung der Studenten“* ansahen.

5 Fazit

Das Konzept des Flipped Classroom ließ sich ohne Probleme auch auf eine universitäre Veranstaltung übertragen. Die Aufmerksamkeit konnte vom Dozenten weg, hin zu den Studierenden und dem Lerngegenstand an sich gerichtet werden [Ru14]. Bemerkenswert ist die gewonnene Zeit, welche in Präsenzphasen für Diskurs und individuelle Probleme verwendet werden kann. Des Weiteren steigert die Verzahnung von Online- und Präsenzphase sichtlich die Vernetzung und Kooperation unter den Studierenden, da Zeitfenster für Diskussionen und sogar kleinere Projekte geschaffen werden.

Die Methode wurde vom Großteil der Studenten sehr gut angenommen, viele meinten sogar, deshalb besser gelernt zu haben. Die geäußerte Kritik der Studenten zur Umsetzung im Detail kann im größeren Zusammenhang als Erreichen eines weiteren Zieles gewertet werden: Die Auseinandersetzung von Lehramtsstudenten/innen mit modernen Lehr- und Lernmethoden ab dem ersten Fachsemester.

Literaturverzeichnis

[Be12] Becher, A. : Lernvideos auf YouTube (Masterarbeit), http://www3.sn.schule.de/fileadmin/_special/gruppen/40/MASTERARBEIT.pdf, 27.04.2015.

[Be13] Berendt, B.: Neues Handbuch Hochschullehre - Lehren und Lernen effizient gestalten. Raabe, Stuttgart, 2013.

[La14] Larcara, M.: „Benefits of the Flipped Classroom Model“. In: (Keengwe, Jared et al., Hrsg.) Promoting active learning through the flipped classroom model. IGI Global, Hershey, S. 132–144, 2014.

[Mo07] Studienordnung für das Fach Informatik im Lehramtsbezogenen Bachelor-Studiengang Allgemeinbildende Schulen: Modulbeschreibungen, https://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zlsb/studium/dokumente/abs/SO_Informatik_300708_ungeprft.pdf, 27.04.2015.

[On14] BPS Bildungsportal Sachsen GmbH: ONYX TESTSUITE – Funktionalität, https://www.bps-system.de/cms/fileadmin/Onyx/Funktionsbeschreibung_ONYX.pdf, 27.04.2015.

[Ru14] Rudolph, M.: Flipped Classroom in der Universität – ein didaktisches Konzept (Masterarbeit), http://www3.sn.schule.de/fileadmin/_special/benutzer/71/masterarbeit_flippedclassroom_online_version.pdf, 27.04.2015.

Lichtharfe - ein interdisziplinäres Unterrichtsprojekt

David Baumgärtel[1], Christopher Bednorz[2], Bastian Boger[3], Leonore Dietrich[4], Jan Hofmann[5], Hannes Koderisch[6], Anna Pössniker[7] und Oliver Schuppe[8]

Abstract: Dieser Beitrag stellt ein interdisziplinäres Unterrichtsprojekt aus dem Bereich Physical Computing vor, das auf Basis eines Arduinos eine elektronische Harfe realisiert. Durch seinen modularen Aufbau ist das Projekt vielfältig realisierbar und damit in die äußerst heterogene Landschaft des Informatik- und auch Naturwissenschaftlichen Unterrichts in Deutschland sehr gut integrierbar. Die Schülerinnen und Schüler erarbeiten sich vielfältige technische Kenntnisse, verknüpfen diese mit Programmierung und Gestaltung und lernen ein Informatiksystem als sichtbares, erlebbares System kennen, das sie selbst erschaffen und in der Schulgemeinschaft sichtbar gemacht haben.

Keywords: Arduino, Interdisziplinär, Projekt, STEAM

1 Einleitung

Bei einer Lichtharfe handelt es sich um ein rechnergesteuertes Instrument, das einer elektronischen Harfe ähnelt. Die Saiten werden durch Lichtschranken ersetzt, die vom Musiker unterbrochen werden. Der Ton wird elektronisch erzeugt, die Steuerung erfolgt über einen Arduino. Motiviert wurde dieses Projekt durch die Idee, fächerübergreifend zu arbeiten und künstlerische Aspekte mit einzubinden. Als Vorbilder dienten hier das Projekt Lichtharfe mit AVR ATMega16.3gp [Er11], das für eine Ausstellung entstand und das Projekt My Interactive Garden [PR13] als schulisches Physical Computing Projekt. Der Anreiz bestand darin, das zunächst sehr komplex erscheinende Lichtharfen-Projekt auch auf den Schulkontext abzubilden.

Beim Bau der Harfe erwerben Schüler grundlegende Kenntnisse zu elektronischen Schaltungen, die Programmierung des Arduinos bietet einen einfachen und schülergerechten Einstieg in die Arbeit mit Mikrocontrollern und bietet

[1] Universität Heidelberg, INF, 69120 Heidelberg, D.Baumgaertel@stud.uni-heidelberg.de
[2] Universität Heidelberg, INF, 69120 Heidelberg, Bednorz@stud.uni-heidelberg.de
[3] Universität Heidelberg, INF, 69120 Heidelberg, Boger@stud.uni-heidelberg.de
[4] Universität Heidelberg, Didaktik der Informatik, 69120 Heidelberg, Leonore.Dietrich@uni-heidelberg.de
[5] Universität Heidelberg, INF, 69120 Heidelberg, Hofmann@stud.uni-heidelberg.de
[6] Universität Heidelberg, INF, 69120 Heidelberg, Koderisch@stud.uni-heidelberg.de
[7] Universität Heidelberg, INF, 69120 Heidelberg, Poessniker@stud.uni-heidelberg.de
[8] Universität Heidelberg, INF, 69120 Heidelberg, Schuppe@stud.uni-heidelberg.de

Anknüpfungspunkte zu Fächern wie Physik und Naturwissenschaft/Technik: Es müssen Sensorwerte eingelesen, verarbeitet und passende Töne erzeugt werden. Ebenso spielen Design und Entwicklung des Instrumentes eine Rolle und legen eine Kooperation mit dem Musikunterricht sowie die Konstruktion des Gehäuses über handwerkliche Fähigkeiten eine Zusammenarbeit mit künstlerischen Fächern nahelegt.

Die Lichtharfe zeigt, wie sich verschiedene und oftmals getrennt behandelte Themen sowie einander scheinbar ferne Fachbereiche und heterogene Schülerinteressen verbinden lassen. Die Entwicklung der Hard- und Software ermöglicht, den Unterrichtsgegenstand sowohl geistig als auch haptisch zu "begreifen". Insbesondere der künstlerische Anteil motiviert auch eine neue Schülergruppe für informatische Themen [Ta10]. Der Einsatz des Instrumentes innerhalb schulischer AGs wie Theater oder Orchester macht Informatik sichtbar und bringt sie in einen schulgemeinschaftlichen Kontext, der eine hohe Motivation für die Schüler bedeutet.

2 Rahmenbedingungen

Das Gesamtprojekt lässt sich je nach den möglichen organisatorischen Rahmenbedingungen sehr gut modularisieren. Dies eröffnet eine Umsetzung im Rahmen einer AG, eines Projektes in einer Kompaktphase oder auch die Integration in den Fachunterricht eines oder sogar mehrerer kooperierender Fächer. Um diese Modularisierung angemessen zu dokumentieren, weisen die einzelnen Abschnitte entsprechende Vermerke zu Minimalanforderungen und optionalen Komponenten bzw. Umsetzungsschritten auf.

3 Fachliche Voraussetzungen

Das nötige Basiswissen für das Projekt Lichtharfe kann großenteils während der Bearbeitung erworben werden – in diesem Fall muss hierfür zusätzlich Zeit eingeräumt und enger angeleitet werden. Wir gehen von einer Schülergruppe am Ende der Mittelstufe aus, die die entsprechenden Fähigkeiten mitbringt, sodass nur eine Auffrischung der Kenntnisse notwendig ist. Daher werden die fachlichen Voraussetzungen für das eigentliche Projekt an dieser Stelle nur kurz erläutert. Da die Autoren des Beitrags in Baden-Württemberg unterrichten, wurden die hier gültigen Lehrpläne auf die Voraussetzungen hin untersucht und es wurde festgestellt, dass diese spätestens nach Klasse 8 bei allen Schülern vorhanden sein müssten.

3.1 Physik

Die Schülerinnen und Schüler benötigen für die Arbeit mit dem Arduino und die Konstruktion der Lichtharfe grundlegende Kenntnisse zu Schaltkreisen. Diese können beispielsweise im ersten Teil des Projekts – dem Bau eines Mini-Prototypen – erworben werden, sofern dieser dann enger angeleitet wird. Alternativ verweisen wir auf das Einstiegsmodul aus dem Projekt Informatik Enlightened [EKQ13], das alle wesentlichen Grundlagen zum Umgang mit Schaltkreisen behandelt.

Darüber hinaus werden geringe Kenntnisse aus dem Bereich der Optik vorausgesetzt – hier konkret der Umgang mit Linsen und deren Eigenschaften zur Bündelung von Licht. Die für das Projekt nötigen Kenntnisse sind mit einem Arbeitsblatt zur Brennweite vollumfassend gegeben und können gut auch innerhalb des Projekts ohne großen zusätzlichen Zeitaufwand erworben werden. In Baden-Württemberg sind diese Themen Inhalt des Physikunterrichts in Klasse 7.

3.2 Informatik

Für die Realisierung der Lichtharfe müssen Sensorwerte gelesen und ausgewertet, sowie auf diese mithilfe von Verzweigungen reagiert werden. Die grundlegenden Konzepte einer Wiederholung, Verzweigung, im fortgeschrittenen Modul auch switch/case Anweisungen sowie der Umgang mit Bibliotheken sind notwendig. Ein Verständnis für digitale und analoge Werte sowie ggf. deren Umrechnung sind ebenso hilfreich wie Erfahrungen im Umgang mit Variablen. Vorerfahrungen in Form eines Einstiegs in Programmierung sind daher von Vorteil, allerdings können sämtliche Programmierkenntnisse in der Arduino Umgebung wenn nötig ebenfalls im Rahmen der Einführungsstation des Projekts Informatik Enlightened [EKQ13] mit einem Zeitaufwand von etwa einer Doppelstunde erworben werden. Je nach Lerngruppe wird der Betreuungsaufwand dann gegebenenfalls größer oder eine weitere Station als Übungsprojekt zwischengeschaltet. Je geringer das Vorwissen im Programmierbereich ist, desto wichtiger werden Hilfestellungen wie Beispielcodes, Arbeitsblätter und erläuternde Unterrichtsbestandteile.

3.3 Konstruktive Fähigkeiten

Insbesondere für den Bau des Gehäuses werden auch handwerkliche Fähigkeiten benötigt. Hierbei können Zuschnitte natürlich auch direkt bei der Materialbeschaffung passend erfolgen, zumindest Bohren in Holz, Einsenken bzw. Versäubern von Bohrlöchern sowie Schrauben und Leimen müssen aber in Eigenarbeit erfolgen. Hier bietet sich eine Kooperation beispielsweise mit dem Werk- oder Kunstunterricht an. Je nach Bundesland haben die Schüler auch im Rahmen von Querschnittsfächern bereits Erfahrungen gesammelt – in Baden-Württemberg beispielsweise während ihres NWT-

Projekts in Klasse 8.

4 Organisatorische Bedingungen

Das Projekt ist an unterschiedlichste Organisationsformen anpassbar. So kann es mit einer kleineren Schülergruppe in einer individuellen Lernphase, mit einer AG im Ergänzungsbereich oder auch mit einer ganzen Lerngruppe im Regelunterricht bis hin zu einer Projektwoche mit fächer- und jahrgangsübergreifenden Konzepten erarbeitet werden.

Sämtliche Materialien wurden online bestellt. Da Bauteile sowohl aus dem optischen als auch aus dem Elektronikbereich benötigt werden, sind zwei Bezugsquellen nötig. Beide in unserem Projekt angesprochenen Lieferanten[9] versenden an Bildungseinrichtungen auf Rechnung, sodass problemlos über die Schule bestellt werden kann. Fachliche Nachfragen wurden schnell und kompetent beantwortet, sodass die Beschaffung problemlos verlief. Diese Hürde ist in Schulen oft ein Hinderungsgrund für interessante Projekte und ein kompetenter Lieferant somit wichtige Voraussetzung zur Realisierung des Projekts.

Die Kosten verteilen sich zum einen auf den Arduino und seine Umgebung (Netzteil, Steckverbindungen, Widerstände), zum anderen auf Linsen, Sensoren und Dioden sowie Holz für den Bau des Gehäuses. Insgesamt kommen Kosten von momentan ca. € 100,- für eine Lichtharfe zusammen, sofern alle Bauteile gekauft werden. Sind Arduino und Holzreste vorhanden, reduziert sich der Bedarf um etwa die Hälfte.

5 Projektdurchführung

Das Projekt ist modular aufgebaut und kann unterschiedlich schnell und umfangreich umgesetzt werden. Die einzelnen Module werden im Folgenden beschrieben. Die Minimalkonfiguration kann als Folge Miniprotoyp – Lichtprototyp – Music-Shield-Version – Harfe umgesetzt werden, eine komplette Durchführung aller Schritte ermöglicht auch die Erarbeitung grundlegender Kompetenzen im Umgang mit Arduino und weiteren Bauteilen und kann so fehlendes Vorwissen ergänzen.

5.1 Miniprototyp

Der Bau eines Miniprototypen besteht aus einer einzelnen Lichtsaite. Ein

[9] Optik: astromedia.de / Elektronik, Arduino: watterott.com

Fotowiderstand liefert ein Signal als Input, das wiederum eine LED oder einen Piezo-Signalgeber über einen digitalen Ausgang ansteuert. Umgesetzt wird also die Basisfunktionalität einer Harfensaite. Damit wird in diesem ersten Schritt bereits ein prototypischer Teil des Endprodukts erstellt.

Der Mini-Prototyp ist eine Möglichkeit, im Anfangsprojekt einzelne konstruierte Saiten im weiteren Verlauf zum vollständigen Instrument zu kombinieren. Dieses Vorgehen bietet den Vorteil, dass alle Schüler zunächst Zeit haben, sich individuell oder in Partnerarbeit mit Hard- und Software auseinanderzusetzen. Ergebnisse können vorgestellt, verglichen und damit eine einheitliche Ergebnissicherung durchgeführt werden. Im weiteren Verlauf kann dann der beste Code übernommen werden, wodurch ein Wissenstransfer stattfindet.

5.2 Miniprototyp 2*

Im zweiten Schritt kann nach dem Modell der wachsenden Gruppe eine kleine dreisaitige Harfe entstehen. Drei Gruppen vereinen ihre Saiten aus den Miniprototypen zu einem Aufbau und passen dann – entweder gemeinsam oder als drei alternative, parallel entwickelte Lösungen – den Code entsprechend an.

Der Prototyp 2 muss zwecks Unterscheidbarkeit der Saiten mit LEDs betrieben oder aber das Piezo-Signal entsprechend unterscheidbar angesteuert werden. Die Schüler müssen sich hier also Gedanken machen, wie sie die Korrektheit ihres Codes sinnvoll testen können. Der Einsatz des Music-Shields ist an dieser Stelle noch nicht zielführend, da mit der Einbindung des Bauteils auch Bibliotheken notwendig werden, was für die Schüler zu diesem Zeitpunkt noch eine sehr große Herausforderung darstellt.

5.3 Licht-Prototyp

Den Miniprototyp kann man nun auf die gewünschte Anzahl an Saiten ausbauen und als vollständige Harfe betreiben. Die Programmierung muss im Idealfall nicht großartig angepasst werden – dieser Schritt ist aber auch für die Erkenntnis, dass unterschiedliche Implementierungen zu unterschiedlich viel Aufwand bei Anpassungen führen zentral.

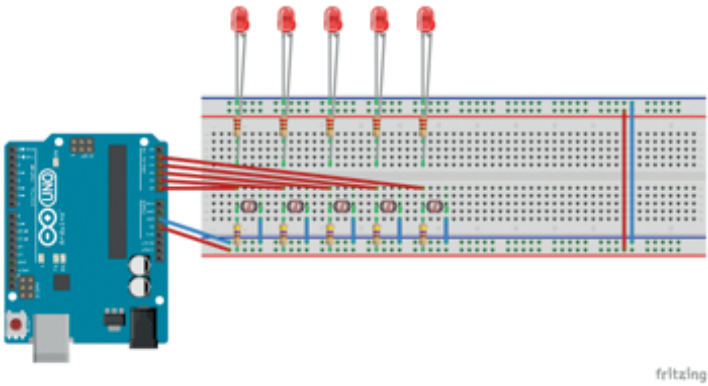


Abb. 1: Übersicht Licht-Prototyp

Mit dem Bau vieler Saiten wird je nach Grüße des verwendeten Breadboards auch der Umstieg auf eine größere Konstruktion sowie eine alternative Verkabelung nötig. Insgesamt kann aber auch diese Phase bei ausreichend großem Breadboard noch ohne die spätere Harfenkonstruktion auskommen.

5.4 Music-Shield ansteuern

Um ein musikalisch ansprechendes Resultat zu erzeugen, muss für die Harfe ein Weg gefunden werden, um Töne als Midi-Signal auszugeben. Hierfür kann ein Music-Shield[10] (vgl. Abb. 2) verwendet werden, das mit allen Arduino Boards zusammenarbeitet. Für den ersten Kontakt mit dem Music-Shield ist eine Version des Miniprototypen aus dem Projektbeginn äußerst hilfreich. Die Komplexität wird so auf das neu einzubindende Bauteil begrenzt und der Fokus auf den Umgang mit der nötigen Bibliothek und die Funktion als Tonausgeber gelegt. Die Schülerinnen und Schüler erzeugen so mit einer Einzelsaite einen Ton unter Verwendung des Music-Shields und entwickeln eine Teillösung, die sie später in den Code ihres Licht-Prototypen übertragen können.

Testläufe mit unterschiedlichen Reaktions- bzw. Wartezeiten der Sensoren sind sinnvoll, um ein bestmögliches Klangergebnis zu erreichen und sollten ggf. von der Lehrkraft angeregt werden. Hierbei sind auch Vorstellungen von Gruppenergebnissen zum gegenseitigen Austausch wichtig. Ideen zur Lautstärke- und Anschlagsdynamik kommen in dieser Phase von den Schülerinnen und Schülern und sollten für die spätere Optimierungsphase festgehalten werden.

[10] http://www.watterott.com/de/Music-Shield

Sollte kein Miniprototyp mehr verfügbar sein, kann auch schlicht nur ein Sensor des Licht-Prototypen verwendet werden.

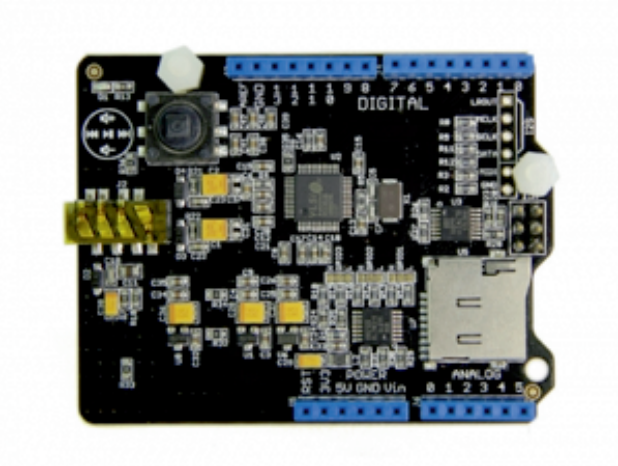


Abb. 2: Music Shield für den Arduino

5.5 Die Lichtharfe

Der nun entwickelte Code muss in die vollständige Harfe übertragen werden. Das Vorgehen sollte wiederum mit den Schülerinnen und Schülern reflektiert werden, da aber alle Voraussetzungen zu diesem Zeitpunkt bereits vorhanden sind, stellt dieser Schritt im Wesentlichen eine organisatorische Aufgabe dar.

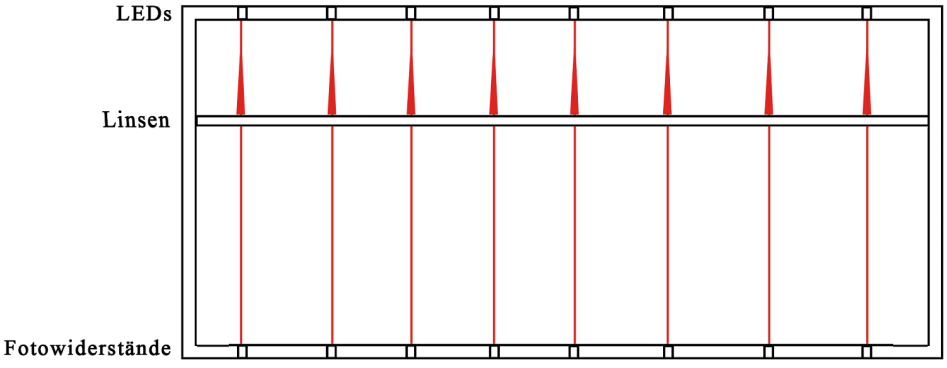


Abb. 3: Aufbau der Lichtharfe

Gegebenenfalls bietet es sich – je nach Gruppengröße – an, diesen Schritt nur von einem Teil der Gruppe umsetzen zu lassen. Zeitgleich wäre für die anderen Schülerinnen und Schüler bereits die Arbeit am Gehäuse der Lichtharfe möglich. Hier sind – bis auf die Grenzen die die optischen Eckdaten der Linsen und Lichtquellen darstellen – wenig Vorgaben nötig. Solange die Brennweite der Linsen berücksichtigt wird, kann das Gehäuse in Form eines kleinen Rechtecks, über die klassische Harfe, bis hin zu einzelnen Saitenmodulen, bestehend aus Lichtquelle, Linse und Fotowiderstand, die in einzelnen Boxen räumlich verteilt aufgebaut werden, konstruiert werden. So kann beispielsweise entlang einer Treppe auf jeder Stufe eine eigene Saite aufgebaut und die Harfe so im Schulhaus präsentiert werden.

5.6 Ausblick / Verfeinerung

Die Lichtharfe stellt ein Basisprojekt dar, mit dem bereits ansprechend musiziert werden kann. Es sind jedoch zahlreiche Verfeinerungen denkbar und auch inhaltliche Anpassungen oder Erweiterungen für fächerübergreifenden Unterricht bieten sich an.

So ist Anschlagsdynamik eine anspruchsvolle Erweiterung, die über den Umstieg auf analoge Sensorwerte erreicht werden kann. Wird der Lichtstrahl zwischen Quelle und Linse nur langsam unterbrochen, könnte die Harfe aus der Veränderung der Lichtintensität einen weichen Anschlag generieren und so natürlicher klingen.

Die Anpassung der Lautstärke durch partielle Abdeckung der Lichtquelle setzt ebenfalls ein analoges Eingangssignal voraus. Die Umsetzung setzt eine ausreichend große Brennweite der verwendeten Linsen voraus, damit zwischen Lichtquelle und Linse 'gespielt' werden kann.

Eine weitere Variante mit sehr hohem Aufforderungscharakter ist der Einbau einer Wahlmöglichkeit für das aktuelle Instrument.

6 Technische Umsetzung

Verwendet wurden helle LEDs mit Vorwiderständen, Sammellinsen, Fotowiderstände und Widerstände für Spannungsteiler sowie ein Arduino mit MusicShield 2.0 (Seeedstudio). Alternativ kann das Projekt auch mit einem Raspberry Pi umgesetzt werden. Zusätzlich werden eine Steckplatine, sowie Steckkabel und Verlängerungsstücke benötigt. Als Peripherie wurden PC-Lautsprecher verwendet, ein Anschluss an ein Mischpult und Ausgabe darüber, beispielsweise im Theaterkontext, ist natürlich ebenso möglich. Das Gehäuse für den Prototypen wurde mit Holz gebaut – andere Werkstoffe mit ausreichender Stabilität und einfacher Verarbeitung sind genauso verwendbar.

Abbildung Abb. 4 zeigt den Schaltplan der fertigen Lichtharfe, vereinfacht auf zunächst fünf Saiten.

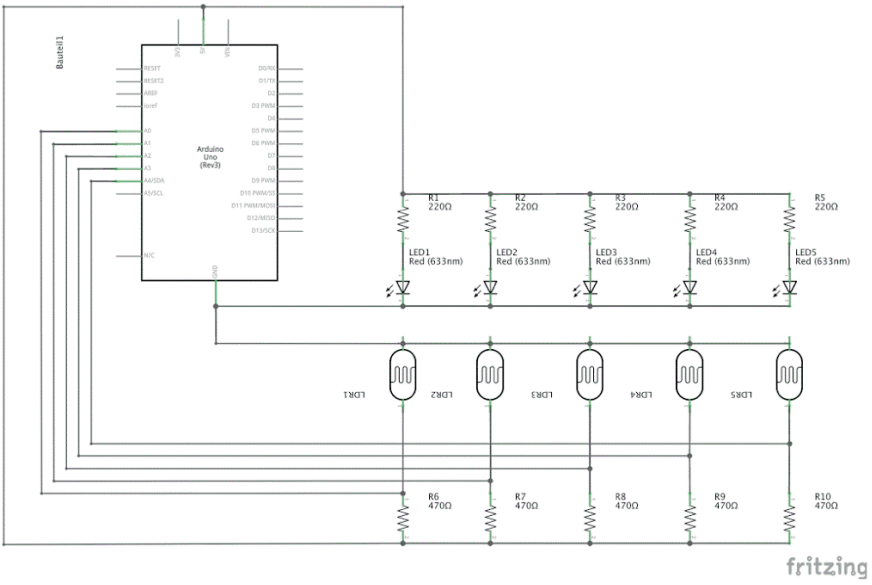


Abb. 4: Schaltplan

7 Erfahrungen

Das Projekt wurde prototypisch umgesetzt und befindet sich in der weiteren Erprobung. Lehrermaterial zum Projekt mit Beispielcode, Schalt- und Bauplänen ist in Arbeit und wird bis zur Veröffentlichung fertiggestellt.

Erste Anpassungen betreffen die Materialauswahl, die von Lasern auf LEDs umgestellt wurde, um eine Gefahrenbeurteilung zu vermeiden, die in vielen Bundesländern verpflichtend ist und zu einer eingeschränkten Nutzbarkeit geführt hätte. Die LED-Variante erwies sich als umsetzbar und wird nun als Standard übernommen.

Weitere Änderungen gegenüber dem Prototypen sind im Gesamtaufbau geplant. Umgesetzt werden unterschiedliche Aufbauten vom kompakten Instrument bis zur Treppenharfe, die durch mehrere Akteure gespielt wird.

Die Lichtharfe zeigt sich als hoch motivierendes Projekt mit großer Selbstwirksamkeit. Auch die gemeinschaftliche Arbeit an einem Informatikprojekt, das im Schulkontext wahrgenommen wird, stellt für Schüler einen hohen Aufforderungscharakter dar und

regt diese zu überdurchschnittlichem Engagement an.

Nicht zuletzt trägt ein solches Projekt zur Sichtbarkeit der Informatik und zu einem differenzierteren Bild des Faches in der Schulgemeinschaft bei.

Literaturverzeichnis

[EKQ13] Ehlenz, M.; Kühn, H.; Quix, T.: Informatik enlightened - Was Blumen, Autos und Solarzellen verbindet. Aachen, 2013. http://schuelerlabor.informatik.rwth-aachen.de/modul/informatik-enlightened-was-blumen-autos-und-solarzellen-verbindet, Stand 15.3.2015

[Er11] Ermer, Christoph: Lichtharfe mit AVR ATMega16.3gp. http://homepages.uni-regensburg.de/~erc24492/AVR-Midi/AVR-Midi.html, Stand 30.4.2015

[Lu15] Lukas, Mario: Lightharp. http://www.mariolukas.de/, Stand: 25.4.2015.

[PR13] Przybylla, Mareen; Romeike, Ralf: Physical Computing im Informatikunterricht. In (Breier, Norbert; Stechert, Peer; Wilke, Thomas Hrsg.): Informatik erweitert Horizonte. INFOS 2013. LNI 219, Bonn S. 137-146, 2013.

[Ta10] Tarnoff, John: STEM to STEAM – Recognizing the Value of Creative Skills in the Competitive Debate. http://steam-notstem.com/articles/stem-to-steam/, Stand 25.4.2015.

Datenschutz im 21. Jahrhundert – Ist Schutz der Privatsphäre (noch) möglich?

Bettina Berendt[1], Gebhard Dettmar[2], Bernhard Esslinger[3], Andreas Gramm[4], Andreas Grillenberger[5], Alexander Hug[6] und Helmut Witten[7]

Abstract: Die Bedrohung der Privatsphäre hat im 21. Jahrhundert im Wesentlichen zwei Dimensionen: zum einen die Datensammelindustrie (Facebook, Google & Co.) zusammen mit der „freiwilligen" Veröffentlichung personenbezogener Daten der Internet-Nutzer, zum anderen die anlasslose Massenüberwachung durch die Geheimdienste. Im Workshop sollen unterschiedliche Ansätze zur Behandlung des Privacy-Diskurses im Unterricht der Sekundarstufen I und II zur Diskussion gestellt werden. Besondere Aktualität hat das Thema durch die z. Zt. wieder von interessierten politischen Akteuren geforderte Möglichkeit gewonnen, auch verschlüsselte Informationen mitlesen zu können (Crypto Wars 2.0 bzw. 3.0). Auf der anderen Seite gibt es z. B. den Versuch, durch Massenklagen Facebook zu zwingen, sich an die europäischen Normen des Datenschutzes zu halten.

Keywords: Datenschutz, Privacy bzw. Privatsphäre, Datensammelindustrie, Ausspähung durch Geheimdienste, informatische Bildung, fächerübergreifender Unterricht

1 Einleitung

Es steht schlecht um den Datenschutz in Deutschland. Jochen Koubek[8] fasst die Situation folgendermaßen zusammen:

> Es gibt ein historisch gewachsenes und sowohl durch Rechtsnormen ausformuliertes als auch durch die Urteilspraxis ausgeformtes Persönlichkeitsrecht, das insbesondere den Umgang mit personenbezogenen Daten umfasst. Wir haben Firmen, denen dieses Recht gleichgültig ist, weil sie in ihrer Selbstwahrnehmung gar nicht im deutschen Rechtsraum operieren. Wir haben eine Regierung, der dieses Recht gleichgültig ist, wenn es um die Wahrung ihrer

[1] KU Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgien, bberendt@gmx.net
[2] BSB Hamburg, B 52-4, Moorkamp 7-9, 20357 Hamburg, g.dettmar@web.de
[3] Universität Siegen, IT-Sicherheit und Kryptologie, bernhard.esslinger@uni-siegen.de
[4] Gymnasium Tiergarten, Altonaer Straße 26, 10555 Berlin, gramm@gymnasium-tiergarten.de
[5] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, Martensstr. 3, 91058 Erlangen andreas.grillenberger@fau.de
[6] Universität Koblenz-Landau, Fachbereich Informatik, Universitätsstraße 1, 56070 Koblenz, hug@uni-koblenz.de
[7] GI-Fachgruppe IBBB (Informatik-Bildung in Berlin und Brandenburg),Brandenburgische Str.23, 10707 Berlin, helmut@witten-berlin.de
[8] (Ko14), s. a. Vortrag „Datenschutz und Persönlichkeitsrechte" vom 23.9.2014: http://medienwissenschaft.uni-bayreuth.de/assets/Uploads/Koubek/forschung/KoubekDatenschutzPersoenlichkeitsrechtePreprint.pdf.

> eigenen Interessen geht. Und wir haben Bürger, denen dieses Recht ebenfalls gleichgültig ist, wenn es mit Einschränkungen der persönlichen, digitalen Lebensgestaltung einhergeht oder wenn zu einer Durchsetzung politische Aktivierung erforderlich wäre.

Im Volkszählungsurteil von 1983 hat das Bundesverfassungsgericht aus dem allgemeinen Persönlichkeitsrecht (ein Grundrecht, das sich aus Art. 1, Absatz 1 GG in Verbindung mit der freien Entfaltung der Persönlichkeit aus Art. 2 ergibt) das Recht auf ***informationelle Selbstbestimmung*** abgeleitet (s. z. B. [Ko14]).

Wir stellen im folgenden drei Unterrichtsreihen vor, die informatische mit politischer Bildung auf eine Weise kombinieren, die es Schülerinnen und Schülern ermöglicht, die Auswertungsmechanismen der Datensammelindustrie nachzuvollziehen und die daraus resultierenden Konsequenzen für sie als Grundrechtssubjekte zu erkennen, um daraus mögliche Handlungskonzepte und -strategien zu entwickeln.

Dazu ist es erforderlich, den Lernenden zu vermitteln, warum die unbegrenzte Weitergabe persönlicher Daten überhaupt ein Problem darstellt und wie man sich mit den heute zur Verfügung stehenden Mitteln schützen kann. Mit welchen Argumenten man den bekannten Einwände „Ich habe doch nichts zu verbergen“ etc. begegnen kann, behandeln wir im folgenden Abschnitt.

2 Erfahrungen

Das Thema Datenschutz in all seinen Facetten ist ein Unterrichtsthema, das ein fester Bestandteil im Informatikcurriculum ist. Im Folgenden werden einige gelungene Reihen zu diesem Thema kurz vorgestellt.

2.1 Spuren im Internet – Das „Planspiel Datenschutz 2.0 – Wer weiß was über dich im Internet“

Schon Ende der 80er, Anfang der 90er Jahre gab es die ersten Vorläufer des kontextorientierten Unterrichtsprojekts „Planspiel Datenschutz 2.0“ [IniK], einem Rollenspiel, in dem es darum geht, Themen und Zusammenhänge des Datenschutzes zu erarbeiten. Ausgangspunkt des Rollenspiels ist die Aussage „Ich habe nichts zu verbergen!“, um dann am Ende des Spiels die Schüler zu überzeugen, dass aus ihren Informationen leicht falsche Rückschlüsse u. Ä. gezogen werden können.

Während die ursprüngliche Version eine Fassung ist, die mit Papier und Bleistift arbeitet, so ist die Version 2.0 eine von Frank Oppermann und Alexander Dietz weiterentwickelte Datenbank-gestützte Alternative. Nachdem die Schüler in einer ersten Phase des Spiels auf Basis einer ihnen zugewiesenen Rollenbeschreibung Handlungen im Netz durchgeführt haben, die wiederum von einem „fiktiven“ Provider, den der Lehrer zuvor eingerichtet hat, protokolliert werden, wechseln die Lernenden in der zweiten Spielphase die Rolle und ermitteln das Verhalten eines anderen Mitspielers. Dabei erstellen sie ein

Profil aus den protokollierten Daten des Providers, um z. B. eine möglicherweise begangene Straftat aufzuklären. In der folgenden Vertiefungs- und Vernetzungsphase werden im Rahmen einer Gruppenarbeit Themen wie „informationelle Selbstbestimmung“, „Cybermobbing“, „Datenschutzgesetz“, … behandelt.

Einer der Autoren hat im Rahmen der fachdidaktischen Ausbildung mit den Studierenden das Planspiel in einem GK Informatik der Stufe 11 an einem Gymnasium im Februar 2015 durchgeführt, wobei jeweils einer der Studierenden die Lehrerrolle innehatte. Am Ende der Reihe wurden die Schüler gefragt, inwiefern sie aufgrund des neu erworbenen Wissens und der neuen Erkenntnisse ihr Verhalten ändern werden. Auch wenn vereinzelte Schüler kritische Anmerkungen machten und nach der Stunde weitergehende Fragen z. B. nach Tools stellten, zeigte die Mehrheit doch ein Verhalten, wie es schon in [Be14b] beobachtet wurde: Da sie ja noch jung seien, über kein Girokonto verfügen und keine Terroristen seien, hätten sie nichts zu verbergen; weder ihr Verhalten würden sie ändern noch irgendwelche Werkzeuge (Browser-Plug-Ins, kryptographische Hilfsmittel, …) nun nutzen.

2.2 „E-Mail (nur?) für Dich“ revisited: Sicher chatten statt mailen mit PGP?

Mit der Unterrichtsreihe „E-Mail (nur?) für Dich“[9] schlagen [GHW11] vor, Möglichkeiten des Kompromittierens unverschlüsselten E-Mail-Verkehrs in einer geschützten Umgebung im Unterricht erlebbar zu machen und so die Erarbeitung verschiedener kryptographischer Verfahren von Caesar bis zu RSA zu motivieren, um am Ende E-Mails ganz praktisch mit PGP (bzw. enigmail) zu verschlüsseln und zu signieren.

Einer der Hauptkritikpunkte an der Unterrichtsreihe ist, dass viele Jugendliche zwar E-Mail-Benutzerkonten haben, diese aber kaum noch für ihre eigentliche Kommunikation mit Freunden nutzen. Hinweise auf mögliche spätere Mailnutzung im beruflichen Umfeld und der Gefahr der Industriespionage werden zwar akzeptiert, aber als „weit weg“ vom persönlichen Leben empfunden. Darüber hinaus werden z. B. die Risiken des „Web of Trust“ als Infrastrukturelement von PGP kritisch diskutiert[10].

Wir schlagen daher vor, Möglichkeiten einer vertraulichen und verlässlichen Kommunikation zu thematisieren, die der tatsächlichen Kommunikation von Jugendlichen so nahe kommen, dass sie sie als ernsthafte Alternative zu bisher genutzten Angeboten erkennen. Hier haben sich verschiedene Anbieter wie etwa *TextSecure*, *Cryptocat*, *Silent Text* oder *Threema* etabliert. Die *Electronic Frontier Foundation (EFF)* hat eine Liste solcher Kommunikations-Anwendungen erstellt[11], in der sie diese hinsichtlich verschiedener Kriterien wie Ende-zu-Ende-Verschlüsselung, Authentifikation von Kommunikations-

[9] http://informatik-im-kontext.de/index.php/entwuerfe/email-nur-fuer-dich/
[10] vgl. z. B. Padmos, Arne: Why is GPG "damn near unusable"? [31c3]: https://www.youtube.com/watch?v=4gz9TBt-DAQ
[11] https://www.eff.org/secure-messaging-scorecard

partnern oder der Nachvollziehbarkeit des Quellcodes untersucht. Für den Unterricht bietet sich vor allem das *Off-the-Record (OTR) Messaging*[12] an, das auf dem auch als *Jabber* bekannt gewordenen *Extensible Messaging and Presence Protocol (XMPP)* aufsetzt und von verschiedenen Client-Anwendungen unterstützt wird[13].

Im Bereich von Desktop-Anwendungen ist *Pidgin*[14] ein weit verbreiteter Client. Der SPIEGEL hat eine schrittweise Anleitung zum verschlüsselten Chatten mit OTR/XMPP mit Pidgin veröffentlicht[15]. Damit Chatten mit OTR aber zu einer tatsächlichen Alternative zu WhatsApp & Co. wird, sollten mobile Implementierungen im Unterricht vorgestellt werden, wie z. B. *Xabber*[16] für Android-Smartphones. Den Netzwerkverkehr eines Smartphones zu verfolgen ist etwas aufwendiger als auf einem PC. Zwar gibt es entsprechende Apps, wie z. B. das *Wireshark*-Pendant *Shark for Root*[17], doch erfordern diese Apps meist Root-Rechte und sind deutlich weniger intuitiv zu bedienen als entsprechende Versionen für einen PC. Bis hier einfach zu bedienende Anwendungen zur Verfügung stehen, schlagen wir vor, die Grundlagen der Verschlüsselung und Authentifizierung von Kommunikationsteilnehmern mit E-Mail zu erarbeiten, um dann aufzuzeigen, welche Möglichkeiten es gibt, diese Anforderungen auch für einen Chat mit Instant Messaging Apps auf mobilen Endgeräten zu erfüllen.

2.3 Zur Bedeutung der Privatsphäre in Zeiten der Datensammelindustrie

Diese in [Be14b] genauer beschriebene Unterrichtsreihe behandelt fächerübergreifend die Auswirkungen des Trackings im Internet und der Datenauswertung der Datensammelindustrie, Facebook, Google & Co. auf eine staatliche Ordnung, die das Recht auf freie Persönlichkeitsentfaltung in das Zentrum ihrer Wertordnung stellt[18].

Die Reihe befasst sich explizit *nicht* mit dem, was im Vordergrund vieler Unterrichtsempfehlungen zur „Kompetenz in sozialen Netzwerken und im Internet“ steht [z. B. Kl13]: Welche Fotos stellt man online, wie viel postet man und an welchen Em-

[12] https://otr.cypherpunks.ca

[13] Die Eigenschaften von OTR wurden kürzlich gut nachvollziehbar beschrieben in Lautebach, Urs: „Neee, das hab ich nie gesagt! Das Chatprotokoll Off-the-Record (OTR)“ in LOG IN Heft 181 (2015), im Druck

[14] https://pidgin.im

[15] http://www.spiegel.de/netzwelt/netzpolitik/mit-jabber-pidgin-und-otr-so-chatten-sie-verschluesselt-a-912957.html (29. April 2015) Als Jabber-Server sollte man jabber.de verwenden, nicht den jabber-Server vom CCC, der im Spiegel-Artikel empfohlen wurde und danach so überlastet war, dass seitdem keine Neuanmeldungen mehr entgegen genommen werden.

[16] http://www.xabber.org

[17] https://play.google.com/store/apps/details?id=lv.n3o.shark

[18] Materialien und weitere Links finden sich auf http://people.cs.kuleuven.be/~bettina.berendt/Privacybildung/ (29. April 2015). Zu Erweiterungen von Literaturbasis und curricularen Vorschlägen s. [Be14a]. Eine interessante Alternative sind die von verschiedenen öffentlich-rechtlichen Fernsehsendern (u. a. Arte) erarbeiteten interaktiven Selbstlernmaterialien zum Themenkomplex „Tracking“: https://donottrack-doc.com/de/. Eine Zusammenfassung der Hauptaussagen dieser Materialien findet sich in dem Film http://www.ardmediathek.de/tv/Do-Not-Track/Do-Not-Track-Internet-Tracking-das-/Bayerisches-Fernsehen/Video?documentId=29004966&bcastId=29004948

pfängerkreis? So berechtigt diese Fragen und so wichtig diese Kompetenzen auch sind: Es geht nicht nur um die Entscheidungen des Einzelnen, wie viel er oder sie durch bewusste und rational fundierte Entscheidungen von sich „preisgibt". Vielmehr geht es darum, zu zeigen, dass eine bewusste Datensparsamkeit zwar nützlich sein kann (z. B. kann man sich entscheiden, nichts zu „liken", Anonymisierungsdienste zur Reduktion des Trackings nutzen, oder seine Kommunikation verschlüsseln), dass dies aber nur punktuell wirkt. Wenn die Nicht-Nutzung sämtlicher Kommunikations- und Informationsmedien *keine* Option ist, dann kann der *Einzelne* nur begrenzt effektiv handeln. In einer vernetzten digitalen Welt müssen Grundrechte auch gesetzlich effektiv geschützt werden. Aber Bürger müssen dies auch einfordern, und dazu müssen sie sich sowohl der Realitäten von Datensammlung und -nutzung als auch ihrer Rechte bewusst sein. Dieses Bewusstmachen durch entsprechende Wissensvermittlung ist unser Ziel.

Zunächst wird in der Reihe mit der Hilfe eines Browser-Plugins visualisiert, wie „Tracker" (z. B. Cookies) ohne unser Zutun und seitenübergreifend aufzeichnen, was wir tun, auch wenn wir „einfach nur" im Netz unterwegs sind, selbst ohne überhaupt etwas zu posten. Anschließend wird erklärt, wie mit Hilfe von Data Mining aus solchen Verhaltensdaten, ggf. kombiniert mit Transaktions- und anderen Daten, Zusammenhänge und Vorhersagen abgeleitet werden. Ein (echtes) Beispiel ist die von einem Kreditkartenunternehmen gefundene Korrelation, dass Menschen, die in Gitarrengeschäften einkaufen, weniger kreditwürdig sind. Ein weiteres (ebenfalls echtes) Beispiel ist die Vorhersage von Persönlichkeitseigenschaften aus Facebook-Likes: Wer Converse-Schuhe „liket", der gilt als dumm.

Diese Zusammenhänge sind rein korrelativ. Damit entfällt die Grundannahme der in anderen Unterrichtsplänen und Materialien zum Thema „Privatsphäre" so gern genannten Verhaltensregeln, die von (zumindest sozial) kausalen Zusammenhängen ausgehen: *Weil* jemand, der viel trinkt, wahrscheinlich kein zuverlässiger Arbeitnehmer ist, wird er schlechter einen Job bekommen. *Daher* ist es sinnvoll, keine Partyfotos für die Öffentlichkeit zu posten; bzw. wenn man das *nicht* tut, dann erscheint man auch als zuverlässig. In einer Welt, in der man aufgrund intransparenter Korrelationen (statt aufgrund bekannter und erlernbarer sozialer „kausaler" Regeln) als guter oder schlechter Kunde oder Arbeitnehmer (etc.) erscheint, gibt es aber keine Möglichkeit mehr, sich richtig oder falsch zu verhalten.

Anhand von Assoziationsregeln und einem Basisverfahren zum Lernen solcher Regeln wird in der Reihe dann die algorithmische Basis der Schlussfolgerungen im Data Mining illustriert[19]. Warum solche Schlussfolgerungen getroffen werden und warum sie problematisch sind, wird im Rollenspiel zwischen Nutzern mit Interesse an Umsonst-Diensten versus Unternehmen mit Interesse an vermarktbaren Daten erarbeitet. Hierbei werden Daten nicht nur für personalisierte Werbung vermarktet, sondern auch z. B. zur

[19] Grillenberger und Romeike [Gr15] merken zu Recht an, dass man bei der Darstellung und Erarbeitung von „Big-Data-Analyseverfahren" wie Assoziationsregeln und Clustering mit geeigneten Tools arbeiten sollte, die diese Algorithmen gut nachvollziehbar machen, und stellen hierfür gute Beispiele (Snap!) zur Verfügung.

Kundensegmentierung von Kreditkartenunternehmen oder Vorauswahl in Bewerbungsverfahren. Aus Datenanalysen kann somit soziale Ausgrenzung entstehen.

Wenn aber für den Bürger intransparent ist, wie er sich „richtig" verhalten kann, um nicht als Kreditrisiko, unzuverlässiger oder dummer Arbeitnehmer etc. zu gelten, dann besteht die Gefahr, dass er gar nichts mehr sagt und tut und auch sonstige demokratische Rechte nicht mehr wahrnimmt. (Eine detaillierte Analyse solcher „Chilling-Effekte" findet sich in [So11].) Mit anderen Worten: wenn man nicht weiß, was wer wie mit den eigenen persönlichen Daten tut, sind Meinungsfreiheit und freie Persönlichkeitsentfaltung in Gefahr. Genau dies (dass Bürger also das Recht haben müssen, Daten in Kenntnis der Auswertungsmethoden und ihrer Resultate übermitteln zu können) ist die Argumentation des Bundesverfassungsgerichts in seiner Herleitung des Rechts auf informationelle Selbstbestimmung. Dieses Recht und seine Begründung wie auch der Anspruch des Bürgers auf den Schutz der Grundrechte werden im letzten Teil der Reihe hergeleitet, und es wird in einem abschließenden Rollenspiel auch erarbeitet, warum es keine einfachen Lösungen gibt: Das Recht auf informationelle Selbstbestimmung und der Anspruch auf den Schutz dieses Rechts (z. B. durch Datenschutzgesetze und deren Durchsetzung) kann durchaus im Konflikt mit anderen Grundrechten wie der Vertragsfreiheit stehen. Denn hat der Nutzer nicht „freiwillig" seine Daten gegeben, um den „kostenlosen" Dienst zu bekommen (den der Anbieter natürlich anderweitig finanzieren muss)? Eben diese Frage ist im sog. Lüth-Urteil vom BverfG allerdings beantwortet – in den Worten Dreiers:

> „In Umkehr der Schutzrichtung des subjektiv-defensiven Abwehranspruchs sinnt der Schutzpflichtgedanke dem Staat an, den einzelnen Bürger vor Ein- und Übergriffen in dessen Rechtssphäre durch private Dritte zu schützen und (...) eine Rechtsgutsverletzung zu vermeiden." [Dr93, S. 47]

Aus diesem Urteil ergab sich für die Grundrechte a) ihre „Ausstrahlungswirkung": Jedes Gesetz, jede Rechtsnorm muss in ihrem Geist abgefasst sein, und b) die „mittelbare Drittwirkung: die Grundrechte wirken nicht unmittelbar zwischen den Grundrechtssubkjekten, also den Bürgern, doch ist ihre Geltung bei der Anwendung des Privatrechts zu beachten, mit anderen Worten: Grundrechte stehen über privatrechtlichen Vereinbarungen. Damit werden aus Grundrechten Grundnormen, die über dem positiven Recht stehen, welches sich aus ihnen ableitet. Freie Persönlichkeitsentfaltung und Meinungsfreiheit sind dabei von so entscheidender Bedeutung, dass privatrechtliche Vereinbarungen, die auf ihre Aushöhlung hinauslaufen, der freiheitlich-demokratischen Grundordnung widersprechen und als verfassungsfeindlich einzustufen sind.[20]

[20] Ähnlich argumentiert der lesenswerte Artikel des prominenten Datenschützers Thilo Weichert am Beispiel Facebook: „Datenschutzverstoß als Geschäftsmodell – der Fall Facebook", s.: https://www.datenschutzzentrum.de/facebook/20120921-facebook-geschaeftsmodell.pdf

3 Neue Unterrichtsideen zur Stärkung der informationellen Selbstbestimmung

Nachdem im vorangegangenen Abschnitt schon fertig ausgearbeitete Unterrichtsreihen vorgestellt wurden, zu denen auch Erfahrungsberichte aus dem Unterricht vorliegen , werden nun erste Ansätze und Ideen präsentiert, die sich noch in der Entwicklung befinden. Hierbei geht es auch darum, die in Abschnitt 2 vorgestellten Projekte mit den neuen Ideen zu verbinden, da jedes für sich bestimmte Fragen offen lässt.

3.1 Erste Überlegungen aus der Königsteiner Arbeitsgruppe

In der Arbeitsgruppe „Datenschutz im 21. Jahrhundert“ der 22. Fachdidaktischen Gespräche[21] zur Informatik in Königstein (Sachsen) vom 25.03. bis 27.03.2015 haben wir Erfahrungen in der Vermittlung dieses Themas zusammen getragen. Dabei haben wir uns gefragt, welche Themen und Oberbegriffe im Zusammenhang mit „Privacy und Datenschutz“ stehen, welche davon im Rahmen einer Unterrichtsreihe behandelt werden und welche möglichen Vernetzungen untereinander herausgestellt werden sollten. Darüber hinaus wurden frei zugängliche Materialien auf ihre Eignung für eine solche Unterrichtsreihe hin untersucht.

Zur Themensammlung und einer möglichen Strukturierung wurde eine Mind Map[22] erstellt, die einen guten Überblick über das gesamte Gebiet gibt. Als Projekttitel wurde „Meine Privatsphäre“ gewählt, weil dadurch zum einen der persönliche Bezug zur Sache („meine“) und zum anderen der gesellschaftswissenschaftliche Schwerpunkt der Unterrichtsreihe betont werden kann. Ziel ist die Erstellung einer oder mehrerer kontextorientierter Unterrichtsreihe(n) [IniK], die modular aufgebaut sein sollen, sodass die Lehrkraft je nach Altersstufe (Sek. I oder Sek. II) und Vorkenntnissen eine auf die Lerngruppe zugeschnittene Unterrichtssequenz zusammenstellen kann.

Einstiege in eine solche Reihe bieten sich aus drei Themenfeldern an: a) Kommunikationsmittel, b) gesellschaftlicher Diskurs, und c) informationelle Selbstbestimmung als zugrundeliegendes Prinzip. Ein Beispiel zu a) skizzieren wir im folgenden Abschnitt.

Darüber hinaus ist eine Sensibilisierung für das Thema „Meine Privatsphäre“ auch durch kurze Filme (z. B. „Wir lieben Überwachung“[23]) oder kritische Jugendbücher (z. B. „Little Brother“ von Cory Doctorow[24]) möglich. Im letzten Fall bietet sich zudem ein fächerverbindender Unterricht mit den Fächern Deutsch und/oder Ethik an.

---

[21] http://dil.inf.tu-dresden.de/Koenigsteiner-Gespraeche.262.0.html
In der Arbeitsgruppe haben mitgearbeitet: Rita Freudenberg, Andreas Grillenberger, Marc Hannappel, Alexander Hug, Peter Juknat, Holger Rohland, Michael Unger, Helmut Witten.

[22] http://bscw.schule.de/pub/bscw.cgi/d1205723/Meine%20Privatsph%C3%A4re.pdf

[23] https://www.youtube.com/watch?v=qGvZveB1osw und auch http://alexanderlehmann.net/

[24] http://de.wikipedia.org/wiki/Little_Brother_%28Roman%29

3.2 Ein möglicher Weg durch einen Einstieg über Messenger-Dienste

Der Einstieg in die Unterrichtsreihe „meine Privatsphäre“ könnte dadurch erfolgen, dass man mit den Schülern über die von ihnen genutzten Messenger-Diensten diskutiert. Dazu bieten sich folgende Leitfragen an: Welche Dienste nutzt Du? Wozu nutzt Du die jeweiligen Dienste konkret? Wie häufig ist dies am Tag? Welche Vorteile bieten Dir die Messenger-Dienste gegenüber anderen Kommunikationsmöglichkeiten? Es bietet sich je nach Kursgröße an, erste Diskussionen in Kleingruppen zu beginnen und anschließend ins Plenum zu wechseln, um alle Lernenden besser einbeziehen zu können und eine Vielzahl von Antworten zu erhalten. Damit diese Betrachtung nicht nur auf einer theoretischen Ebene verbleibt und geeignete Dienste auch genutzt werden, kann sich die Lerngruppe an dieser Stelle für ihre weitere Kommunikation auf einen Dienst einigen.

Im nächsten Schritt kann man unter Zuhilfenahme von Materialien (Artikel, Berichte auf Webseiten, …) einerseits die unverschlüsselte Kommunikation[25] und andererseits die Zusammenhänge diverser Dienste wie Facebook, WhatsApp, usw. herausstellen[26]. Aufgrund dieser Kenntnis muss den Schülern einsichtig werden, dass durch die Nutzung all dieser Dienste ein Datenschatten bzw. ein zweites „Ich“ von ihnen in der digitalen Welt existiert, dass dem zweiten „Ich“ Eigenschaften zugeordnet werden, die nicht gültig sein müssen, und dass das „Recht auf Vergessen“ in der digitalen Welt nicht automatisch existiert. In dieser Unterrichtsphase bietet sich an, analog der Beschreibung in [Be14b] im praktischen Teil Tracker einzusetzen.

Die Lernenden werden mit der Antwort „Ich habe nichts zu verbergen!“ argumentieren, was dann zur Diskussion über die Frage „Was ist Privatsphäre? Und wo sind die Grenzen zur Öffentlichkeit?“ überleitet. Wie in [Be14b] beschrieben, ist hier der Unterschied zwischen der institutionellen und der sozialen Privacy herauszuarbeiten.

An diesem Punkt der Reihe wird man sicherlich Schüler finden, die dem Gebrauch z. B. von Facebook kritisch oder gar ablehnend gegenüberstehen, und andererseits Schüler in der Lerngruppe haben, die ein System wie Facebook begrüßen. Daher bietet es sich an, Argumente und Gründe für oder gegen eine Nutzung von Facebook zu sammeln, wobei hier der Standpunkt, aus dem man argumentiert, eine entscheidende Rolle spielt. Als Fazit könnte sich schließlich eine Aussage wie „If you‘re not paying for it, you are the product” stehen.

Der Nutzung der Daten hat man durch Kenntnisnahme der AGB bestätigt (s. aber oben, Abschnitt 2.3: „Lüth-Urteil“). Somit steht nun die Analyse solcher Bedingungen an, wobei gruppendifferenziert verschiedene Dienste betrachtet werden können. Unweigerlich steht man nun an einem Punkt, wo der Begriff Datenschutz und damit Datenschutzgesetz und informationelle Selbstbestimmung eine Rolle spielen. Das Volkszählungs-

[25] Probleme bei der (angeblichen?) Verschlüsselung von WhatsApp wurden von Heise Security aufgezeigt: http://www.heise.de/security/artikel/Der-WhatsApp-Verschluesselung-auf-die-Finger-geschaut-2629020.html
[26] http://medien-mittweida.de/55145/whatsapp-instagram-oculus/

urteil von 1983 und seine Konsequenzen stehen nun im Folgenden im Mittelpunkt der Betrachtungen.

Die Lauschangriffe im Netz haben eine andere Dimension: Der Meinungspluralismus, der ein Kennzeichen der Demokratie ist, geht verloren und Uniformität gewinnt die Oberhand (vgl. dazu auch [Be14b] und [Ko14]). Durch Verwendung entsprechender Software und Werkzeuge (z. B. kryptografischer Tools oder verschlüsselnde Messenger-Programme) kann dem entgegen gewirkt werden. Daher scheint es nur konsequent, jungen Menschen die Verwendung solcher Selbstschutz-Möglichkeiten aufzuzeigen.

3.3 Crypto-Wars als Unterrichtsthema

Ein weiterer, historisch orientierter Einstieg zum Thema Datenschutz und Privacy kann über die Crypto Wars geschehen. Crypto Wars ist ein inoffizieller Name für den Versuch der U.-S.-amerikanischen Regierung, den Zugang zur Kryptographie für die Öffentlichkeit und andere Nationen soweit zu begrenzen, dass diese sich Entschlüsselung durch die NSA und andere Geheimdienste nicht widersetzen können[27].

4 Abschlussdiskussion

In diesem Artikel haben wir fertig ausgearbeitete und erprobte Unterrichtsvorschläge diskutiert und Überlegungen und Ideen zu neuen Unterrichtsreihen zur Stärkung der informationellen Selbstbestimmung vorgestellt. Diese Ideen gilt es nun weiter auszuarbeiten. Im Rahmen des Intensivworkshops möchten wir bislang gesammelte Materialien und Ideen zur Diskussion stellen, vor allem aber mit Hilfe der Teilnehmer weitere, noch nicht bedachte Aspekte herausarbeiten und ggf. weitere Unterrichtsideen entwickeln.

Literaturverzeichnis

Alle Internetquellen wurden am 20. Juni 2015 überprüft.

[Be14a] Berendt, B.; De Paoli, S.; Laing, C.; Fischer-Hübner, S.; Catalui, D. & Tirtea, R. (2014). Roadmap for NIS education programmes in Europe. ENISA Report. https://www.enisa.europa.eu/activities/stakeholder-relations/nis-brokerage-1/roadmap-for-nis-education-programmes-in-europe.

[27] https://de.wikipedia.org/wiki/Crypto_Wars und http://en.wikipedia.org/wiki/Crypto_Wars. Inzwischen wurden von Helmut Witten und Frank Oppermann erste Materialsammlungen zu diesem Thema zusammengestellt: http://de.padlet.com/frank_oppermann/bi9ysn0gl7dp/wish/53630394 und http://bscw.schule.de/pub/bscw.cgi/1198421

[Be14b] Berendt, B.; Dettmar, G.; Demir, C. & Peetz, T.: Kostenlos ist nicht kostenfrei. Oder: If you're not paying for it, you are the product". LOG IN Heft 178/179 (2014), S. 41-56.

[Dr93] Dreier, H:. Dimensionen der Grundrechte. Von der Wertordnungsjudikatur zu den objektiv-rechtlichen Grundrechtsgehalten, Hannover 1993.

[Gr15] Grillenberger, A., Romeike, R.: Big-Data-Analyse im Informatikunterricht mit Datenstromsystemen: Eine Unterrichtsbeispiel. (2015) In diesem Band.

[GHW11] Gramm, A.; Hornung, M.; Witten, H.: "E-Mail (nur?) für Dich - Eine Unterrichtsreihe des Projekts Informatik im Kontext". Beilage zu LOG IN Heft 169/170 (2011).

[IniK] http://www.informatik-im-kontext.de.

[Kl13] Klicksafe: Datenschutz TIPPS für Jugendliche. (2013) http://www.klicksafe.de/fileadmin/media/documents/pdf/klicksafe_Materialien/Jugendliche/klicksafe_Flyer_Datenschutztipps_Jugend_2013.pdf.

[Ko14] Koubek, J.: Datenschutz und Persönlichkeitsrechte. LOG IN Heft 178/179 (2014), S. 21-26.
http://medienwissenschaft.uni-bayreuth.de/assets/Uploads/Koubek/forschung/KoubekDatenschutzPersoenlichkeitsrechtePreprint.pdf.

[So11] Solove, D.: Nothing to Hide: The False Tradeoff between Privacy and Security, Chapter 1, Yale University Press, 2011

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 43

Informatik Enlightened - Informatik (neu) beleuchtet dank Physical Computing mit Arduino

Nadine Bergner[1] und Ulrik Schroeder[2]

Abstract: Informatik (in ihrer Breite) begreifbar machen - so lautet das Ziel des Schülerlabor-Moduls „Informatik enlightened“. Mittels Mikrocontrollern können Mittelstufenschülerinnen und -schüler die Vielseitigkeit der Informatik entdecken. Aufbauend auf Eigenschaften des entdeckenden Lernens und des Physical Computing werden die verschiedenen Stationen unter Berücksichtigung inhaltlicher sowie didaktischer Entscheidungen beschrieben. Ziel des Moduls ist es, Schülervorstellungen über Informatik zu einem der Wissenschaft gerecht werdenden Bild hin zu verändern. Dies wurde über ein Pre-Post-Testdesign mit 128 Teilnehmerinnen und Teilnehmern evaluiert.

Keywords: Informatikunterricht, Bild der Informatik, Programmiereinstieg, entdeckendes Lernen, individuelles Lerntempo, Hands-On-Material, Mikrocontroller, Evaluation

1 Einleitung und Motivation

Schon Konfuzius stellte fest, dass die meisten Menschen dann am besten lernen, wenn sie mit dem Lerngegenstand aktiv arbeiten. Für die Informatik als Wissenschaft abstrakter Konzepte und digitaler Produkte stellt dies eine besondere Herausforderung dar. „Physical Computing“ ist ein Ansatz, Informatik begreifbar zu machen. Er verdeutlicht, wie (meist) interaktive, aus Hard- und Software bestehende Systeme direkte Wechselwirkungen mit der realen Welt haben [OI04]. Eine Möglichkeit, Physical Computing auf Schulniveau umzusetzen, stellt die Mikrocontrollerplattform Arduino[3] dar. Aufbauend auf dieser Hard- und Software entstand das im Folgenden präsentierte Lernangebot, welches seit Anfang 2014 im InfoSphere, dem Schülerlabor Informatik der RWTH Aachen, angeboten wird. Kindern und Jugendlichen ab Klasse acht wird durch lebensnahe Kontexte und großen Freiraum ein individueller Einstieg in die textuelle Programmierung ermöglicht sowie ein Einblick in den Facettenreichtum der Informatik gegeben. Um die Lernenden bestmöglich zu fördern, erhielten die Aspekte aktives, selbstbestimmtes Lernen, alltagsnahe Kontexte, individuelles Lerntempo und Teamarbeit besondere Beachtung.

2 Hintergründe und Related Work

Begründet ist die methodische Leitlinie - Lerngegenstände anfassbar zu machen und so das aktive Arbeiten mit dem Lerngegenstand zu unterstützen - in der konstruktivistischen

[1] RWTH Aachen, Lehr- und Forschungsgebiet Informatik 9, Ahornstr. 55, 52074 Aachen, bergner@informatik.rwth-aachen.de
[2] dto., schroeder@informatik.rwth-aachen.de
[3] `http://www.arduino.cc/`

Lerntheorie von Piaget [Vo97]. Wichtige didaktische Modelle in diesem Zusammenhang sind das entdeckende wie auch forschende Lernen, welche dazu beitragen, dass die Lernenden sich aktiv mit den Lerngegenständen auseinandersetzen [Ae02], [AHR05]. Physical Computing ermöglicht, Informatiksysteme als erfahr- und erkundbare Lerngegenstände einzusetzen. Gegenüber recht geschlossenen Plattformen wie beispielsweise Lego Mindstorms[4]) bietet eine Mikrocontroller-Plattform für Schulen den Vorteil, dass sie eine weit größere kreative Gestaltungsfreiheit bietet [PR13]. Für die Arbeit mit einzelnen Bauteilen (Sensoren, Aktoren, Kabeln etc.) spricht zudem, dass die Identifikation der Schülerinnen und Schüler mit dem Lerngegenstand durch den eigenständigen Aufbau der Schaltungen oder Systeme gestärkt wird.

Vorarbeiten zu Schülervorstellungen über Informatik ([Hu01], [Lo03], [SM05]) betonen, dass hauptsächlich das Medium Computer mit dem Begriff Informatik assoziiert wird. Experten definieren die Informatik dagegen wesentlich vielfältiger und breiter [Fr05]. Mit dem Ziel, die Sichtweise der Jugendlichen dem Selbstbild der Wissenschaft anzunähern und die von Claus etablierte Dreiteilung in technische, praktische und theoretische Informatik [Cl75] zu berücksichtigen, wurde das Modul „Informatik Enlightened“ entwickelt. Eine umfassende Diskussion zum Bild der Informatik aus Sicht der Kinder und Jugendlichen, wie auch aus Sicht der Fachcommunity, ist in [Be14] zu finden.

Eine Orientierung, welche informatischen Inhalte und Kompetenzen es in der Sekundarstufe I zu vermitteln gilt, bieten die Empfehlungen für Bildungsstandards der Gesellschaft für Informatik [Ge08], welche als Grundlage der im Weiteren formulierten Leitlinien dienen. Dabei ist insbesondere der Inhaltsbereich „Informatiksysteme“ durch den Arduino, als alternatives Informatiksystem abgedeckt. Auch die Inhaltsbereiche „Algorithmen“ und „Informatik, Mensch und Gesellschaft“ werden im Modul explizit behandelt. Darüber hinaus spielen die Prozessbereiche „Modellieren und Implementieren“ sowie „Kommunizieren und Bewerten“ eine wichtige Rolle.

3 Ausgestaltung des Moduls

Mit dem hier vorgestellten Modul wurde die *Forschungsfrage* untersucht, inwiefern ein Physical Computing Lernangebot geeignet ist, das Bild der Informatik als ein Zusammenspiel von Hard- und Software, Logik und Teamarbeit darzustellen. Die *Zielgruppe des Konzepts* sind Schülerinnen und Schüler am Ende der Sekundarstufe I. Das Modul soll ihnen als Ausblick auf den Informatikunterricht der gymnasialen Oberstufe dienen. Die grundlegende *didaktische Leitlinie* ist die eigenständige Arbeit in kleinen Teams. Dabei soll jedes *Zweierteam* im individuellen Tempo voranschreiten.

Strukturell gliedert sich die Maßnahme in eine kurze Einführung, eine grundlegende Station 0, frei wählbare vertiefende Stationen 1 bis 4 sowie einen abschließenden Museumsgang. Die etwa 15-minütige Einführung gibt einen Überblick über das Modul und stellt insbesondere die zur Auswahl stehenden vier vertiefenden Stationen (*Sonnenblume, Einparkhilfe, Geschwindigkeitsmessung und Farbthermometer*) vor. Anschließend er-

[4] `mindstorms.lego.com`

arbeiten sich die Teams selbstständig die benötigten Grundlagen, sowohl bezüglich des Aufbaus von elektronischen Schaltungen, als auch hinsichtlich der Programmierung des Arduino-Mikrocontrollers (ca. 1h 15min). Im Anschluss geht es an die vertiefenden Stationen (ausgelegt auf je 1h 45min), wobei die Auswahl, Reihenfolge und auch der Grad der Vertiefung den Teams selbst überlassen bleibt. Den Abschluss bildet ein Museumsgang, in dem alle Teilnehmerinnen und Teilnehmer an den verschiedenen Stationen ihre Erfolge, aber auch Hürden und wie diese gemeistert wurden, schildern.

Zur Ausgestaltung des Moduls bedarf es sowohl didaktischer als auch inhaltlicher Überlegungen. Dazu wird im Folgenden die konkrete Umsetzung jeweils ausgehend von Leitlinien und ihren theoretischen Begründungen erläutert.

3.1 Didaktische Überlegungen

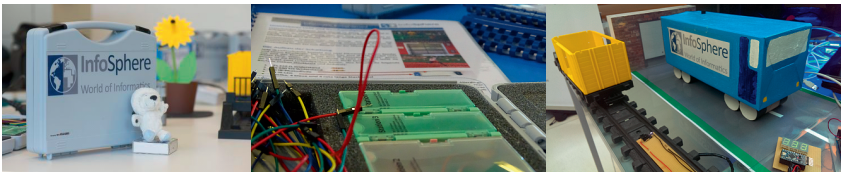
Die *didaktischen Leitlinien*[5] umfassen insbesondere folgende Aspekte:

1. Die Aktivität liegt hauptsächlich bei den Teilnehmerinnen und Teilnehmern.
2. Die Schülerinnen und Schüler lernen und arbeiten in ihrem persönlichen Lerntempo.
3. Der Lernprozess wird von den Teilnehmerinnen und Teilnehmern aktiv mitbestimmt.
4. Die Verlaufsmotivation wird durch zahlreiche Erfolgserlebnisse hoch gehalten.
5. Das Feedback zum Lernprozess erfolgt hauptsächlich durch die Materialien.
6. Sozialkompetenzen (z.B. Teamwork und sinnstiftende Kommunikation) werden explizit gefördert.
7. Das informatische Arbeiten besteht aus Planungs- und Umsetzungsphase.

Die Relevanz von *eigenaktivem Lernen* (Punkte 1 und 3), insbesondere im Fach Informatik, wird unter anderem in [SS11] ausführlich dargelegt. Auch zur Bedeutung der *individuellen Förderung* in Punkt 2 und 3 gibt es viele allgemein didaktische Abhandlungen. Klieme und Warwas beschreiben „Individuelle Förderung als selbstverständliches Merkmal pädagogischen Handels“ [KW11]. Aufgrund des reinen Wahlpflichtcharakters des Faches Informatik (in Nordrhein-Westfalen wie auch einigen weiteren Bundesländern) und damit der häufig sehr unterschiedlichen Vorkenntnisse der Schülerinnen und Schüler - auch aus dem privaten Bereich - spielt die individuelle Gestaltung der Lernprozesse eine besondere Bedeutung. Auch das Thema *Motivation* (Leitlinie 4), insbesondere Einstiegs- und Verlaufsmotivation, sind ausschlaggebend für den Erfolg von Lernprozessen, nicht nur im Kontext Schule [St02], [HD14]. Das automatisierte Feedback durch das Informatiksystem ermöglicht Lehrkräften neue Freiräume zur individuellen Unterstützung der Lernenden. Die sechste und siebte Leitlinie werden durch die Prozessbereiche der GI-Standards [Ge08] ausführlich motiviert. Im weiteren wird die konkrete *Umsetzung der didaktischen Leitlinien* beschrieben.

[5] Diese gelten für alle weiteren Module des Schülerlabors InfoSphere ist analoger Weise und werden mit unterschiedlichen Schwerpunkten in allen Altersgruppe umgesetzt.

Leitlinien 1 & 2: Um sowohl die *Identifikation mit dem Lerngegenstand*, als auch im weiteren Verlauf des Moduls das *individuelle Lerntempo* zu ermöglichen, erhält jedes Zweierteam bereits zu Beginn des Kurses ein InfoSphere-Kit mit allen elektronischen Bauteilen (u.a. LEDs, Taster, Sensoren, Motoren), die für das Modul benötigt werden (siehe Abbildung 1). Auch alle Arbeitsmaterialien in Form von laminierten Begleitheften stehen den Lernenden frei zugänglich zur Verfügung. Um die Lernenden zu Beginn nicht zu überfordern, finden die Teams an ihrem vorbereiteten Arbeitsplatz nur das Begleitheft der Station 0 sowie die vorsortierten Bauteile.



(a) InfoSphere-Kit (b) Material (c) Ausstellungsstücke

Abb. 1: InfoSphere-Kit und Exponate

Damit keine weitere Frontalphase den eigenständigen Arbeitsprozess der Teams unterbricht, werden die vier vertiefenden Stationen bereits im Einstieg präsentiert. Durch Arbeitsblätter mit anspruchsvollen Bonusaufgaben ist neben der Anzahl auch der Vertiefungsgrad der Stationen individuell gestaltbar, beispielsweise können beim Farbthermometer über die klassischen Grundfarben hinaus fließende Farbübergänge implementiert werden.

Leitlinien 3 & 7: Um gleichzeitig Lernenden viel Freiraum zu lassen, aber dennoch eine effektive Unterstützung durch das Betreuerteam zu ermöglichen, sind in den Begleitheften Lücken vorgesehen, in die die Schülerinnen und Schüler Notizen zu den verwendeten Anschlüssen des Arduino oder auch Überlegungen zum Programmcode schriftlich festhalten. So wird der Fokus neben der aktiven Umsetzung am Computer auch auf den, für die Informatik sehr wichtigen, Planungsprozess gelegt. Damit wird der Prozessbereich „Modellieren und Implementieren“ aus den Standards der GI vertieft.

Leitlinien 4 & 5: Diese beiden Leitlinien begründen die Auswahl eines Mikrocontrollers als Werkzeug für den Programmiereinstieg. So kann das Verhalten der an den Mikrocontroller angeschlossenen Aktoren und Sensoren als direktes Echte-Welt-Feedback den Lernenden Rückmeldung zum Erfolg ihres Lernprozesses geben. Wenn beispielsweise in Station 0 die LED aufgrund des Tasterdrucks aufleuchtet, wissen die Schülerinnen und Schüler, dass sowohl die Schaltung als auch die Programmierung korrekt sind und können selbstständig zur nächsten Herausforderung fortschreiten.

Leitlinie 6: Diese Leitlinie betrifft den Prozessbereich „Kommunizieren und Kooperieren“ und betont Teamarbeit als eine wichtige Informatikkompetenz. Hierbei werden die Kompetenzen verschiedener Lernender gewinnbrngend kombiniert, indem sich Herausforderungen im Bereich Aufbau elektronischer Schaltungen mit solchen im Bereich der Programmierung ergänzen.

3.2 Fachliche Inhalte

Die *fachlichen (Lern-)Ziele*[6] gestalten sich in diesem Modul wie folgt. Die Lernenden

1. ...beschreiben Mikrocontroller samt Komponenten als alternatives Informatiksystem.
2. ...können einfache und im weiteren Verlauf auch komplexere Programmabläufe (Algorithmen) entwickeln und implementieren.
3. ...wenden ihr Wissen über Programmierkonstrukte (u.a. Variablen, Schleifen, bedingte Verzweigungen und Methoden) an.
4. ...erläutern den Umgang mit Sensorwerten (einlesen, umrechnen, ausgeben).
5. ...diskutieren den Lebensweltbezug sowie die Interdisziplinarität der Informatik.

Aus den Standards der GI lässt sich die Motivation ableiten, Mikrocontroller und entsprechende Bauteile als alternatives Informatiksystem einzuführen (Lernziel 1). Das zweite Lernziel beschäftigt sich mit einer fundamentalen Idee der Informatik, der „Algorithmisierung“ [Sc93]. Dabei geht es sowohl um die gedankliche Entwicklung als auch um die konkrete Umsetzung von Algorithmen. Da die Zielgruppe dieses Moduls in erster Linie Schülerinnen und Schüler umfasst, welche bereits mit einer visuellen Programmiersprache gearbeitet haben, ist das Thema Algorithmik eine Vertiefung. Ähnliches gilt auch für Lernziel 3, da die Schülerinnen und Schüler bereits bekannte Programmstrukturen vertiefen. Falls im Vorfeld ein Einstieg in die Robotik stattgefunden hat, bietet das nächste Thema (Lernziel 4) eine hervorragende Anknüpfung. Diese umfasst dabei neben dem Auslesen auch die Umrechnung und Darstellung der Sensorwerte und bedient damit den Prozessbereich „Darstellen und Interpretieren“. Das fünfte Lernziel fällt in den Inhaltsbereich „Informatik, Mensch und Gesellschaft“. Die Schülerinnen und Schüler sollen erkennen, welche Alltagsrelevanz und auch welch vielseitiges fächerverbindendes Potential die Informatik hat. Im Folgenden wird die konkrete *Umsetzung der fachlichen Lernziele* beschrieben.

Zu Lernziel 1: Zum einen sollen die Schülerinnen und Schüler erkennen, dass neben dem Computer noch unzählige weitere Arten von Informatiksystemen existieren. Zum anderen sind die drei Komponenten - Eingabe, Verarbeitung und Ausgabe - in einem hardwarenahen System deutlich zu erkennen.

Zu Lernziel 2: Bereits in Station 0 erarbeiten sich die Teams mit einer LED und einem Taster einen einfachen Algorithmus. In den weiterführenden Stationen werden die Algorithmen schrittweise komplexer. In der Station zur Sonnenblume wird aufgrund der bedingten Richtungswechsel (je nach Helligkeit) der schwierigste Algorithmus implementiert. Auch die elementaren Schritte der Softwareentwicklung, beispielsweise das Modellieren auf Papier, das anschließende Implementieren und letztendlich das Testen, bevor es unter Umständen an die Fehlersuche geht, werden von den Lernenden durchlaufen.

[6] Diese sind teilweise angelehnt an die Inhaltsbereiche der GI-Bildungsstandards.

Zu Lernziel 3: Alle Programmierkonstrukte werden an geeigneter Stelle eingeführt und wiederholt. Bedingte Verzweigungen („if-else“-Konstrukt) werden bei der Tastersteuerung in Station 0 erstmalig erklärt und in den vertiefenden Stationen in unterschiedlichen Kontexten wiederholt. Aufbauend auf Integer-Variablen werden Float-Variablen erst eingeführt, wenn diese zur Messung der Geschwindigkeit benötigt werden. Die Themen Methoden und Schleifen werden implizit angewendet („setup“ und „loop“), aber nicht aktiv thematisiert.

Zu Lernziel 4: In allen Stationen wird der Programmablauf durch eingehende Sensorwerte verändert. Bei der einführenden Station 0 werden lediglich die beiden Zustände eines Tasters ausgelesen, in der Station „Sonnenblume“ wird bereits mit einem komplexen Lichtwiderstand bearbeitet. Durch den Einsatz verschiedener Sensoren erkennen die Jugendlichen den Unterschied zwischen digitalen und diskretisierten analogen Eingängen. Dabei wird verdeutlicht, dass Sensorwerte häufig umgerechnet werden müssen. Beispielsweise gibt der Temperatursensor in der Station „Farbthermometer“ nicht direkt Werte in °C zurück, sondern Sensorwerte zwischen 0 und 1023.

Zu Lernziel 5: Dieses Lernziel wird hauptsächlich durch die Wahl der Kontexte erreicht. Dabei reichen diese von alltagsnahen Beispielen wie der piepsenden Einparkhilfe bis zu fächerverbindenden Kontexten wie der Steuerung von Solaranlagen, welcher den Lebensweltbezug der Sonnenblume darstellt. Der fächerverbindende Charakter der Informatik zur Elektrotechnik bzw. Physik wird bereits im kurzen Intro durch die Themen Stromkreis, LEDs mit Widerstand, Polung von LEDs und Funktionsweise von Tastern deutlich. Die physikalischen Hintergründe stehen hier nicht im Fokus, stellen jedoch eine Option der Vertiefung dar. Weiter wird in der Station „Farbthermometer“ zusätzlich die Verbindung zum Kunstunterricht deutlich, indem mit dem Thema RGB-LED die additive (im Gegensatz zur subtraktiven) Farbmischung eingeführt wird.

4 Empirische Analyse

Das Modul wurde im Schülerlabor Informatik InfoSphere an der RWTH Aachen im Zeitraum Juni bis Dezember 2014 *achtmal* mit Schülergruppen durchgeführt.[7] Evaluiert wurden die Durchführungen mittels eines Pre-Post-Testdesigns, welches aus zwei Online-Fragebögen[8] besteht, wobei der Pretest meist wenige Tage vor, der Posttest direkt nach dem Besuch des Moduls durchgeführt wurde. Durch die zeitliche Nähe beider Befragungszeiträume ist eine Kontrollgruppe verzichtbar, da davon ausgegangen werden kann, dass sich die erhobenen Aspekte ohne Intervention im Laufe weniger Tage nicht signifikant verändern. Insgesamt liegen zur statistischen Analyse *128 gepaarte Pre- und Posttest-Datensätze* vor. Alle Durchführungen wurden von Programmieranfängerinnen und -anfängern besucht, dabei entstammten 88 Schülerinnen und Schüler der Zielgruppe der Mittelstufenkurse (Klasse 8 und 9 von Realschulen und Gymnasien) und die übrigen 40

[7] Im Vorfeld fanden bereits weitere Durchführungen statt, mit dem Ziel das Feedback der Schülerinnen, Schüler sowie Lehrkräfte in die Weiterentwicklung und Verbesserung einfließen zu lassen.

[8] Diese wurden aufbauend auf etablieren Fragebögen, u.a. INCOBI (siehe [RNG01]), SUCA (siehe [MS05]), FEWI (siehe [Lo03]), selbst entwickelt.

Personen der Einführungskurse an einem Berufskolleg. Insgesamt nahmen *32 Mädchen (25%)* und *96 Jungen (75%)* an der Evaluation teil.

Die erste Frage befasst sich mit spontanen Assoziationen zum Begriff Informatik. Im Vergleich der Pre- und Posttestdaten zeigen sich bei den Begriffen „Programmierung“, „Computer“, „Technik“ und „Logik“ signifikante Veränderungen. Der Begriff „Computer“ wurde nach dem Workshop in Relation zu anderen Aspekten signifikant seltener genannt ($z = -4.964_p$, $p < .001$, $r = -.438$)[9]. Im Gegensatz dazu erhielten die Begriffe „Programmierung“, „Technik“ und „Logik“ mehr Gewicht (vgl. Tabelle 1). Dies zeigt bereits, dass das Ziel, die Breite der Informatik im Sinne der drei Säulen: *technische*, *praktische* und *theoretische Informatik* aufzuzeigen, durchaus als gelungen anzusehen ist.

Begriff	Anzahl vorher	Anzahl nachher	nur vorher	keine Veränderung	nur nachher	Teststatistik
Computer	75 / 58,6%	44 / 34,4%	35 / 27,3%	89 / 69,5%	4 / 3,1%	$z = -4.964_p$, $p < .001$, $r = -.438$
Programmierung	86 / 67,2%	98 / 76,6%	12 / 9,4%	92 / 71,9%	24 / 18,8%	$z = -2.000_n$, $p < .05$, $r = -.177$
Technik	8 / 6,3%	17 / 13,3%	3 / 2,3%	113 / 88,3%	12 / 9,4%	$z = -2.324_n$, $p < .05$, $r = -.205$
Logik	5 / 3,9%	17 / 13,3%	3 / 2,3%	110 / 85,9%	15 / 11,7%	$z = -2.828_n$, $p < .01$, $r = -.250$

Tab. 1: Signifikante Veränderungen in den Freitextantworten

Weiter konnte mittels t-Test[10] gezeigt werden, dass der Workshop dazu führt, dass die Jugendlichen weniger der Meinung sind, dass Informatik ein *Männerfach* sei ($t(78) = -2.550$, $p < .05$, $r = .264$)[11]. Leider erreicht der Workshop nicht das Ziel der *Interessenssteigerung*; bei einem Teil der Besucherinnen und Besucher ist dieses sogar gesunken ($t(125) = -2.027$, $p < .05$, $r = .178$). Diese Veränderung betrifft allerdings nicht die eigentliche Zielgruppe der Mittelstufenschülerinnen und -schüler, was sich unter anderem an der signifikanten Korrelation zum Alter zeigt ($p < .001$, $r = -.014$)[12].

Auch die Sicht auf die Profession und den Stereotyp Informatikerin bzw. Informatiker veränderte sich punktuell. So stimmten die Schülerinnen und Schüler nach der Durchfüh-

[9] Als statistisches Testverfahren wird hier der Wilcoxon-Vorzeichen-Rang-Test verwendet, für nähere Informationen siehe [Fi09]. Wichtig ist der Index, welcher angibt, ob die Berechnung auf positiven (p) oder negativen (n) Rängen beruht und entsprechend ein Absinken (bei p) oder einen Anstieg (bei n) anzeigt. p gibt weiter das Signifikanzniveau, also die Sicherheit des Effekts, und r die Effektstärke an.

[10] Auch hier sei auf [Fi09] verwiesen. Relevant ist, dass bei $t < 0$ ein positiver Zusammenhang vorliegt, sprich, dass allgemein eine Verschiebung zum rechten Skalenrand stattgefunden hat. Entsprechend zeigt $t > 0$ einen negativen Zusammenhang und somit eine Verschiebung nach links an. Bei der Interpretation kommt es somit immer auf die Zuordnung auf der Skala an.

[11] Zur Erhebung dieser Aspekte wurde ein Schieberegler verwendet, der an beiden Extrema mit gegensätzlichen Wortpaaren beschriftet ist, hier beispielsweise „Männerfach“ und „Frauenfach“.

[12] Dies wurde mittels Chi-Quadrat-Test ermittelt.

rung der Aussage über Informatik als *(reine) Computerwissenschaft* signifikant weniger zu ($t(127) = 3.619$, $p < .001$, $r = .306$). Obwohl im Workshop durchgehend in Partnerarbeit gearbeitet wird, sinkt bei den Lernenden von Gymnasien und Realschulen die Zustimmung zur Aussage *„Informatikerinnen und Informatiker müssen gut im Team arbeiten können."* signifikant ($t(85) = 2.214$, $p < .05$, $r = .234$). Die Gründe dazu konnten bisher nicht ermittelt werden und müssen in einer (vielleicht auch qualitativen) tiefergehenden Evaluation erforscht werden.

Ein Ranking von zehn vorgegebenen Begriffen zum Thema Informatik deckt weiter auf, dass der Begriff *„Computerkenntnisse"*, den viele Schülerinnen und Schüler vor dem Besuch für sehr relevant hielten, insbesondere gegenüber *„Programmieren"* an Bedeutung verliert ($z = -5,155_n$, $p < .001$, $r = -.456$). Hier wird noch einmal deutlich, dass viele Jugendliche hauptsächlich das Medium Computer mit der Informatik assoziieren und vor ihrem Besuch nur eine stark eingeschränkte Sicht hatten, aber schon ein eintägiger Workshop darauf Einfluss haben kann. Dabei sei explizit angemerkt, dass allein die Erweiterung des Bildes um den Aspekt „Programmieren" keineswegs als abschließendes Ziel angesehen werden darf. Speziell zur Verdeutlichung von Aspekten der theoretischen Informatik eignen sich andere Module des Schülerlabors InfoSphere zur späteren Erweiterung des Bildes.

Aus den Fragen, nach dem eigenen Interesse *Programme selbstständig zu entwickeln* und gleichzeitig der Einschätzung, ob sie *lieber eigenständig versus angeleitet arbeiten* möchten, zeigt sich insbesondere bei den Schülerinnen und Schüler der Mittelstufe ein leichter Rückgang des Interesses ($z = -2.487_p$, $p < .05$, $r = -.267$) und der Wunsch nach mehr Anleitung ($t(117) = -2.412$, $p < .05$, $r = .218$.). Daraus lässt sich schließen, dass die Teilnehmenden zumindest teilweise etwas überfordert waren und sich den Freiheiten an einigen Stellen nicht gewachsen fühlten. Dies verdeutlicht noch einmal, dass gerade beim entdeckenden Lernen darauf geachtet werden muss, Lernende keinesfalls zu überfordern.

5 Fazit

Aus der Evaluation ergibt sich, dass es dem hier präsentieren Schülerlabor-Modul gelungen ist, die Informatik als eine Kombination aus *technischer* (Begriff „Technik"), *praktischer* (Begriff „Programmierung") und auch *theoretischer Informatik* (Begriff „Logik") darzustellen. Auch dem Vorurteil des reinen „Männerfachs" kann teilweise entgegen gewirkt werden.

Weniger geeignet ist dieses Modul zur ersten Interessensgenerierung, bei einem Teil der Schülerinnen und Schüler nahm dieses sogar leicht ab. Auch die Herausforderungen der Methode des entdeckenden Lernens wurden sichtbar, so dass sich ein Teil der Lernenden von der Offenheit der Aufgaben etwas überfordert fühlte. Der Hands-On-Ansatz konnte diesem Effekt auch nicht vollständig entgegen wirken.

Insgesamt zeigt sich, dass der gewählte Ansatz durch Kombination der praktischen Arbeit beim Aufbau der Schaltungen und der gedanklichen Herausforderungen bei der Umset-

zung der Algorithmen durchaus Potential hat, Lernenden bereits in der Mittelstufe ein breites Bild der Informatik zu vermitteln.

6 Ausblick

Generell geben die Erkenntnisse Anlass zu zwei Erweiterungen bzw. Anpassungen. Zum einen wird das Modul zu einem *mehrtägigen Feriencamp* erweitert, sodass nach einer verlängerten, angeleiteten Einführung in die Mikrocontroller-Programmierung, die eigenen Ideen der Teilnehmerinnen und Teilnehmer mehr Raum finden. Somit soll die Motivation und damit das Interesse verstärkt sowie die Identifikation mit dem Lerngegenstand erhöht werden, was zumindest nach ersten Eindrücken der Testdurchführung im Januar 2015 gelingen kann. Zum anderen soll für Schülerinnen und Schüler vor oder zu Beginn des Wahlpflichtbereiches (ohne Programmiervorerfahrung) alternativ ein vereinfachter Zugang mittels visueller Programmierung ermöglicht werden.

Darüber hinaus wird das Angebot an Lehrerfortbildungen zu dem Modul ausgeweitet. Solche wurden bereits an elf verschiedenen Standorten Deutschlands (u.a. Bremen, Berlin, Cottbus, Dresden, Heidelberg und Aachen) durchgeführt, um auch die effektive Einbettung in den Schulunterricht zu ermöglichen und so eine große Breite der Schülerschaft zu erreichen.

7 Danksagung

Unser herzlicher Dank, auch im Namen aller Teilnehmerinnen und Teilnehmer, geht an die MakeLight-Initiative des Bundesministeriums für Bildung und Forschung (BMBF), wodurch die nötige Hardware (für mehrere Standorte) finanziert werden konnte.

Literaturverzeichnis

[Ae02] Aepkers, Michael; Liebig, Sabine; Bönsch, Manfred; Kaiser, Astrid: Entdeckendes, forschendes und genetisches Lernen, Jgg. 4 in Unterrichtskonzepte und -techniken. Schneider-Verl. Hohengehren, Baltmannsweiler, 2002.

[AHR05] Arnold, Ruedi; Hartmann, Werner; Reichert, Raimond: Entdeckendes Lernen im Informatik-Unterricht. 11. GI-Fachtagung Informatik und Schule (INFOS 2005), S. 197–205, 2005.

[Be14] Bergner, Nadine: Wie die Informatik sich selbst sieht und wie sie gesehen wird. In (Leicht-Scholten, Carmen; Schroeder, Ulrik, Hrsg.): Informatikkultur neu denken - Konzepte für Studium und Lehre Integration von Gender and Diversity in MINT-Studiengängen. Springer-Verlag, Heidelberg, 2014.

[Cl75] Claus, Volker: Einführung in die Informatik: Mit 48 Beispielen und 18 Aufgaben. Mathematik für das Lehramt an Gymnasien. Teubner, Stuttgart, 1975.

[Fi09] Field, Andy P.: Discovering statistics using SPSS: (and sex and drugs and rock 'n' roll). Introducing statistical methods. SAGE Publications, Los Angeles, London, 3. Auflage, 2009.

[Fr05] Frieze, Carol: Diversifying the images of computer science. In (ACM, Hrsg.): Proceedings of the 36th SIGCSE technical symposium on Computer science education. ACM, New York, NY, S. 397–400, 2005.

[Ge08] Gesellschaft für Informatik e.V. (GI): , Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I, 2008.

[HD14] Hildebrandt, Claudia; Diethelm, Ira: Students' Motivations, Self-concepts of Ability and Expectations Regarding the Subject Informatics: Results of a School Experiment. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education. WiPSCE '14, ACM, New York, NY, USA, S. 126–127, 2014.

[Hu01] Humbert, Ludger: Theoretischer und empirischer Vergleich zum Bild der Wissenschaft Informatik in der Schule. Informatica Didactica, 2001.

[KW11] Klieme, Eckhard; Warwas, Jasmin: Konzepte der Individuellen Förderung. Zeitschrift für Pädagogik, 57(6):805–818, 2011.

[Lo03] Lobbenmeier, Denise: Einstellungen von Schülerinnen zum Informatikunterricht und zur Informatik: Eine empirische Studie. Dissertation, Universität Paderborn, Paderborn, 18.06.2003.

[MS05] Magenheim, Johannes; Schulte, Carsten: Erwartungen und Wahlverhalten von Schülerinnen und Schülern gegenüber dem Schulfach Informatik – Ergebnisse einer Umfrage. Unterrichtskonzepte für informatische Bildung. Köllen, Bonn, S. 111–121, 2005.

[OI04] O'Sullivan, Dan; Igoe, Tom: Physical computing: sensing and controlling the physical world with computers. Course Technology Press, 2004.

[PR13] Przybylla, Mareen; Romeike, Ralf: Physical Computing im Informatikunterricht. In (Breier, Norbert; Stechert, Peer; Wilke, Thomas, Hrsg.): 15. GI-Fachtagung "Informatik und Schule": Praxisband. Kiel Computer Science Series, CAU Kiel, Kiel, S. 137–146, 2013.

[RNG01] Richter, Tobias; Naumann, Johannes; Groeben, Norbert: Das Inventar zur Computerbildung (INCOBI): Ein Instrument zur Erfassung von Computer Literacy und Computerbezogenen Einstellungen bei Studierenden der Geistes- und Sozialwissenschaften. Reinhardt, 2001.

[Sc93] Schwill, Andreas: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik, 25(1):20–31, 1993.

[SM05] Schulte, Carsten; Magenheim, Johannes: Novices' expectations and prior knowledge of software development: Results of a Study with High School Students. In (Anderson, Richard; Fincher, Sally; Guzdial, Mark, Hrsg.): International Journal of Environmental & Science Education. S. 143–153, 2005.

[SS11] Schubert, Sigrid; Schwill, Andreas, Hrsg. Didaktik der Informatik. Spektrum Akademischer Verlag, Heidelberg, 2011.

[St02] Stipek, Deborah J.: Motivation to learn: Integrating theory and practice. Allyn & Bacon, 2002.

[Vo97] Vollmers, Burkhard: Learning by doing–Piagets konstruktivistische Lerntheorie und ihre Konsequenzen für die pädagogische Praxis. International review of education, 43(1):73–85, 1997.

Das Bild der Informatik von Sachunterrichtslehrern

Erste Ergebnisse einer Umfrage an Grundschulen im Regierungsbezirk Münster

Alexander Best[1] und Sarah Marggraf[2]

Abstract: Der Sachunterricht an deutschen Grundschulen wurde von der Fachdidaktik Informatik in den letzten Jahren mehrfach als schulischer Ort informatischer Bildung vorgeschlagen. Über das bestehende Bild der Informatik von Sachunterrichtslehrern liegen bislang keine gesicherten Erkenntnisse vor. Zur Entwicklung verständlicher und praxisorientierter Unterrichtskonzepte sowie Bereitstellung von Weiterbildungsangeboten stellen diese ein Schlüsselelement dar. Denn nur dann, wenn den Lehrern ein differenziertes und fachlich korrektes Bild der Informatik vermittelt wird, können sie dieses auch an ihre Schüler weiterreichen. In einer schriftlichen Umfrage wurden deshalb Grundschullehrer des Faches Sachunterricht im Regierungsbezirk Münster (NRW) zu ihren Vorstellungen zur Informatik befragt. Die Ergebnisse werden in diesem Beitrag dargestellt, diskutiert und interpretiert.

Keywords: Grundschule; Primarbildung; Sachunterricht; Bild der Informatik; Umfrage

1 Skizzierung und Problematisierung des Forschungsstandes

In den vergangenen Jahren hat die Anzahl der fachdidaktischen Veröffentlichungen sowie Forschungsvorhaben zur informatischen Bildung in der Primarstufe zugenommen; siehe die aktuellen Arbeiten von Herper/Hinz [HH09], Borowski [BDM10, Bo14], Brumma [Br11], Batur/Bergner [BB12], Straube et al. [St13], Brakensiek [Br14], Klassen [Kl14], Schäffer/Mammes [SM15] u.a. International zeichnet sich ein ähnliches Bild ab. So stellt etwa die *Computer Science Teachers Association (CSTA)* fest: *„Recently, CSTA has witnessed a growing interest in computer science at the elementary and middle school levels"* [ACM12, S. V]. Auch außerhalb der Fachdidaktik ist ein steigendes Interesse zu erkennen. Beiträge in der c't [CT14], den Onlineausgaben des Handelsblattes [HAN14] oder der BBC [BBC14] sind Indikatoren hierfür und auch Internetbotschafterin Gesche Joost fordert ein konkretes Konzept zur digitalen Bildung ab der Grundschule [NOZ15]. Durch Schwills deduktive Studie „Ab wann kann man mit Kindern Informatik machen?" [Sc01] liegt bereits seit längerem eine theoretische Fundierung zur informatischen Bildung in der Primarstufe vor. Allerdings bleiben zur Ausarbeitung eines umsetzbaren Konzeptes eine Vielzahl von Fragen offen: Welche

[1] Westfälische Wilhelms-Universität Münster, Didaktik der Informatik, Fliednerstr. 21, 48149 Münster, alexander.best@uni-muenster.de
[2] Davertschule Amelsbüren, Städtische Katholische Grundschule, Zum Häppner 10, 48163 Amelsbüren, sarahmarggraf@gmx.de

Kompetenzen sollen gefördert werden? Welche Inhalte sollen vermittelt werden? Wann sollen sie vermittelt werden? Wie sollen sie vermittelt werden? Wo und von wem sollen sie vermittelt werden? Dieser Beitrag setzt sich mit der letztgenannten Fragestellung auseinander.

2 Konzeption und Ziele des Forschungsvorhabens

Von Seiten der Fachdidaktik wurde der Sachunterricht[3] an deutschen Grundschulen mehrfach als schulischer Ort informatischer Bildung vorgeschlagen (exemplarisch [BDM10]). Jedoch waren die Lehrer[4] im Fach Sachunterricht bislang kaum Gegenstand der Diskussion. Von ihnen wird aber erwartet, dass sie die Forderungen nach informatischer Bildung von Grundschülern in ihrem Unterricht umsetzen. Bereits im Rahmen des IniK-Projekts richteten Diethelm/Koubek/Witten die eindringliche Bitte zur Zusammenarbeit mit den Lehrkräften, etwa im Rahmen der sogenannten „Lehrersets", an die Fachdidaktik [DKW11, S. 100/101]. Auch bei der Entwicklung von Konzepten zur informatischen Bildung in der Primarstufe sollte dieser Forderung nach Ansicht der Autoren eine hohe Bedeutung beigemessen werden. Hierbei spielt besonders das Bild der Sachunterrichtslehrer zur Informatik eine wesentliche Rolle. Denn auf Grundlage dieses Bildes werden sie bereitgestellte Materialien auswählen sowie nutzen, Alltagsbezüge herstellen, Medien einsetzen, Lernziele fokussieren, Schüler anleiten und vieles weitere. Die Fachdidaktik sollte demnach das Ziel verfolgen den Lehrern im Fach Sachunterricht ein differenziertes Bild der Informatik zu vermittelt. Um dieses Ziel zu erreichen stellt sich vorab die Frage: Welches Bild der Informatik besteht derzeit unter Sachunterrichtslehrern?

Dieses Forschungsdesiderat wird im Forschungsprojekt zu „Informatik in der Grundschule" am Arbeitsbereich Didaktik der Informatik der Westfälischen Wilhelms-Universität Münster derzeit untersucht. Dieser Beitrag stellt die ersten Ergebnisse des Projekts dar, welches als dreischrittige Interventionsstudie angelegt ist. Nach Erfassung des Bildes der Informatik (IST-Zustand) von Sachunterrichtslehrern (Schritt 1) werden in Kooperation mit ihnen konkrete Unterrichtsplanungen und -materialien entwickelt (Schritt 2). Dabei soll auch untersucht werden, inwieweit bestehende Lehrpläne und Richtlinien des Faches Sachunterricht einen inhaltlichen und rechtlichen Rahmen bieten, informatische Inhalte sowie Kompetenzen in ihnen zu verankern. Abschließend wird erfasst, ob und wie sich das Bild der Informatik bei den Sachunterrichtslehrern durch die Auseinandersetzung mit diesen Unterrichtskonzepten verändert hat (Schritt 3). Ziel des Vorhabens ist es, praxisnahe und -erprobte Module zu

[3] In den meisten Bundesländern als Sachunterricht bezeichnet, gibt es abweichend in Schleswig-Holstein und Bayern die Bezeichnung „Heimat- und Sachunterricht" (HUS) sowie in Baden-Württemberg „Mensch, Natur und Kultur" (MeNuK). In diesem Beitrag wird durchgängig der Begriff Sachunterricht verwendet, da die durchgeführte Befragung an nordrhein-westfälischen Grundschulen stattfand. An Hochschulen wird bundesweit die Bezeichnung Sachunterricht verwendet.

[4] In diesem Beitrag wird aus Gründen der besseren Lesbarkeit die männliche Form verwendet. Sie schließt die weibliche Form stets mit ein.

konzipieren, welche von den Lehrkräften selbstständig durchgeführt werden können und ein differenziertes Bild der Informatik vermitteln. Es baut dabei auf zwei am Arbeitsbereich Didaktik der Informatik der WWU Münster entstandenen Masterarbeiten von Brumma [Br11] und Klassen [Kl14] auf, welche erste Ergebnisse bei der kooperativen Unterrichtsgestaltung zwischen den Studenten mit Sachunterrichtslehrern im Bereich „Informatik im Sachunterricht“ liefern.

In diesem Beitrag soll das Bild der Informatik von Sachunterrichtslehrern unter folgendem ausgewählten Aspekt untersucht werden:

> **h_1: Sachunterrichtslehrer verwenden die Begriffe Informatik und Medienbildung synonym**

Diese Hypothese wurde von Borowski [Bo14, S. 34] aufgestellt und kann mit der durchgeführten Umfrage nun exemplarisch überprüft werden.

3 Methodik und Auswertung der Umfrage

3.1 Design der Umfrage

Da den Autoren bislang keine empirischen Studien zu Vorstellungen von Sachunterrichtslehrern zur Informatik bekannt sind, wurde diese Umfrage qualitativ und explorativ angelegt. So kann zunächst ein Fundus an existierenden Vorstellungen und Äußerungen gesammelt werden. Aus diesem Fundus können für spätere Untersuchungen konkrete Merkmale und Operationalisierungen abgeleitet werden. Die durchgeführte Studie dient sowohl der Hypothesengenerierung als auch der Hypothesenüberprüfung (h_1).

Die Umfrage enthielt die folgenden vier Fragen:

#	Frage	Fragetyp
1	Sind Sie der Ansicht, dass Informatik für die gegenwärtige Lebenswirklichkeit Ihrer Schülerinnen und Schüler eine Rolle spielt?	Geschlossen (Ja/Nein)
2	Warum sind Sie der Ansicht, dass Informatik für die gegenwärtige Lebenswirklichkeit Ihrer Schülerinnen und Schüler eine Rolle bzw. keine Rolle spielt?	Offen
3	Was könnten Ihrer Ansicht nach mögliche Ziele und Inhalte von Informatik im Sachunterricht sein?	Offen
4	Könnten Sie sich vorstellen, informatische Inhalte im Rahmen des Sachunterrichts selbstständig zu vermitteln?	Geschlossen (Skala [1-5])

Tab. 1: Fragen und Fragetypen der durchgeführten Umfrage

Darüber hinaus wurde erfasst, an welcher Schule die Befragten tätig waren. Dies könnte Aufschluss darüber geben, ob etwa die Arbeit an Grundschulen mit MINT-Profil einen Einfluss auf das Bild der Informatik der Befragten hat. Zudem wurden die Unterrichtsfächer der Befragten erfasst. Hier ließe sich bspw. erkennen, ob Sachunterrichtslehrer mit dem zusätzlichen Unterrichtsfach Mathematik über ein anderes Bild der Informatik verfügen, als diejenigen Lehrkräfte, welche das Fach Mathematik nicht unterrichten. Die Auswertung der genannten Merkmale ist jedoch nicht Teil dieses Beitrages. Name und Geschlecht der Befragten wurden nicht erfasst.

Zur Darstellung und Diskussion der Antworten auf die offenen Fragen zwei und drei, wurde auf die Analyseform der induktiven Kategorienbildung nach Mayring zurückgegriffen [May15, S. 86]. Die finalen Kategorien sind das Produkt mehrmaliger Revisionen.

Borowski begründet seine Forderung eines Bezugs von informatischen Inhalten zur Medienbildung, Medienerziehung und Medienpädagogik in der Grundschule mit der Aussage, dass „bei Lehrerinnen und Lehrern häufig informatische Bildung und Medienbildung synonym benutzt“ [Bo14, S. 34] werde. Weiterhin weist er darauf hin, dass auch in der fachdidaktischen Literatur des Sachunterrichts eine „starke Fokussierung auf den Bereich Medium und Werkzeug“ [BDM10, S. 6] erkennbar sei. Beim Design der Umfrage wurde deshalb ausschließlich der Begriff „Informatik“ bzw. „informatische Inhalte und Methoden“ verwendet; ein Bezug zur Medienbildung fand nicht statt. Sollten von der Mehrheit der Befragten zu den Fragen drei und vier eigenständig Bezüge zur Medienbildung und nicht zur Informatik hergestellt werden kann Borowskis Aussage (h_1) für die Stichprobe als zutreffend betrachtet werden. Darüber hinaus wurden keine konkreten informatischen Inhalte oder Methoden genannt, sodass davon ausgegangen werden kann, dass das existierende Bild der Sachunterrichtslehrer durch die Umfrage nicht beeinflusst wurde.

3.2 Darstellung der Umfrageergebnisse

Die durchgeführte Umfrage fand im Januar 2015 statt. Es wurden insgesamt 423 Grundschulen im Regierungsbezirk Münster (NRW) elektronisch angeschrieben. Davon konnten 52 Anschreiben auf Grund von überfüllten Postfächern nicht zugestellt werden. Das Anschreiben erreichte demnach 371 Grundschulen. Es beteiligten sich 27 Lehrer an der Umfrage, welche alle das Fach Sachunterricht an ihrer Schule unterrichteten. Diese verteilten sich wiederum auf 13 Grundschulen. Dies entspricht einer Schulbeteiligung von 3,1% insgesamt und 3,5% unter Berücksichtigung der erfolgreich zugestellten Anschreiben an die Sekretariate. Von den Umfragebögen wurden 11 Exemplare digital ausgefüllt und elektronisch an den Arbeitsbereich Didaktik der Informatik zurückgesandt. 16 Befragte zogen es vor, die Umfragebögen per Hand auszufüllen und diese dem Arbeitsbereich postalisch zukommen zu lassen. Die durchschnittliche Beteiligungsquote lag bei zwei Teilnehmern pro Schule.

Die folgende Tabelle stellt die Ergebnisse der ersten Frage „Sind Sie der Ansicht, dass Informatik für die gegenwärtige Lebenswirklichkeit Ihrer Schülerinnen und Schüler eine Rolle spielt?“ dar:

	Befragte		Schulen		
	absolut	in %	absolut	gewichtet	in %
Ja	20	74	13	12,2	94
Nein	7	26	1	0,8	6

Tab. 2: Einschätzung der Befragten zum Lebensweltbezug von Informatik für SuS[5] in der Grundschule (Frage 1)

In nachfolgender Tabelle werden die Ergebnisse auf die zweite Frage „Warum sind Sie der Ansicht, dass Informatik für die gegenwärtige Lebenswirklichkeit Ihrer Schülerinnen und Schüler eine Rolle bzw. keine Rolle spielt?“ dargestellt und den induktiv gewonnen Kategorien zugeordnet. Die grau hinterlegten Zellen repräsentieren Kategorien und Einzelnennungen, welche mit der Antwort „Nein“ auf Frage eins korrelieren:

	Befragte (1) und Nennungen (2)		
	absolut	in % (1)	in % (2)
Außerschulischer Kontakt der SuS mit Computern muss aufgegriffen und die SuS in deren Bedienung geschult werden	10	37	21
Medienbildung, nicht Informatik, ist für SuS in der Grundschule relevant	6	22	13
Bedeutung von Computer-, Video- und Internetspielen im Alltag der SuS	5	19	11
Verantwortungsvoller, kritischer und produktiver Umgang mit dem Computer muss erlernt werden	5	19	11
Bedeutung von Mediengestaltung für Schule und Beruf	4	15	9
Relevanz der Informatik für Beruf, Schule und Alltag	4	15	9
Bedeutung von konkreten Softwareanwendungen für Schule und Beruf	3	11	6
Relevanz der Internetrecherche in Schule und Beruf	3	11	6
Kritische Nutzung sozialer Netzwerke	2	7	4
„PC entzaubern“	1	3	2
„Technische Informatik“	1	3	2

[5] Schülerinnen und Schüler

„SuS überfordert“	1	3	2
„SuS nur Konsumenten“	1	3	2
Keine Angabe	1	3	2

Tab. 3: Gründe für bzw. gegen einen vorhandenen Lebensweltbezug von Informatik bei SuS in der Grundschule nach Einschätzung der Befragten (Frage 2)

Für Einzelnennungen wurde keine Kategorie erstellt, sondern die Antworten als wörtliche Zitate übernommen. Die Antworten, welche der Kategorie „Bedeutung der Informatik für Beruf, Schule und Alltag“ zugeordnet wurden, ließen keine Schlüsse auf konkrete Inhalte oder Methoden zu. Informatik wurde hier von den Befragten als Schlagwort verwendet.

In folgender Tabelle werden die Ergebnisse auf Frage drei „Was könnten Ihrer Ansicht nach mögliche Ziele und Inhalte von Informatik im Sachunterricht sein?“ dargestellt:

	Befragte (1) und Nennungen (2)		
	absolut	in % (1)	in % (2)
Anwendungsschulungen (bspw. Microsoft Word, Microsoft PowerPoint, Browser)	13	48	18
Umgang mit dem PC (Tippen, Abspeichern von Dokumenten, etc.)	10	37	14
Kritische Mediennutzung	6	22	8
Internetrecherche	6	22	8
Sicherheit und Gefahren beim Einsatz von Software sowie des Internets	5	19	7
Soziale Netzwerke und Cybermobbing	5	19	7
Programmieren	4	15	5
Lernspiele/-programme	3	11	4
„Bildbearbeitung“	1	4	1,25
„Abspielen von Inhalten“	1	4	1,25
„Rechneraufbau/-komponenten“	1	4	1,25
„Briefkontakte analog und digital aufbereiten“	1	4	1,25
„andere Perspektiven Ingenieurwesen“	1	4	1,25
„Software testen“	1	4	1,25
„Netzwerke und WLAN“	1	4	1,25
„Technische Informatik“	1	4	1,25
Diffuse Antworten	6	22	8
Keine Angabe	8	30	11

Tab. 4: Mögliche Ziele und Inhalte von Informatik im Sachunterricht nach Einschätzung der Befragten (Frage 3)

Die unter der Kategorie „Diffuse Antworten“ zusammengefassten Nennungen sind: „Grundlagenkenntnisse vermitteln„ „Chancen + Risiken von Informatik erkennen“, „Kinder für Informatik sensibilisieren“, „Nutzen der Informatik für Schule, Beruf und

Alltag vermitteln“ sowie „Möglichkeiten und Gefahren der Informatik erkennen“. Diese Äußerungen sind dadurch charakterisiert, dass die Informatik hier als Schlagwort verwendet und mit keinen Inhalten oder Methoden verknüpft wurde.

Auf die abschließende Frage vier, ob die Befragten sich vorstellen könnten, informatische Inhalte im eigenen Sachunterricht zu vermitteln, ergab sich folgendes Bild:

	Befragte		Schulen		
	absolut	in %	absolut	gewichtet	in %
Trifft ganz zu	3	11	3	2,5	19
Trifft zu	8	30	6	4,1	32
Neutral	7	26	7	5,3	41
Trifft nicht zu	0	0	0	0	0
Trifft gar nicht zu	8	30	2	0,9	7
Keine Angabe	1	3	1	0,2	1

Tab. 5: Selbsteinschätzung der Befragten zur Vermittlung informatischer Inhalte im eigenen Sachunterricht (Frage 4)

3.3 Diskussion und Interpretation der Umfrageergebnisse

Es wurde in den Umfrageergebnissen deutlich, dass die Mehrheit der Befragten der Informatik einen Lebensweltbezug für Schüler in der Grundschule zuspricht. Dieses Ergebnis ist zunächst positiv zu bewerten es muss aber eingeschränkt und hinterfragt werden. So zeigte sich, dass von Seiten einer Schule eine starke Ablehnung gegenüber der Informatik in der Primarstufe bestand. Ob es sich hierbei um einen Einzelfall handelt, müsste in einer größeren Stichprobe untersucht werden. Auch über die Ursachen für diese Ablehnung können keine verlässlichen Aussagen getroffen werden. Es wurden jedoch von einigen Befragten konkrete Gründe genannt. So gab ein Befragter an, dass informatische Inhalte die Schüler überfordern würden. Auch der Einwand, dass in der Grundschule lediglich Medienbildung relevant sei, wurde geäußert. Insbesondere bei den Befragten, welche die Frage eins mit „Nein“ beantworteten und keine Angaben zu den Fragen zwei und/oder drei machten, könnten auch affektive Ursachen eine Rolle für ihr Abstimmverhalten gespielt haben. Beispiele wären etwa eine ablehnende Haltung gegenüber der Informatik oder gegen neue, fachfremde Inhalte. Diese Vermutungen können jedoch anhand der durchgeführten Umfrage nicht überprüft werden.

Die Umfrage ergab weiterhin, dass die Befragten die Gründe für den Lebensweltbezug von Informatik überwiegend im frühen Kontakt der Grundschüler mit Computern sahen. Einige Teilnehmer gaben explizit an, dass dieser erste, meist außerschulische, Kontakt sich in den vergangen Jahren deutlich verfrüht habe. Dieser von der Fachdidaktik häufig herangezogene Aspekt zur Legitimation von informatischer Bildung in der Grundschule

[exemplarisch Br11, S.4], scheint somit auch von den Lehrkräften geteilt zu werden. Während die Fachdidaktik hier allerdings von Informatiksystemen ausgeht, beschränken sich die Lehrkräfte auf den Computer.

Insbesondere bei den Ergebnissen der Frage drei zeigte sich, dass die Befragten die Inhalte und Ziele der Informatik mit der Anleitung zur Bedienung spezifischer Software und Hardware gleichsetzen. Positiv hervorzuheben ist hierbei, dass die Befragten neben der Anleitung zur Recherche im Internet auch die „Sicherheit und Gefahren beim Einsatz von Software und des Internets“ sowie „Programmierung“ als Inhalte und Ziele der Informatik benannten.

Positiv festzuhalten ist weiterhin, dass 41% der Befragten sich zutrauen würden, informatische Inhalte in ihrem Sachunterricht zu vermitteln. Da die 30% der Befragten, die dies strikt ablehnten, an einer einzigen Schule tätig waren, müsste eine größere Stichprobe herangezogen werden, um hieraus eine konkrete Aussage ableiten zu können. Ob es weitere schulspezifische Tendenzen zu dieser Frage gab, konnte nicht festgestellt werden. Die Beteiligungszahlen pro Schule waren hierfür zu gering.

Der Aussagecharakter der Fragen eins und vier ist zudem zu hinterfragen. So wird aus den Ergebnissen der Fragen zwei und drei deutlich, dass von der Mehrheit der Befragten Informatik mit Medienbildung gleichgesetzt wird. Dies führt dazu, dass die Fragen nach dem Lebensweltbezug (Frage 1) und nach der Selbsteinschätzung zur Vermittlung informatischer Inhalte im eigenen Unterricht (Frage 4) sich auch auf die Medienbildung beziehen. Dies bedeutet, dass informatische Inhalte und Methoden für die wenigsten Befragten als Gründe für die Antworten auf diese Fragen als Ursachen geltend gemacht werden können. Da diese Fragen jedoch auch einen affektiven Charakter haben, welcher nicht von der synonymen Verwendung von Medienbildung und informatischer Bildung abhängig ist, wird gegenüber der Informatik eine überwiegend positive Einstellung erkennbar. Dies wird in den Ergebnissen auf die Fragen eins und vier deutlich. Diese positive Grundhaltung der Informatik seitens der Befragten sollte, trotz der mangelnden inhaltlichen Differenzierung, eine hohe Bedeutung beigemessen werden. Die Chancen und die Akzeptanz für eine erfolgreiche Umsetzung zur Vermittlung informatischer Kompetenzen und Inhalte in der Grundschule erhöhen sich dadurch. Es wird in Tabelle 3 jedoch auch erkennbar, dass Vorbehalte bezüglich einer zu hohen Komplexität informatischer Inhalte für Grundschüler existieren.

4 Reflexion und Ausblick

Durch die geringe Anzahl der Umfrageteilnehmer lassen sich aus den Umfrageergebnissen der Stichprobe keine Rückschlüsse verallgemeinern. Zudem könnte selbst bei einer größeren Stichprobe nur eine Aussage bezüglich Nordrhein-Westfalens getroffen werden. In der Stichprobe wird die Hypothese h_1 bestätigt. Sowohl in den Fragen zur Bedeutung der Informatik für die Lebenswelt ihrer Schüler wie auch bei der Frage nach möglichen Inhalten und Zielen von Informatik im Sachunterricht wurde

hauptsächlich der Einsatz – wenn auch kritisch und reflektiert – von Computern sowie Softwareanwendungen als Medium und Werkzeug genannt.

Aus den Ergebnissen der Umfrage sollen als Ausblick für das Forschungsvorhaben weitere Hypothesen abgeleitet werden:

1. Eine begriffliche und inhaltliche Trennung zwischen Informatik und Medienbildung ist Voraussetzung dafür, dass das Bild der Informatik unter Sachunterrichtslehrern ausdifferenziert werden kann.

2. Bei der Entwicklung eines Konzepts zur informatischen Bildung in der Grundschule stellt die Zusammenarbeit mit den Lehrern ein Schlüsselelement für die Fachdidaktik dar.

Literaturverzeichnis

[ACM12] Computer Science K-8. Building a Strong Foundation. Erarbeitet von der Computer Science Teachers Association (CSTA) und Association for Computing Machinery (ACM). ACM Press, New York, 2012, http://csta.acm.org/Curriculum/sub/CurrFiles/CS_K-8_Building_a_Foundation.pdf, 28.01.2015.

[BB12] Batur, F.; Bergner, N.: Grundschulkinder begeistern mit der Zauberschule Informatik. In (Thomas, M.; Weigend, M. Hrsg.): Ideen und Modelle. 5. Münsteraner Workshop zur Schulinformatik, 7. Mai 2012 an der Westfälischen Wilhelms-Universität Münster. Books on Demand GmbH, Norderstedt, S. 87-94.

[BBC14] Computer coding taught in Estonian primary schools, http://www.bbc.com/news/education-25648769, 28.01.2015.

[BDM10] Borowski, C.; Diethelm, I.; Mesaroş, A.: Informatische Bildung im Sachunterricht der Grundschule. Theoretische Überlegungen zur Begründung. In (Pech, D. Hrsg.): widerstreit-sachunterricht, Ausgabe 15, Oktober 2010, http://www.widerstreit-sachunterricht.de/ebeneI/superworte/infor/BorDieMe.pdf, 28.01.2015.

[Bo14] Borowski, C.: Informatik in der Grundschule: Kriterien für eine erfolgreiche Umsetzung. In (Brakensiek, J.): Informatik in der Grundschule – Welche Kompetenzen sind bereits in der Primarstufe von Relevanz? Bochum, 2014, S. 31-35, http://www.ham.nw.schule.de/pub/bscw.cgi/d5028374/Brakensiek_Forschungsbericht.pdf, 28.01.2015.

[Br11] Brumma, J.: Konzeption von Unterrichtsmodulen zur Vermittlung kerninformatischer Inhalte in der Grundschule. Münster, 2011, http://ddi.uni-muenster.de/ab/pu/dok/Masterarbeit_Jens_Brumma_op.pdf, 28.01.2015.

[Br14] Brakensiek, J.: Informatik in der Grundschule – Welche Kompetenzen sind bereits in der Primarstufe von Relevanz? Bochum, 2014,

http://www.ham.nw.schule.de/pub/bscw.cgi/d5028374/Brakensiek_Forschungsbericht.pdf, 28.01.2015.

[CT14] Schulfach „Computing“ ab Klasse 1 – Interview mit Simon Peyton Jones, http://shop.heise.de/katalog/schulfach-computing-ab-klasse-1-interview-mit-simon-peyton-jones, 28.01.2015.

[DKW11] Diethelm, I.; Koubek, J.; Witten, H.: IniK – Informatik im Kontext. Entwicklungen, Merkmale und Perspektiven. In (Koerber, B. Hrsg.): LOG IN 169/170, S. 97-105.

[HAN14] Algorithmen in der Grundschule, http://www.handelsblatt.com/10320802.html, 28.01.2015.

[HH09] Herper, H.; Hinz, V.: Informatische Bildung im Primarbereich. In (Koerber, B. Hrsg.): Zukunft braucht Herkunft. 25 Jahre „INFOS - Informatik und Schule“. INFOS 2009. 13. GI-Fachtagung Informatik und Schule, 21.-24. September 2009 in Berlin. Köllen Verlag + Druck, Bonn, S. 74-85.

[NOZ15] Internetbotschafterin für Programmier-Unterricht an Grundschulen, http://www.noz.de/deutschland-welt/medien/artikel/534634, 28.01.2015.

[Kl14] Klassen, I.: Informatik in der Grundschule am Beispiel eines Moduls zur Funktionsweise von Computern. Münster, 2014, http://ddi.uni-muenster.de/ab/pu/dok/Masterarbeit_Klassen_2014.pdf, 28.01.2015.

[Ma15] Mayring, P.: Qualitative Inhaltsanalyse. Grundlagen und Techniken. Beltz Verlag, Weinheim und Basel, 2015.

[Sc01] Schwill, A.: Ab wann kann man mit Kindern Informatik machen? Eine Studie über informatische Fähigkeiten von Kindern. In (Keil-Slawik, R; Magenheim, J. Hrsg.): Informatikunterricht und Medienbildung. INFOS 2001. 9. GI-Fachtagung Informatik und Schule, 17.-20. September 2001 in Paderborn. Köllen Verlag + Druck, Bonn, S. 13-30.

[SM15] Schäffer, K.; Mammes, I.: Zur Bedeutung informatischer Bildung in der Grundschule – Das Konstrukt des informatischen Verständnisses von Grundschülern. In (Blömer, D. et al. Hrsg.): Perspektiven auf inklusive Bildung. Gemeinsam anders lehren und lernen. Springer VS, Wiesbaden, S. 174-180, 2015.

[St13] Straube, P. et al.: DoInG – Informatisches Denken und Handeln in der Grundschule. In (Groetzebauch, H.; Nordmeier, V. Hrsg): PhyDid B – Didaktik der Physik – Beiträge zur DPG-Frühjahrstagung, Jena 2013, http://phydid.physik.fu-berlin.de/index.php/phydid-b/article/view/422, 28.01.2015.

Wir bedanken uns bei allen an der Umfrage teilgenommen Grundschulen sowie Sachunterrichtslehrern.

Agile Softwareentwicklung – Erfahrungsbericht eines Oberstufenprojekts im Wahlpflichtunterricht

Peter Brichzin[1]

Abstract: Der Praxisbericht zeigt wie die konkreten Methoden Pair Programming, User Stories, Tasks, Task Board und Standup Meeting in einem Softwareentwicklungsprojekt der Oberstufe umgesetzt wurden, welchen Beitrag sie zum Projekterfolg geleistet haben und welche Erfahrungen für das Lehren daraus gewonnen wurden. Letzteres beinhaltet auch einen Blick auf die didaktische Reduktion der Methoden, denn im Vergleich zu agiler Prozesssteuerung aus dem Lehrbuch wurde bewusst auf Methoden wie Schätzen und typische Rollen wie den Scrum Master verzichtet. Trotz anspruchsvoller Projektthemen überzeugten die Ergebnisse der Schülerinnen und Schüler im Vergleich zu den nicht agil erarbeiteten Projektergebnissen der gleichen Zielgruppe aus den Vorjahren. Wesentlicher Schlüssel zum Erfolg war einerseits das Prototyping und andererseits die positive Unterstützung kollaborativen Arbeitens durch die oben genannten agilen Methoden.

Keywords: agil, Softwareentwicklung, Projekt, Oberstufe, Erfahrungsbericht, kollaboratives Arbeiten

1 Agile Methoden lösen das Wasserfallmodell ab

Eines der ältesten Vorgehensmodelle in der Softwareentwicklung ist das Wasserfallmodell, welches sequentiell die Phasen Anforderungen, Entwurf, Implementation, Überprüfung und Wartung durchläuft. Obwohl es im Laufe der Zeit z. B. in Form vom erweiterten Wasserfallmodell und Spiralmodell weiterentwickelt wurde, hat es in der IT-Branche heute nur noch eine geringe Bedeutung. In den letzten Jahren haben immer mehr Unternehmen zu agilen Methoden gewechselt. Diese verringern den bürokratischen Aufwand, unterstützen eine kontinuierlichen Interaktion mit dem Kunden sowie eine iterative Softwareentwicklung mit hoher Flexibilität bei der Festlegung der nächsten Schritte (vgl agiles Manifest agile Manifest [Be01]). Bei einer im Herbst 2012 von VersionOne durchgeführten Umfrage, gaben 84% der Befragten an, dass in Ihren Unternehmen agile Prozesse eingesetzt werden [Ve12, S.6].

Die Ursachen hierfür sind vielfältig, drei seien hier herausgegriffen:
Bei Projekten mit einer Dauer von mehr als einem halben Jahr, ergeben sich in der Regel durch neue Entwicklungen – auf dem Markt, im Unternehmen, bei angebundenen Softwaresystemen usw. – auch neue Anforderungen an die zu erstellende Software. Agile Methoden sind im Gegensatz zum Wasserfallmodell interaktiv und flexibel in allen Phasen, auf Veränderungen kann schnell reagiert werden.

[1] QAware GmbH, Aschauer Str. 32, 81549 München, peter.brichzin@qaware.de und
LMU München, Institut für Informatik, Oettingenstr. 67, 80538 München, brichzin@tcs.ifi.lmu.de

Findet die Kommunikation zwischen Kunden und Softwaredienstleister (hauptsächlich) nur in der Anforderungserhebung statt, so besteht die Gefahr bei Missverständnissen ein Produkt zu entwickeln, das nur teilweise den Vorstellungen des Kunden entspricht, Änderungen sind in einer späten Phase sehr teuer. Bei einer agilen Entwicklung erhält der Kunde in regelmäßigen Abständen (4 - 12 Wochen) einen Prototypen. Dadurch kann er eigene Erwartungen mit dem Produkt vergleichen und Änderungen hinsichtlich der Anforderungen oder der Priorisierung einsteuern. Nicht zuletzt erhöht sich bei einer agilen Softwareentwicklung die Motivation und Produktivität im Team, da diesem mehr Eigenverantwortung und Selbstorganisation bei Planung und Umsetzung zugestanden wird.

Im Anschluss an die INFOS 2013 hat sich ein bundesweiter Arbeitskreis formiert, der den Gewinn agiler Methoden im Informatikunterricht diskutiert und Scrum Methoden zu Methodenbausteinen für den Schulunterricht didaktisch reduziert hat. Zwei Jahre später zeigen verschiedene Praxisberichte durchweg positive Erfahrungen. Dieser Artikel hier beschreibt den Erfolg agiler Methoden in einem Informatikprojekt in Jahrgangsstufe 11 an einem bayerischen Gymnasium.

2 Rahmenbedingungen

Im Lehrplan der Jahrgangsstufe 11 ist ein Projekt zur Softwaretechnik mit ca. 26 Unterrichtsstunden verankert. Die Schülerinnen und Schüler bringen als Vorwissen einerseits aus Jahrgangsstufe 10 Grundlagen zur Objektorientierten Modellierung und Programmierung mit, andererseits aus Jahrgangsstufe 11 praktische und theoretische Kenntnisse zu den Datenstrukturen Liste, Baum und Graph [LP09]. Programmierung grafischer Benutzeroberflächen, Datenbankanbindung, das Entwurfsmuster MVC und vertiefte Methoden der Projektorganisation sind durch den Unterricht vor dem Projekt noch nicht bekannt, werden aber im Lehrplan gefordert. Dadurch reduziert sich, bei drei Stunden Unterricht pro Woche, die Projektarbeitszeit auf 6 – 7 Wochen.

Als Programmiersprache wurde wie auch im Unterricht Java verwendet. Als Entwicklungsumgebung wurde im Unterricht BlueJ verwendet. Im Projekt verwendete ein Teil der Schüler Eclipse bzw. NetBeans.

Der Kurs bestand aus 21 Schülerinnen und Schülern. Die Leistungsstärke des Kurses war sehr inhomogen. Einige Schüler brachten bereits Erfahrungen z. B. in der Programmierung graphischer Benutzeroberflächen und Verwendung professionellerer Entwicklungsumgebungen ein, andere hatten Schwierigkeiten auch kleine Teilaufgaben innerhalb des Teams selbstständig zu lösen.

3 Agile Methoden in der Softwareentwicklung

3.1 Agiles Manifest

Es gibt eine Vielzahl unterschiedlicher Vorgehensweisen Software agil zu entwickeln, beispielsweise Scrum, KanBan, Extreme Programming. Da die Zielsetzung des Artikels der Einsatz in der Schule ist, sei hinsichtlich des Einsatzes agiler Methoden in Unternehmen auf Literatur wie z. B. [Pi11][An07] bzw. [HeRoLi05] verwiesen. Gemeinsam ist allen Vorgehensweisen das agile Manifest [Be01] mit vier Werten und 12 Prinzipien. Von den Prinzipien sind folgende für die Schule besonders relevant:

- „Lieferung von funktionierender Software in regelmäßigen, bevorzugt kurzen Zeitspannen“
- „Bereitstellung des Umfeldes und der Unterstützung, welche von motivierten Individuen für die Aufgabenerfüllung benötigt wird“
- „Informationsübertragung nach Möglichkeit im Gespräch von Angesicht zu Angesicht“
- „Als wichtigstes Fortschrittsmaß gilt die Funktionsfähigkeit der Software“
- „Einfachheit ist essenziell (KISS-Prinzip)“
- „Selbstorganisation der Teams bei Planung und Umsetzung“

3.2 Methodenauswahl

Die für das Oberstufenprojekt ausgewählte Vorgehensweise orientiert sich an Scrum, jedoch wurde die Anzahl der Methoden, Rollen und Ereignisse deutlich reduziert und adaptiert. Folgende Aspekte der Projektorganisation wurden verwendet:

- **Prototyping – iterativ inkrementelle Softwareentwicklung:**
  Regelmäßiger Meilenstein (iterativer Prozess) ist ein lauffähiger und getesteter Prototyp, der mehr Funktionalitäten bietet, als der vorhergehende Prototyp (inkrementell). Jedes Inkrement umfasst dabei alle Schichten (siehe Abb. 1) Die Schüler sehen dadurch nicht nur schrittweise ihre Software wachsen, sondern die Produktausrichtung und Reihenfolge der umzusetzenden Funktionalitäten ist flexibel. Weiterhin werden ungenaue Schnittstellenabsprachen zwischen Teilgruppen des Teams frühzeitig erkannt und können korrigiert werden.

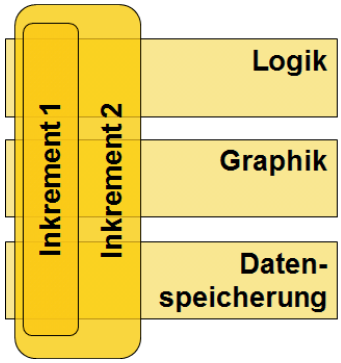


Abb. 1: Inkrementelle Softwareentwicklung über mehrere Schichten

- **Sprint:**
  Das Zeitintervall, in dem ein Prototyp erstellt wird, wird Sprint genannt. Der Sprint beginnt mit einer Planung, hat eine Implementierungsphase und schließt mit Tests ab. In dem Oberstufenprojekt wurde als Zeitintervall von 2 Wochen (6 Schulstunden) gewählt.

- **User Story:**
  Jede User Story beschreibt eine Anforderung aus der Sicht des Endanwenders, d.h. eine Funktionalität der Software. User Stories sind aufgeteilt in Titel und Beschreibung, so kurz formuliert, dass sie auf Karteikarte passen. Sie sind Ergebnis einer Teambesprechung, priorisiert und testbar. Bei der Formulierung einer User Story sind (informatische) Fachausdrücke und Implementierungsdetails verboten.

- **Modeling Story:**
  Für die Schülerinnen und Schüler ist einerseits die Motivation für eine lange Planungsphase gering, andererseits ein vollständigen Systementwurf wie im Wasserfallmodell zu anspruchsvoll. Eine Planung auf Ebene von User Stories (und Tasks) ist eine akzeptierte Grobplanung und kommt den Schülern entgegen, frühzeitig mit ersten Implementierungsschritten zu beginnen. Diese ersten Schritte sind im Lernprozess sehr wichtig, weil die Schüler selbst ein Gefühl erhalten, an welchen Stellen Schwierigkeiten auftreten bzw. ein schnelles Voranschreiten möglich ist (und nicht der Lehrer einen Großteil der konzeptionellen Arbeit abnimmt).
  Jedoch ist bei zunehmender Prototyp-Größe die Zahl der Klassen nicht mehr so einfach zu überblicken bzw. es kann notwendig sein, dass eine sehr zentrale Klasse von mehreren Paaren verändert werden muss. Es entsteht im Entwicklungsverlauf ein Bedarf, einen Überblick über die Klassen, deren Schnittstellen und Beziehungen zu bekommen. Als besondere User Story – Modelling Story - ist es an diesem Punkt sinnvoll, ein Klassendiagramm zu erstellen, welches die Grundlage für weitere Überlegungen und Besprechungen ist.

In der Regel muss der Lehrer den Anstoß dazu geben und die Modeling Story (in einem Standup-Meeting) einsteuern.

- **Task:**
  Tasks sind elementare Aufgaben aus der Sicht des Entwicklers, die durch das Unterteilung einer User Story entstehen Tasks sind aufgeteilt in Titel und Beschreibung, so kurz formuliert, dass sie auf ein „Post it“ passen. Sie sind Ergebnis einer Teambesprechung, testbar und in der Bearbeitung einem Paar zugeordnet.

- **Taskboard:**
  Die gesamte Planung erfolgt über ein im Computerraum aufgehängtes Taskboard (s. Abb. 2). Es dokumentiert über die Spalten „to do“, „in progress“, „in test“ und „done“ zu Erledigendes und bereits Abgeschlossenes, sowie wer aktuell an welchen Arbeitspaketen arbeitet. Auch ist das Sprintende als nächster Meilenstein deutlich vermerkt. Bei den Arbeitspaketen, deren Reihenfolge durch die Priorisierung bestimmt wird, gibt es zwei unterschiedliche Granularitäten: User Stories und Tasks. Niedrig priorisierte müssen nicht von Anfang an als Tasks ausgearbeitet werden.
  Als Taskboards wurden tragbare Tafelelemente bzw. Styroporplatten verwendet. Letztere können schnell auf- und wieder abgebaut werden. Dies ist hilfreich, da der Computerraum von vielen Klassen verwendet wird.

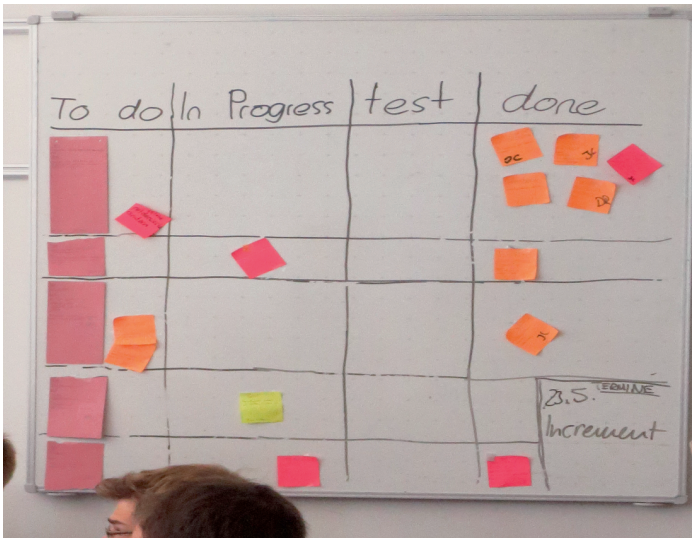


Abb. 2: Taskboard als zentrale Planungsübersicht

- **Programmieren im Pair:**
  Entwickelt wird paarweise, wobei folgende zwei Rollen mindestens einmal pro Unterrichtsstunde gewechselt werden: Der Driver verwendet Tastatur und Maus

und verfasst den Quelltext. Entscheidungen und Absichten werden dem Partner mündlich mitgeteilt. Der Navigator hinterfragt Quelltext, spricht mögliche Fehler an, und sucht elegantere Lösungen. Er hat die Aufgabe das große Ganze im Auge zu behalten.

- **KISS-Prinzip:**
  Die Zeit zum Implementieren ist selten ausreichend. Deshalb ist eine geringe Anzahl an Funktionalitäten (pro Iterationsschritt) und Einfachheit (der Implementierung) sehr wichtig, um sich nicht zu verzetteln: Keep It Small and Simple.

- **Standup-Meeting:**
  Jede Woche findet eine Besprechung im Stehen vor dem Taskboard statt. Dazu äußert sich jedes Teammitglied bzw. jedes Paar knapp und präzise welche Aufgaben es bearbeitet hat. Gegebenenfalls gibt es kurze Hinweise zum Lösungsweg (insbesondere wenn Schnittstellen betroffen sind) und zu aufgetretenen Problemen. Weiterhin nennt jeder den Task, den er als nächstes bearbeiten wird. Die maximale Dauer des Standup-Meetings ist 10 Minuten.
  Diese Besprechung ist wichtig für die Transparenz: Jeder erhält einen Überblick über die aktuellen Aktivitäten, bei der Kommunikation von Problemen können Hilfestellungen gegeben werden. Die Transparenz betrifft in erster Linie das Team. Aber auch für den Lehrer ist das Standup-Meeting ideal, um einen Überblick über den Projektstand zu erhalten.
  Das Standup-Meeting am Ende eines Sprints wird erweitert um die Planung des nächsten Prototyps. Dazu gehört gegebenenfalls eine neue Priorisierung der User Stories, Ergänzung von Tasks und die Aufteilung der Tasks an die Paare.
  Hinweis: Bei den Profis findet dieses Meeting täglich statt (Daily), in der Schule müsste man es Weekly nennen.

3.3 Nicht verwendete (Scrum) Methodenbausteine:

Da die Anzahl der Bearbeitertage für die Softwareerstellung in der Schule deutlich niedriger ist als in einem Unternehmen, muss der Aufwand für die Projektorganisation dort auch deutlich niedriger sein, um ein angemessenes Verhältnis zwischen Organisation und Implementierung zu erhalten. Folgende Standardelemente aus Scrum wurden aus diesem Grunde nicht eingeführt:

- **Schätzen - Planning Poker - Burn Down Chart:**
  Das Schätzen von Arbeitsaufwänden setzt Erfahrung voraus und ist für Schülerinnen und Schüler sehr schwer zu bewältigen. Die ist ein Grund, warum auf diese Methoden verzichtet wurde. Ein anderer ist, dass bei einer guten Aufteilung der User Stories in Tasks, für die Schüler "ein Task" sehr bald eine "greifbare" Zeiteinheit wird. Greifbar auch deshalb, weil nach Abarbeitung des Tasks, das entsprechende Post It auf dem Taskboard händisch verschoben wird.

- **Rollen Product Owner und Scrum Master:**
  Neben dem Team sind bei Unternehmensprojekten der Product Owner und der Scrum Master zwei weitere wichtige Rollen. Bei den vergleichsweise kleinen Projekten in der Schule, kann auf die letzten beiden genannten Rollen verzichtet werden. Da die Schüler sich i. A. selbst die Projektidee ausgedacht haben, übernehmen Sie indirekt Aufgaben des Prodct Owners. Sie führen Entscheidungen in der Gruppendiskussion herbei. Eine der Hauptaufgaben des Scrum Masters, (organisatorische) Hindernisse aus dem Weg zu räumen, sollte der Lehrer übernehmen. Er erfährt darüber im Daily.

4 Projekteinstieg und -durchführung

4.1 Einführung in Agile Softwareentwicklung

In Form eines Lehrervortrags erhielten die Schülerinnen und Schüler eine Einführung in die in Kapitel 3.2 beschriebenen agilen Methoden. Es war in der Projektdurchführung Pflicht diese Methoden zu einzusetzen.

Um das Prototyping begreifbar zu machen, wurde eine Marshmallow Challenge durchgeführt [Wu15]. Aufgabe dabei ist es, mit wenig Material (20 Spagetti, 0,5 m Schnur, 0,5 m Klebeband und 1 Marshmallow) in einer fest vorgegebenen Zeit von 15 Minuten ein möglichst hohes frei stehendes Gebäude zu bauen, dessen höchster Punkt der Marshmallow ist. Durch statistische Auswertungen dieses Wettbewerbs wurde belegt, dass der Erfolg dann größer ist, wenn in der Bauphase der Marshmallow bereits frühzeitig integriert wird, um die Stabilität zu überprüfen. Ausgehend von diesem Prototypen kann dann das Gebäude zum nächsten Prototypen mit einer größeren Höhe weiterentwickelt werden usw. Probleme haben die Gruppen, die erst in letzter Minute den Marshmallow positionieren, denn bei mangelnder Stabilität bleibt keine Zeit für Korrekturen, so dass als Endergebnis die Gebäudehöhe 0cm beträgt.

4.2 Themenwahl und Gruppeneinteilung

Vor Projektbeginn hatte jede Schülerin und jeder Schüler den Arbeitsauftrag, über einen Forumseintrag in der Lernplattform des Kurses ein Thema für das Projekt vorzuschlagen. Der Themenvorschlag musste eine Erläuterung in wenigen Sätzen bzw. eine Auflistung von Features enthalten. Die Vorschläge wurden vom Lehrer gruppiert, kommentiert und zur Abstimmung in den Kurs getragen. Die Vorschläge im Einzelnen waren:

- **Spiele:**
  Mario Spiel, Brettspiel TAC, Flappy Stein, Jump ‚n' Run, Black Jack, Poker,

Schafkopf, Space Invaders, Asteroids, Stein, Schere und Papier und Rubiks Cube Löser

- **Anwendungssoftware (ohne Spiele):**
  individualisierte Anzeige des Vertretungsplanes auf dem Handy/Computer, Terminverwaltungsprogramm privat bzw. schulisch (letzteres z. B. mit Klausurenplan, Hausaufgabenliste, Notizen und Noten), Kassensystem für Pausenverkauf, Mediensammlung, Datenverwaltungssoftware (z. B. für schulische Inhalte, wie Hefteinträge und Übungsaufgaben), Nachhilfevermittlung, Bibliothekssoftware für die Lehrmittelbibliothek (zusätzlicher Vorschlag vom Lehrer)

Der Lehrer stellte die Vorschläge vor, kommentierte teilweise (z.B. Nachhilfevermittlung ist nur sinnvoll als Webservice, welcher aber erst im Lehrplan der Jahrgangsstufe 12 Thema ist). Bei der anschließenden Abstimmung gab es in etwa drei gleich große Schülergruppen, mit den Themenwünschen Vertretungsplan (8), Spiel (6) und Bibliothekssoftware (7). Entsprechend der Wünsche wurden drei Teams gebildet. Vorteil dieser Vorgehensweise war, dass für die Themenwahl und Gruppeneinteilung stark auf die Schülerwünsche eingegangen wurde, jedoch nur 20 Minuten Unterrichtszeit benötigt wurden.

4.3 Projektdurchführung und -ergebnis

Die Durchführung erfolgte entsprechend der methodischen Vorgaben aus Kapitel 3.2. An das für die Schülerinnen und Schüler neue Pair Programming musste in der Anfangsphase mehrfach erinnert werden, spielte sich aber dann ein. Die interne Organisation konnte jedes Team frei gestalten. Hier ergab sich nicht die Notwendigkeit von Strukturen, denn sowohl die Aufgabenver- und die Paareinteilung erfolgte (über das Taskboard) problemlos und leistungsdifferenziert als auch die Gesprächsleitung in den Weeklies (und damit eine gewisse Leitungsfunktion) übernahm immer jemand initiativ. Jeder dokumentierte seinen Beitrag zum Projekt als knappe Einträge in einem Projekttagebuch.

Nach sechs Wochen wurde der dritte Prototyp zusammen mit dem Taskboard als Dokumentation im Plenum vorgestellt. Die Ergebnisse und die Selbstorganisation waren sehr unterschiedlich. Eine Gruppe hatte nur ein ausreichendes Projektergebnis, scheiterte an mangelhafter Kommunikation, schlechter Organisation (Absenzen von implementierungsstarken Schülern minderte nicht nur die „Manpower“ sondern blockierte auch das gesamte Team wegen fehlender Quelltexte) und nicht realistischer Selbsteinschätzung. Entsprechende mehrfache Hinweise seitens des Lehrers hinsichtlich Fehlentwicklungen, führten nicht wirklich zur Verbesserung. Z. B. blieb ein zusätzlich abzugebender detaillierten Arbeitsplans ein theoretisches Konstrukt und führte nicht wie erhofft zu realistische Umsetzungen. Die beiden anderen Gruppen hatten hervorragende Endergebnisse. Eine davon ausschließlich selbstorganisiert, getrieben von einem hohen Engagement der Mehrzahl der Mitglieder. Das Team glänzte z.B. durch selbst initiierte

Besprechungen, in denen Wissenstransfer von einzelnen Paaren zum Team betrieben wurde. Die andere Gruppe hat Höhen und Tiefen durchlaufen, jedoch Hilfestellungen vom Lehrer gewinnbringend aufgenommen und umgesetzt. Dazu gehörte insbesondere eine Modeling Story nach dem ersten Prototypen, die Klarheit in der Struktur und der weiteren Vorgehensweise sorgte. Ein Motivationsschub gaben hier positive Rückmeldungen des schulinternen Abnehmers der Software.

Ein wichtiger Beitrag für den Fortschritt bei den durchaus komplexen Themenstellungen sind „Hausaufgabenbeiträge“. Auch wenn Hausaufgaben wie im „nicht projektorientierten“ Unterricht erwartet werden dürfen, fällt das Engagement, ebenfalls wie im „normalen“ Unterricht, sehr unterschiedlich aus.

Aus Lehrersicht war die Betreuung dreier unterschiedlicher Themen inhaltlich und organisatorisch anspruchsvoll. Organisatorisch wurde das Weekly zeitversetzt abgehalten, so dass der Lehrer bei allen Gruppen teilnehmen konnte. Nur so ist es möglich einen Überblick zu haben und passend unterstützende Impulse einbringen zu können. Eine inhaltliche Betreuung fand nicht auf Detailebene statt, sondern auf abstrakteren Niveau bzw. mit dem Hinweis auf schülergerechte Quellen, z.B. eine Datenbankanbindung in [Br09 s. 169ff].

5 Erfahrungen und Ausblick

Das Oberstufenprojekt wurde bereits das vierte Mal durchgeführt, jedoch erstmals agil. Der Erfolg beim agilen Vorgehen war deutlich besser, einfach messbar an der entwickelten Software, die bisher nicht über eine Alpha-Version hinaus ging, in dem dargestellten Schuljahr jedoch in zwei Gruppen ein Release Niveau erreichte. Im Falle der Schulbibliothekssoftware wurde diese unmittelbar nach Projektende von der Lehrmittelbücherverwaltung eingesetzt. Zentrale Gründe für den Erfolg sind einerseits das Prototyping und andererseits eine Unterstützung des kollaborativen Arbeitens. Das Prototyping reduzierte deutlich Schnittstellenprobleme, die in den letzten Jahren zu hohen Reibungsverlusten geführt hatten und auf Grund begrenzter Zeit nicht mehr aufgefangen werden konnten. Alle anderen agilen Methoden schaffen ideale Bedingungen für ein konstruktivistisches Lernen: User Stories und Tasks sind bei geringem organisatorischen Aufwand eine angemessene Unterstützung der inhaltlichen Planung und Prozessorganisation, um mit selbstbestimmter flexibler Zielsetzung etwas Neues zu schaffen. Das Taskboard sorgt für Transparenz (u.a. auch über den Fortschritt – ein wichtiger Motivationsaspekt). Und die Eigenverantwortung im Team, unterstützt durch das Weekly fördert die Gruppendynamik und das Engagement.

Folgende Optimierungen könnten das agile Vorgehen noch gewinnbringender machen: Einzelne Techniken wie Pair Programming, User Stories/Tasks können schon im Vorfeld in den Unterricht erlernt und angewandt werden. Beim Pair Programming würde ein Wechsel der Partner noch zu mehr Kompetenztransfer führen. Jedoch steht diesem Vorteil ein höherer Zeitaufwand hinsichtlich der Einarbeitung in anderen Bereichen

gegenüber. Eine Unterstützung des kollaborativen Arbeitens durch Versionskontrollsysteme (vgl [BrRa15]) würde den Arbeitsaufwand beim Zusammenführen von Quelltext und andere Reibungsverluste deutlich reduzieren.

Wie das Ergebnis in Kapitel 4.3 zeigt, ist das agile Vorgehen auch kein Garant für Gelingen. (Zu) Hohe Ansprüche der Schüler an Ihre Software stehen in Widerspruch zu der (vielleicht üppig klingenden, aber) knappen Projektdauer von 7 Wochen. Drei Prototypen ist das Minimum um den agilen Prozess erfahrbar zu machen: Der erste Prototyp scheitert meist wegen der Schnittstellen, der zweite zeigt einen gangbaren Weg und mit jedem weiteren Prototypen werden dann zügig Features gebaut. Ein bis zwei Prototypen mehr würden den Schülern mehr Raum für einen Irrweg im Lernprozess sowie allen Beteiligten mehr Zufriedenheit und damit Motivation geben.

Insgesamt sind agile Methoden ideal zur Unterstützung kollaborativen Arbeitens in Projekten geeignet und werden deshalb hoffentlich in Zukunft häufiger in der Schule (und nicht nur im Informatikunterricht) eingesetzt.

Literaturverzeichnis

[An11] Anderson, D: Kanban: Evolutionäres Change Management für IT-Organisationen, dpunkt.verlag, Heidelberg, 2011.

[Be01] Beck, K. et al.: Manifesto for Agile Software Development, http://agilemanifesto.org/iso/de/ - (zuletzt geprüft am 27.4.2015).

[Br09] Brichzin, P.; Freiberger, U.; Reinold, K.; Wiedemann, A.: Informatik Oberstufe 1, Objektorientierte Modellierung. Oldenbourg Verlag, München, 2009.

[BR15] Brichzin, P.; Rau, T: Repositories zur Unterstützung von kollaborativen Arbeiten in Softwareprojekten, Tagungsband INFOS 2015, Darmstadt, 2015.

[HRL05] Wolf, H.; Roock, S.; Lippert, M: eXtreme Programming: Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis, dpunkt.verlag, Heidelberg, 2005.

[LP09] Lehrplan Informatik für das Gymnasium in Bayern unter http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/index.php?StoryID=26193&PHPSESSID=e0e818ca1f0e9ec5aca6792e060a29d9 (zuletzt geprüft am 31.01.15).

[Pi07] Pichler, R.: Scrum - Agiles Projektmanagement erfolgreich einsetzen, dpunkt.verlag, Heidelberg, 2007.

[Ve12] Version One, 7th Annual State of Agile Development Survey unter http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf (zuletzt geprüft am 17.1.15) .

[Wu15] Wujec, T.: The Marshmallow Challenge, http://www.marshmallowchallenge.com (zuletzt geprüft am 27.4.2015).

Repositories zur Unterstützung von kollaborativen Arbeiten in Softwareprojekten

Peter Brichzin[1] und Thomas Rau[2]

Abstract: In der professionellen Softwareentwicklung ist die Verwendung eines Repositoriums mit Versionskontrolle ein Standardwerkzeug, um verteilt an gemeinsamen Projekten zu arbeiten. Auch bei Softwareprojekten im Informatikunterricht der Oberstufe unterstützt der Einsatz kollaboratives Arbeiten, etwa beim regelmäßigen Zusammenfügen (zu Prototypen), bei der Verfügbarkeit des Quelltexts auch bei abwesenden Teammitgliedern und bei der Dokumentation der Arbeit. In diesem Aufsatz werden praktische Unterrichtsszenarien mit der Entwicklungsumgebung BlueJ und dem dort integrierten svn-Client exemplarisch vorgestellt.

Keywords: Kollaboratives Arbeiten, Subversion, Softwareentwicklung, Projekt, Java, BlueJ, agil

1 Über die Notwendigkeit von Versionskontroll-Systemen

1.1 Nöte in der Dateiverwaltung bei der Softwareentwicklung

In der bekannten Webcomic-Serie xkcd gibt es einen Cartoon, der sich über die Namensgebung in Dateiverzeichnissen lustig macht (s. Abb. 1). Ähnlich sieht das auch in den Quelltextordnern von Schülerinnen und Schülern aus – dort heißen die gespeicherten Dateien beispielsweise „Game1“, „Game1 (Kopie)“ und „Game2 (funktioniert)“. Es ist verständlich und nützlich, Sicherheitskopien im Laufe der Softwareentwicklung anzulegen. Aber spätestens dann, wenn man mit anderen Leuten gemeinsam an einem Projekt arbeitet, wird die Vielzahl der Projektversionen und Namenskonventionen unübersichtlich, und wenn man gar nach einer längeren Unterbrechung wie den Ferien ein begonnenes Projekt fortsetzen möchte, gleicht es einer Art Ratespiel, die letzte funktionierende Fassung herauszufinden, um daran weiterzuarbeiten.

Es gibt noch weitere Schwierigkeiten bei der Verwaltung von Quelltexten bei Oberstufen-Softwareprojekten: Wie lassen sich die Teilergebnisse regelmäßig zusammenfügen, sodass nicht erst gegen Ende des Projektzeitraums Defizite an Schnittstellen aufgedeckt werden? Wie vermeidet man, dass das Team in einzelnen Unterrichtsstunden blockiert ist, weil der Schüler mit wesentlichen Teilen des aktuellen Quelltextes krank zu Hause ist? Wie ermöglicht man, dass begeisterte Schülerinnen und Schüler zu Hause weiterentwickeln können und damit dem Projektfortschritt einen wesentlichen Impuls geben? Wie lässt sich einfach dokumentieren, wer, wann, welche Teile der Software bearbeitet

[1] LMU München, Institut für Informatik, Oettingenstr. 67, 80538 München, brichzin@tcs.ifi.lmu.de
[2] LMU München, Institut für Informatik, Oettingenstr. 67, 80538 München, thomas.rau@ifi.lmu.de

hat, sodass man sowohl jederzeit einen Überblick über den aktuellen Stand hat, als auch rückwirkend eine Übersicht über das Engagement Einzelner für das Projekt hat?

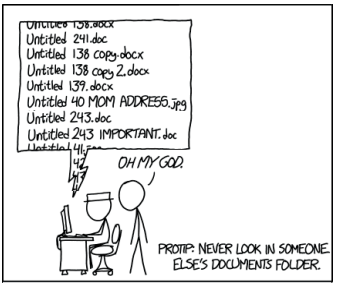


Abb. 1: „Never look in someone else's documents folder", https://xkcd.com/1459/

1.2 Versionskontroll-Systeme als Lösung

In der professionellen Softwareentwicklung in Unternehmen und im Open-Source-Bereich ist der Einsatz von Versionskontroll-Systemen die Antwort auf die oben gestellten Fragen. Zu den Funktionalitäten dieses Werkzeuges gehören:

- Verteilter Zugriff auf Dateien als Voraussetzung für kollaboratives Arbeiten
- Versionierung: Übersicht über verschiedene Versionen inklusiv der Dokumentation wer, wann, was geändert hat
- Datensicherheit, u.a. durch das Rücksetzen der Datei auf eine ältere Version
- Automatisches Zusammenführen (engl. merge) von Quelltexten bzw. Konflikterkennung, falls mehrere Entwickler Veränderungen im selben Bereich durchführen

Bekannte Versionskontroll-Systeme sind u.a. Apache Subversion (seit 2000), Git (seit 2005) und Mercurial (seit 2005). In der Schule werden diese Lösungen bisher selten eingesetzt. Gründe dafür sind möglicherweise, dass diese Werkzeuge oft zu mächtig für den Informatikunterricht erscheinen, dass ihr Einsatz zumindest auf den ersten Blick schwierig wirkt, und dass ihr Mehrwert nicht bekannt ist. Dieser Artikel zeigt, dass eine Vielzahl von Vorteilen der Einarbeitung in eine neue Software überwiegen.

2 Einführung von Versionskontroll-Systemen im Unterricht

Dieses Kapitel zeigt anhand von Beispielen für den Unterricht einen Weg auf, Versionskontroll-Software im Unterricht schrittweise einzuführen. Dabei werden einzelne Funktionalitäten bereits im (nicht projektorientierten) Unterricht eingesetzt, um einerseits auch dort Nutzungsvorteile zu haben und andererseits im Projekt schon auf Kompetenzen im Umgang mit Versionskontrolle zurückgreifen zu können. Damit wurden in der Jahrgangsstufe 11 eines bayerischen Gymnasiums sehr positive Erfahrungen gemacht.

Eine Durchführung bereits in Jahrgangsstufe 10 wird als unproblematisch gesehen. Zu beachten ist jedoch, dass in einem Abschnitt das Vererbungskonzept Voraussetzung ist.

Auf technische Aspekte wird erst in Kapitel 3 eingegangen. Vorausgeschickt wird hier nur, dass die Softwareentwicklung in der Sprache Java mit der Entwicklungsumgebung BlueJ durchgeführt wurde.

2.1 Der erste Kontakt mit einem Repository: Austeilen von Dateien

Im Informatikunterricht stellt die Lehrkraft Schülerinnen und Schülern oft Quelltexte zur Verfügung, z.B. in Form von Programmieraufgaben oder Musterlösungen. Eine Möglichkeit der Verteilung ist die Ablage in einem Ordner des Schulnetzes, auf den aber meist kein Zugriff von zuhause aus möglich ist. So müssen Lehrer und Schüler einen Transfer per USB-Stick oder Internet durchführen. Also macht man als Lehrkraft häufig die Dateien online zugänglich, etwa durch eine Lernplattform wie Moodle. Dem Vorteil des jederzeitigen Zugriffs steht der Nachteil eines höheren Aufwands beim Aktualisieren (etwa bei nachträglichen Verbesserungen) gegenüber, weil die Daten meist zum Hoch- bzw. Herunterladen in einen Datencontainer gepackt bzw. extrahiert werden müssen.

Hier bietet sich das Arbeiten mit Subversion an: Code kann unmittelbar aus BlueJ heraus aktualisiert oder heruntergeladen werden. Für die Lehrkraft ist der Aufwand minimal: Ist zu Hause erst einmal ein komfortabler Client installiert (s. Kapitel **3.4**), markiert die Lehrkraft einfach, welche Verzeichnisse in das Repository hochgeladen oder dort aktualisiert werden sollen und welche nicht (s. Abb. 2). Die Schülerinnen und Schüler benötigen lediglich die Adresse des Subversion-Servers; bei einem öffentlichen Lese-Zugang brauchten sie weder Benutzernamen noch Passwort noch Anmeldung. Diese sind erst dann erforderlich, wenn die Schüler auch eigenen Quelltext zum Projekt beisteuern.

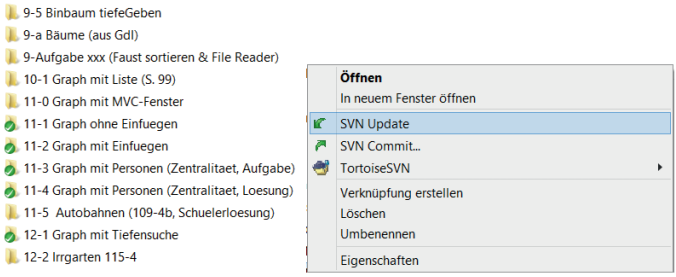


Abb. 2: Screenshot des Dateisystems des Lehrers – Repository Freigabe einzelner Ordner

2.2 Teamarbeit erleben – eine erste Zusammenarbeit mit einem Repository

Im zweiten Schritt bei der Einführung einer Versionskontroll-Software sollen die Schülerinnen und Schüler nicht nur eigenen Quelltext in ein Repository hochladen, sondern

auch erleben, wie man durch Zusammenarbeit im Team sehr schnell eine größere Aufgabe lösen kann – einer der zentralen Vorteile dieses Werkzeuges im Einsatz.

Die Vorgehensweise ist denkbar einfach. Der Lehrer bereitet ein (BlueJ-)Projekt vor, in dem eine Schnittstelle in Form eines Interface oder einer Oberklasse vorgegeben ist. Dazu erstellt jede Schülerin und jeder Schüler eine Implementierung bzw. eine Unterklasse. Das Interface „Tier" verlangt beispielsweise von den implementierenden Klassen eine Methode „String lautGeben()". Zu Beginn ist das mittels „Aktualisieren/Update" aus dem Repository heruntergeladene Projekt noch relativ leer, es enthält nur das Interface und eine Testklasse (s. Abb. 3 links).

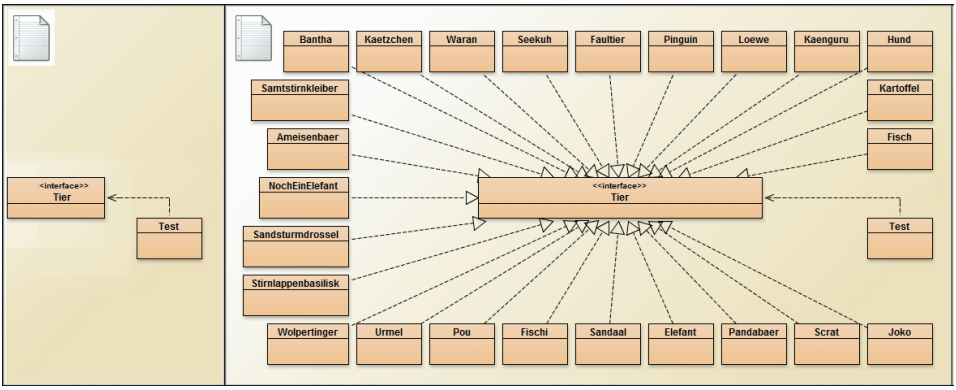


Abb. 3: BlueJ Projekt „Tier" vor und nach der Teamarbeit

Nachdem jeder Schüler eine eigene Klasse erstellt und mittels „Abgeben/Commit" an den Server übertragen hat, ist das Projekt mit einer großen Zahl von Klassen gefüllt – in kurzer Zeit hat man zusammen eine Vielzahl an Ausgestaltungen der Oberklasse geschaffen (s. Abb. 3 rechts). Dabei können die Schülerinnen, wenn sie regelmäßig Updates durchführen, verfolgen, welche Klassen bereits erstellt worden sind. Bei jedem Update können und sollen Schüler kommentieren, was sie am gemeinsamen Codeprojekt geändert bzw. ergänzt haben. Das ist bei größeren Projekten ein wichtiger Bestandteil der Dokumentation des Arbeitsprozesses.

Im Vergleich zum ersten Schritt – das Herunterladen und Aktualisieren von Projekten unter BlueJ/Subversion – kommt wenig hinzu, weil bei der Konzeption darauf geachtet wurde, dass es keine Kollisionen gibt: Da jeder Schüler seine eigene Klasse schreibt, kann es nicht zu Konflikten kommen. Anders als beim reinen Update brauchen die Schüler für einen Commit jeweils eine – möglichst eigene – Benutzerkennung für den Serverzugang (technischen Voraussetzungen s. Kapitel 3.3).

2.3 Konflikte beim Zusammenführen von Quelltextänderungen

Im dritten Schritt sollen die Schüler erfahren, welche Probleme es beim gleichzeitigen

Bearbeiten von Dateien geben kann. Dazu wurde vom Lehrer ein (BlueJ-)Projekt mit einer Klasse „Held“ angelegt, welche eine große Anzahl von Attributen enthielt. Im Paar arbeitend sollten die Schüler sich jeweils zunächst über eine Aktualisierung das Projekt vom Server herunterladen und danach noch nicht vorhandene Getter- und Setter-Methoden zu jedem Attribut anlegen. Wie vorhergesehen kam es dabei zu Konflikten. Schematisch fand dabei folgendes Vorgehen statt:

1. Schülerin A aktualisiert Projekt aus dem Repository.
2. Schüler B aktualisiert Projekt aus dem Repository.
3. Schülerin A legt Methode x an
4. Schülerin A führt Commit durch.
5. Schüler B legt parallel dazu eine gleichnamige Methode x an.
6. Schüler B versucht Commit durchzuführen.

In diesem Fall kann der Commit von Schüler B nicht durchgeführt werden. Er erhält stattdessen eine Warnmeldung; „Ihre Arbeitskopie ist nicht mehr aktuell. Holen Sie eine Aktualisierung aus dem Repository bevor Sie ihre lokalen Änderungen abgeben können.“

Schüler B führt also eine Update durch und erhält dabei, wie sonst auch, eine Nachricht über die zu aktualisierenden Dateien. Die in Konflikt stehende Datei ist mit dem Stichwort „Zusammenführen“ versehen (Abb. 4 links).

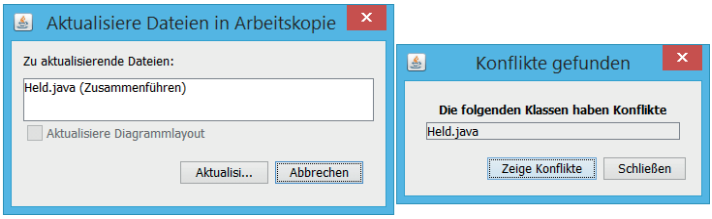


Abb. 4: Konfliktmeldungen

Setzt Schüler B die Aktualisierung fort, gibt es zwei Möglichkeiten: Vielleicht kann der Subversion-Client den Konflikt selbstständig auflösen. Das ist etwa dann der Fall, wenn Schülerin A und Schüler B an verschiedenen Methoden gearbeitet haben. Dann erhält Schüler B eine Code-Version, die seine eigenen Änderungen in den von Schülerin A geänderten und hochgeladenen Code integriert. (Den eigentlichen Commit dieser neuen Fassung muss Schüler B allerdings noch vornehmen.)

Oft kann der Konflikt aber nicht automatisch gelöst werden. In diesem Fall erhält Schüler B eine weitere Warnmeldung (Abb. 4 rechts), worauf er sich die Konflikte anzeigen lassen kann. Dabei wird ihm im Editorfenster von BlueJ der Code mit den in Konflikt stehenden Fassungen gezeigt: Oberhalb einer Reihe von ======= steht der Code des Schülers (bis hin zur Zeile „<<<<<<< .mine“), unterhalb steht der Code aus dem Repository (bis hin zur Zeile „>>>>>>> .r187“, wobei die Zahl die Revisionsnummer, zu der die Änderung gehört (Abb. 5). Jetzt kann Schüler B den Konflikt manuell lösen und die

verbesserte Fassung in das Repository hochladen.

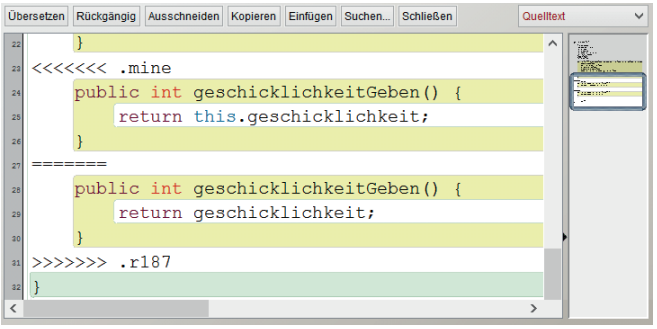


Abb. 5: Anzeige des Konfliktes im Quelltext

Es fällt (Schülerinnen und Schülern) nicht leicht, diese Konflikte zu lösen. Deshalb ist ein wesentliches Lernziel dieses Schritts, ein Bewusstsein dafür zu schaffen, dass bei einem kollaborativen Arbeiten an Quelltext Absprachen (z. B. über ein Taskboard) stattfinden sollten. Arbeiten die einzelnen Teammitglieder an unterschiedlichen Methoden in der Klasse, übernimmt wie oben geschildert die Versionskontroll-Software das Zusammenführen. Dies ist eine erhebliche Erleichterung im Projektablauf (vgl. Ausblick in [Br15]). In diesem Kontext ist auch eine Reflexion über Workflow wichtig (s. Abb. 6).

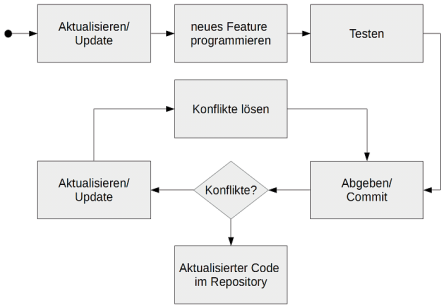


Abb. 6: Workflow für das Arbeiten mit einem Subversion Repository

2.4 Konfliktfreies Zusammenführen durch Teamabsprachen

Im nächsten Schritt sollten die Schülerinnen und Schüler an einem konkreten kleinen Programmierprojekt arbeiten, das Absprachen nötig macht. Ziel war ein schlichtes, funktionierendes Tic-Tac-Toe-Spiel nach dem bereits eingeführten Model-View-Controller-Muster. Dazu einigten sich die Schülerinnen und Schüler erst auf gemeinsame Schnittstellen in Form von abstrakten Model-, View- und Controller-Klassen (s. Abb. 7). Diese Schnittstellen wurden als BlueJ-Projekt allen Schülern über das Repository zur Verfügung gestellt. Zu diesen Klassen programmierten dann Schülerteams konkrete Unter-

klassen, und zwar so, dass es jeweils drei verschiedene konkrete Ausformungen zu allen drei gegebenen Schnittstellen-Klassen gab. Insgesamt arbeiteten also neun Programmierteams am selben BlueJ-Projekt. Wenn sich alle Teams an die Vorgaben der abstrakten Oberklassen hielten, sollten die Ergebnisse beliebig kombiniert werden können.

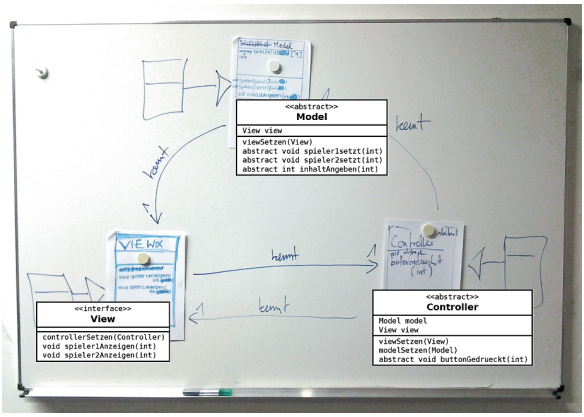


Abb. 7: Ergebnis der Schnittstellendiskussion zu MVC

Da es keine Java-Klasse gab, an der mehrere Gruppen gleichzeitig arbeiteten, konnten keine manuell zu lösenden Konflikte beim Abgeben der Dateien im Repository entstehen. Zwei Regeln stellten sich dabei als hilfreich heraus: a) jede Gruppe arbeitet wirklich nur an der eigenen Klasse; auch zur Lösung von vermeintlichen Fehlern darf nicht in anderen Klassen Code geändert werden, und b) nur fehlerfrei compilierender Code darf ins Repository geladen werden. Es stört die anderen Gruppen beim Arbeiten, wenn Fehlermeldung beim Übersetzungsvorgang erscheinen, und sei es auch in anderen Klassen.

Am Ende entstand ein BlueJ-Projekt mit drei verschiedenen Unterklassen von Model, View und Controller. Die Schülerinnen und Schüler erkannten dabei, dass sich die einzelnen Bestandteile des Spiels – solange sie sich an den vorgegebenen Schnittstellen orientieren – tatsächlich unabhängig von den anderen Teilen einsetzen lassen.

3 Technische Hintergründe und Details

3.1 Grundsätzliches zu Subversion und Git

Für das Verständnis von Subversions-Software sind Hintergrundinformationen zur Funktionsweise hilfreich, insbesondere weil konzeptionelle Unterschiede zwischen den Systemen Subversion [PC09] und git [Git15] [Ch09] existieren.

Bei Subversion sind auf einen zentralen Server in einem Repository (englisch: Depot, Magazin, Lager) alle aktuellen und früheren Dateiversionen gespeichert, ebenso alle

Änderungen und Zusatzinformationen. Die Benutzer arbeiten jeweils mit einem Subversion-Client: In einem Update-Vorgang laden sie diejenigen Dateien des Projekts vom Server herunter, die dort aktueller sind als die lokal gespeicherten. Danach werden die Dateien lokal bearbeitet, und in einem Commit-Vorgang werden die geänderten Dateien auf den Server hochgeladen. Ein anderes Teammitglied kann sich danach – oder auch gleichzeitig – ebenfalls mit „Update“ auf den aktuellen Stand bringen lassen, eigene Änderungen vornehmen, und diese mit „Commit“ an den Server weitergeben. Überschneidet sich das Arbeiten zweier Teammitglieder an derselben Datei, liegt ein Konflikt vor, den Subversion selbstständig zu lösen versucht bzw. anzeigt (vgl. Kapitel 2.3).

Auch bei Git wird mit Repositories gearbeitet. Allerdings speichert hier jeder Client eine vollständige Kopie des gesamten Repository, so dass letztlich alle Git-Repositories gleichberechtigt sind. Die Intention dahinter ist eine hohe Sicherheit gegen unbeabsichtigte bzw. böswillige Verfälschung. Die Hauptfunktion des zentralen Servers ist der Datenaustausch, nicht die Datensicherheit.

3.2 BlueJ und Subversion

Die in Schulen oft eingesetzte Java-Entwicklungsumgebung BlueJ enthält bereits einen funktionierenden Subversion-Client, so dass keine zusätzliche Software installiert werden muss – weder in der Schule, noch bei den Schülern zu Hause. Deshalb wird hier nur das Arbeiten mit Subversion (und BlueJ) vorgestellt.

Die Teamwork-Einstellungen von BlueJ müssen im Menüpunkt „Werkzeuge/Einstellungen“ erst freigeschaltet werden. Danach erscheinen am linken unteren Rand die Knöpfe „Aktualisieren...“, „Abgeben...“ und „Status“; außerdem enthält das Menü „Werkzeuge“ nun ein weiteres Untermenü „Teamarbeit“. Diese Knöpfe entsprechen den Aktionen „Update“ und „Commit“. Bevor man sie verwenden kann, muss sich BlueJ allerdings erst mit einem Repository verbinden. Dazu sind Konfigurationen unter dem Menüpunkt „Werkzeuge/Teamarbeit/Teamarbeitseinstellungen...“ nötig:

- Benutzernamen und Passwort – obwohl dies für ein öffentlich zugängliches Repository beim Update keine Rolle spielt, verlangt BlueJ eine Eingabe im Benutzerfeld
- Servertyp: Subversion – daneben unterstützt BlueJ noch das ältere CVS
- Server bzw. Verzeichnis – die Adresse des zentralen Subversion-Servers und der Pfad ins jeweilige Verzeichnis; ein Beispielprojekt ist unter der Adresse svn.code.sf.net/p/q11/code/trunk/ öffentlich zugänglich
- Protokoll: https

Gibt man diese Daten ein, wird man über den Menüpunkt „Werkzeuge/Teamarbeit/Arbeitskopie erstellen...“ mit dem Server verbunden und kann sich anzeigen lassen, welche Projekte dort zur Verfügung stehen. Das ausgewählte Projekt wird anschließend in ein Verzeichnis des lokalen Rechners übertragen. Wie man als Lehrkraft ein Repository einrichtet und komfortabel verwaltet, wird in Kapitel 3.4 erklärt.

3.3 Anlegen eines Repository

Zentrales Element bei Apache Subversion ist der Server. Subversion ist unter der freien Apache-Lizenz 2.0 veröffentlicht, so dass es sich leicht im Schulnetz installieren lässt (z.B. [svn15]). Da ein Zugriff von außerhalb der Schule gewährleistet werden sollte, wird man aber lieber auf gehostete Subversion-Installationen zurückgreifen.

Sehr komfortabel war das Project Hosting von Google, das unter anderem Subversion anbietet, und mit dem die Projekte in der 11. Jahrgangsstufen durchgeführt wurden. Zum einen hätte ohnehin jeder beteiligte Schüler bereits über einen Google-Account verfügt, zum anderen ist das Passwort, das für den Zugang zum Projekthosting verwendet wird, ein anderes als das zentrale Passwort des Google-Accounts. So war es problemlos möglich, Dummy-Accounts anzulegen, die von der Lehrkraft verwaltet wurden; die Schülerinnen und Schüler erhielten lediglich den Benutzernamen des Dummy-Accounts und das Google-Code-spezifische Passwort. Leider stellt Google sein Projekt-Hosting ab Mitte 2015 ein, so dass für spätere Projekte andere Lösungen gefunden werden müssen.

Bei kostenlosen Hosting-Anbietern wie SourceForge ist das Projekt mit allen seinen Dateien grundsätzlich öffentlich einsehbar; es steht stets unter einer Open-Source-Lizenz. Das hat den Vorteil, dass Schülerinnen den Projektcode auch ohne Zugangsdaten aus dem Repository laden können. Der Nachteil ist, dass Schüler und Lehrkräfte automatisch Nutzungsrechte an ihrem geistigen Eigentum einräumen. Außerdem darf kein Material im Rahmen des Projekts verwendet werden, für das keine Nutzungsrechte eingeräumt wurden, insbesondere urheberrechtlich geschütztes Material aus Schulbüchern.

3.4 Workflow für die Lehrkraft

Für Schülerinnen und Schüler reicht es, wenn sie mit dem in BlueJ integrierten Subversion-Client arbeiten. Als Lehrkraft möchte man aber mehr machen, als nur aktuelle Dateien vom Repository zu laden und geänderte Dateien dort zu aktualisieren. Zu diesem Zweck gibt eine große Zahl von Subversion-Clients mit graphischer Benutzeroberfläche. Für das Arbeiten unter Windows ist TortoiseSVN besonders geeignet. [To15] Nach der Installation erweitert TortoiseSVN das Kontextmenü standardmäßig um drei Einträge (s. Abb. 2). Für ein Windows-Verzeichnis, das mit einem bestehenden Repository-Projekt verbunden ist, können damit Updates und Commits durchgeführt werden.

Das erste Herunterladen eines bereits existierenden Repository mit TortoiseSVN geht in zwei Schritten: Erst legt man einen Ordner an, und wählt nach einem Klick mit der rechten Maustaste aus dem Kontextmenü „SVN Checkout...“ aus. Im nächsten Schritt gibt man die URL des Repository an bestätigt die Auswahl. Danach werden die aktuellsten Fassungen aller Dateien aus dem Repository heruntergeladen. In Zukunft arbeitet man auf diesem Ordner dann nur noch mit den Menüeinträgen zu Update und Commit – bei letzterem muss man natürlich die Zugangsdaten zum Repository angeben.

In die eben heruntergeladene Kopie des Repository kopiert man die BlueJ-Projekte, die

man für einen Informatikkurs braucht. Manche davon möchte man vielleicht nicht gleich mit dem ersten Commit den Schülern und Schülerinnen zur Verfügung stellen. Deshalb kann man mit der rechten Maustaste im TortoiseSVN-Kontextmenü auswählen, welche Ordner mit dem Repository synchronisiert werden sollen und welche nicht. Nur die Inhalte der mit einem grünen Häkchen versehenen Ordner werden im Repository gehalten, die anderen Ordner markiert man mit „Unversion and add to ignore list“ (s. Abb. 2). Das kann auch nachträglich jederzeit geändert werden.

4 Fazit und Ausblick

In diesem Artikel wurde aufgezeigt, dass der Einsatz von Versionskontroll-Systemen im Informatikunterricht sowohl bei der Einführung in die Programmierung, als auch in Softwareprojekten einen Mehrwert hat, weil Workflows im Unterricht unterstützt werden, durch einen verteilten Zugriff und automatisches Zusammenführen von Quelltest kollaboratives Arbeiten im Team deutlich erleichtert wird, sowie die Dokumentation des Arbeitsfortschritts möglich ist.

Bei der technischen Umsetzung fand mit der Auswahl der Teamwerkzeuge von BlueJ eine starke Einschränkung statt. Vorteile dieser Umgebung sind, dass ohne ein zusätzliches Programm der Zugriff auf das Repository direkt aus der Entwicklungsumgebung möglich und die Bedienungsmöglichkeit auf das Notwendigste reduziert ist. Dadurch ist die Einarbeitungszeit und Komplexität für den Bediener minimal. Dem steht der Nachteil gegenüber, dass nicht alle Funktionalitäten eines Repository, z. B. das Arbeiten mit Branches, verfügbar sind und auch die Sicht auf das Repository eingeschränkt sind.

Möchte man hier einen anderen Weg gehen, bietet sich als svn-Client Tortoise bzw. als git-Client SourceTree an. Letzteres visualisiert grafisch auch unterschiedliche Entwicklungen und Zusammenführungen. Dazu haben die Autoren noch keine Unterrichtserfahrung. Leistungsstarke Oberstufenschüler hatten jedoch bei einem Praktikum in einem Softwareunternehmen keine Probleme damit.

5 Literaturverzeichnis

[Br15] Brichzin, P.: Agile Softwareentwicklung, Tagungsband INFOS 2015, Darmstadt.

[Git15] Offizielle git Homepage, http://git-scm.com/ (zuletzt geprüft am 27.04.15).

[Ch09] Chacon, S.: Pro Git, Apress; 2009 http://git-scm.com/book/de/v1 (zuletzt geprüft am 27.04.15).

[PC09] Pilato, M.; Collins-Sussman, B.: Versionskontrolle mit Subversion, O’Reilly, 2009.

[Svn12] Subversion for Windows with Apache server (zuletzt geprüft am 27.04.15).

[To15] Offizielle TortoiseSVN Homepage, http://tortoisesvn.net (zuletzt geprüft am 27.04.15).

Jens Gallenbacher (Hrsg.): INFOS 2015 – Informatik allgemeinbildend begreifen
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 83

Ein Bild vom Wesen der Softwareentwicklung: Erfahrungen aus zwei agilen Projekten

Leonore Dietrich[1], Andreas Gramm[2], Petra Kastl[3] und Ralf Romeike[3]

Abstract: Die beiden Projekte, die hier in getrennten Teilen beschrieben werden, fanden in sehr unterschiedlichen Kontexten statt und zeigen, wie Prozesse unter Verwendung agiler Methoden so an die Bedürfnisse der Beteiligten und die Besonderheiten des Projekts angepasst werden können, dass die intendierten Ziele erreicht werden. Zunächst wird ein „Forschungsprojekt" vorgestellt, das vier Oberstufenschüler mit geringen fachlichen Vorkenntnissen realisierten. Ihre Aufgabe war die Reflexion der erlebten Vorgehensweise. Im zweiten Projekt erlebten Programmieranfänger/innen einer 9. Klasse, wie sie kooperativ selbst Informatiksysteme erschaffen und nach ihren Wünschen gestalten können. Gemeinsam ist beiden Projekten also ihr Fokus auf dem Erlebbar machen eines zeitgemäßen Bilds vom Wesen der Softwareentwicklung.

Keywords: **A**gile Methoden, Softwareentwicklungsprojekt, Unterrichtsprojekt, Informatikprojekt

1 Einleitung

In diesem Bericht sind zwei Projekte zusammengefasst, die in vielerlei Hinsicht verschieden sind: ein kleines Forschungsprojekt, in dem vier Oberstufenschüler eines Hochbegabtenseminars agile Methoden reflektieren und ein Projekt in einer 9. Klasse, in dem Programmieranfänger/innen mit der visuellen Programmentwicklungsumgebung Scratch ein kleines Computerspiel entwickeln. Gemeinsam ist beiden Projekten, dass sie prozessbezogene Aspekte der Schaffung neuer Informatiksysteme in den Vordergrund stellen und so für Schüler/innen näherungsweise erfahrbar machen, wie Informatiker/innen heute ihren Beruf ausüben. Was aber sind wesentliche Aspekte eines solchen Prozesses, was sind Charakteristika moderner Softwareentwicklung? Die Sicht der Wissenschaft darauf hat sich über die Jahre stetig verändert. Zum Ingenieurswesen und seinem Fokus auf Planung und Problemlösungen ist ergänzend ein kreatives, gestalterisches Moment hinzugekommen und es werden Strategien entwickelt, um angemessen auf Anforderungsänderungen zu reagieren [Me14]. Die Idee von dem einen, idealen Entwicklungsprozess wird abgelöst von der Aufforderung: denke selbst und wähle die guten Ideen, die zu dir und deinem Projekt passen [Ru12, Me14]. Eine Aufforderung, die insbesondere an die Entwickler/innen geht, die den Prozess und ihre Rolle im Prozess mit ausgestalten, regelmäßig reflektieren und optimieren sollen. Hinzu kommt eine verstärk-

[1] Universität Heidelberg, Didaktik der Informatik, Im Neuenheimer Feld 326, 69120 Heidelberg, leonore.dietrich@uni-heidelberg.de
[2] Gymnasium Tiergarten, Altonaer Strasse 26, 10555 Berlin, gramm@gymnasium-tiergarten.de
[3] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, Martensstr. 3, 91058 Erlangen, petra.kastl@fau.de, ralf.romeike@fau.de

te Betonung von Eigenverantwortung und Teamorientierung in modernen Entwicklungsprozessen. Die hier beschriebenen Schulprojekte zeigen, dass dieses Wesen von Softwareentwicklung auch in Schulprojekten motivierend und realistisch erlebbar gemacht werden kann. Sie geben Antwort auf die Frage, welches Bild der Informatik, eines Softwareentwicklungsprozesses sowie der Anforderungen von Projektarbeit an Entwickler/innen wir Jugendlichen in einem „agilen Projekt" vermitteln können und wie Schüler/innen ihr Erleben reflektieren.

2 Teil 1: Agile Methoden – nur cool oder mehr? Ein Projekt zur Reflexion des Prozesses durch die Schüler

Das Projekt „Agile Methoden in der Softwareentwicklung" fand im Rahmen des Hector-Seminars statt, einem Baden-Württembergischen Programm zur langfristigen Hochbegabtenförderung, das in der Oberstufe mit einer einjährige Kooperationsphase abschließt, in der Teams, meist in Zusammenarbeit mit einer Hochschule, ein kleines Forschungsprojekt durchführen. Vier Schüler, aus Mannheim und Heidelberg, haben in dieser Phase ein Softwareprojekt mit einer agilen Vorgehensweise [RG12] realisiert. Ziel war, das Vorgehen und die agilen Methoden zu erleben, zu reflektieren und abschließend vorzustellen. Die Projektumsetzung begann im Januar und lief bis Ende des Schuljahres. Im Mittel fand in dieser Zeit ein begleitetes, vierstündiges Treffen pro Monat an der Universität statt, parallel dazu gab es selbständige Treffen. Die fachlichen Vorkenntnisse der Schüler waren überwiegend gering. Sie hatten teils in der Mittelstufe einzelne Kurse zur Robotik (Lego Mindstorms) und zum Einstieg in die Programmierung mit Java (Greenfoot Kara) belegt, aber kaum tiefere, objektorientierte Konzepte verinnerlicht. Ein Schüler hatte grundlegende Java-Kenntnisse, Prozessmodelle aus der Softwareentwicklungstechnik waren nicht bekannt. Allerdings sind die Schüler sehr reflektiert, verfügen über eine extrem schnelle Auffassungsgabe und haben eine außergewöhnlich ausgeprägte Fähigkeit zu analytischem und strukturiertem Denken sowie eine hohe Abstraktionsfähigkeit.

2.1 Der Projektverlauf

Das Spiel, welches mit Greenfoot im Projekt entwickelt wurde, wurde primär als Produkt der Anwendung der ausgewählten Methoden und Strategien thematisiert. Deshalb wurde im ersten Treffen zunächst erläutert, wie Softwareentwicklungsprojekte mit agilen Vorgehensweisen ablaufen. Ausgewählte Videos unterstützten die Ausführungen und vermittelten einen Einblick in das Vorgehen bei professionellen Projekten. Anschließend wurde besprochen, wie es im konkreten Projekt aussehen kann. Dabei sind wir auf folgende Praktiken und Artefakte eingegangen: Stand-Up-Meetings[4], User Stories und

[4] Treffen des Teams, um sich genseitig über die jeweiligen Aktivitäten in arbeitsteiligen Phasen zu informieren.

Tasks[5] sowie ihre Aufwandsabschätzung mit Hilfe des Planning Pokers[6], Sprints[7], Pair Programming mit den Rollen des Drivers und des Navigators, Refactoring[8], Prototypen, die Retrospektive und das Project Board als Planungs- und Informationsbereich.

Im konkreten Kontext hätte sich ein elektronisches Project Board angeboten, da die Treffen an verschiedenen Orten stattfanden und die Schüler von unterschiedlichen Schulen kamen. Allerdings wollte ich den Schülern die Möglichkeit bieten, nach getaner Arbeit für alle sichtbar und haptisch erlebbar einen Zettel umzuhängen. Deshalb bestand unser Project Board aus einem für den Transport gefalteten und zu den Treffen mitgebrachten Plakat (vgl. Abb. 1). Die vierstündigen betreuten Treffen umfassten zwei Sprints von je 45 Minuten. Das Stand-Up-Meeting zu Beginn diente dem Rekapitulieren des vorangegangenen Treffens und ging dann in die Planung des nächsten Sprints über. Während der Planungs- und Entwurfsphase wurde vor allem viel diskutiert. Dokumentiert wurden ein Klassendiagramm, das im Rahmen eines Refactorings erstellt wurde und einige Abläufe in Form von Struktogrammen (vgl. Abb. 1). Für das Projekt und seine Ziele war diese Form der Planung passend, denn die Schüler hatten wenig Vorkenntnisse im Bereich von Spezifikationssprachen und das Produkt war bezogen auf die Fähigkeiten der Schüler zu strukturierendem und abstrahierendem Denken wenig komplex. Außerdem wurde so die "besondere „Schülerklientel" zum Kommunizieren gebracht und es entstand viel Gelegenheit zum Wissenstransfer. Die Reflexionen der Sprints am Ende der Treffen hingegen wurden gut dokumentiert und die Schüler haben Schlussfolgerungen für ihren Entwicklungsprozess schriftlich festgehalten.

Abb. 1: Project Board und Struktogramme

2.2 Beobachtung und Bewertung

Aus Lehrersicht[9]:. Problematisch war die Teamgröße von nur 4 Schülern. Wenn einer der vier Schüler kurzfristig ein Treffen absagte, waren sinnvolle Sprints kaum möglich. Eine Herausforderung stellte das agile Vorgehen bezüglich der Erwartungen an kommu-

[5] Funktionalitäten aus Kundensicht sowie Aufgaben, die zur Umsetzung aus Entwicklersicht notwendig sind.
[6] Praktik zum gemeinsamen und spielerischen Abschätzen des Arbeitsaufwands.
[7] Feste Zeitfenster, in denen der Prototyp inkrementell in Miniprojekten weiterentwickelt wird: auch Iteration.
[8] Das Umstrukturieren des Quelltextes ohne Änderung der Funktionalität.
[9] Hier werden nur die Schülersicht ergänzende Aspekte beschrieben, abweichende gab es nicht.

nikative und kooperative Fähigkeiten dar – zumindest für drei der vier Schüler. Beim ersten Stand-Up-Meeting beispielsweise standen sie da wie „bestellt und nicht abgeholt". Im Verlauf wurde die Kommunikation untereinander beobachtbar besser: nach drei Sprints war beispielsweise zu erkennen, dass die Diskussionen beim Planen und später beim Abschätzen des Aufwands für die Tasks wesentlich differenzierter wurden und die Beteiligung gleichmäßiger verteilt war. Unterstützt wurde die Entwicklung vermutlich auch durch eine wechselnde Zusammenstellung der Pairs beim Programmieren, die für den Wissenstransfer förderlich war. Insgesamt hatte ich den Eindruck, dass sie durch die agile Vorgehensweise und die sogenannten Zeremonien einen für sie praktikablen Fahrplan hatten und eine Struktur, in der sie Fehler erkennen und Probleme lösen konnten. Sie sind planvoll vorgegangen, ohne dass ich als Lehrerin immer den nächsten Schritt vorgeben und extra motivieren musste. Das Planning Poker z. B. traf spontan auf große Zustimmung, trotz anfänglicher Schwierigkeiten aufgrund geringer Programmiererfahrung. Die Schüler reflektierten die erlebten Abweichungen nach jedem Sprint und bezogen die Erfahrungen in die folgenden Abschätzungen ein: „Das denke ich ist etwas / viel mehr / weniger aufwändig als [...], weil [...]". Dabei fielen ihnen auch Aspekte auf, die sie beim Planen der Tasks zu ungenau besprochen hatten, weil dann die Schätzungen auseinander lagen, d. h. die Abschätzung hatte auch eine Kontrollfunktion. Erstaunt hat mich, dass die Schüler von sich aus Fehler, die sie im Test gefunden haben, strukturierten und klassifizierten, um diese dann bei den zukünftigen Prototypen für Tests wieder zu verwenden.

Insgesamt konnte ich viel zuhören und in den Diskussionen von der Planung bis zur Retrospektive verfolgen, wie die Schüler denken, wie sie an Problemstellungen herangehen, sich Unbekanntes erschließen und wie sich ihre Fähigkeiten und Fertigkeiten im Laufe des Projekts entwickeln. Der Aufbau agiler Entwicklung erwies sich für die Schüler als anregend, die Bezeichnungen der Zeremonien sind nah an der Schülersprache und Elemente wie Planning Poker integrieren eine Art Gamification-Ansatz in die Projektsteuerung. Über diese Elemente lassen sich die Schüler gut abholen. Für das nächste Projekt wird die Gruppe allerdings größer ausfallen, um auch bei Krankheit eines Teammitglieds wenigstens zwei Pairs für eine weitere Iteration zusammenstellen zu können. Eine Teamgröße von 6-7 Schülern halte ich auf der Basis der Projekterfahrung für sinnvoll. Interessant wäre auch die Erprobung einer grafischen Programmierumgebung, die den Fokus noch mehr weg von Implementierungsproblemen und hin zum agilen Prozess verschiebt.

Aus Schülersicht: Der erste Durchlauf eines Sprints war für uns eine neue Erfahrung. Die kurzen Phasen erforderten gezieltes Arbeiten am Projekt und damit auch klare Kommunikation und Rollenverteilung. Die Formulierung von User Stories schien uns zunächst schnell von der Hand zu gehen. Bei der Formulierung des Tasks wurde dann aber klar, dass die Stories zum Teil zu klein waren – nur eine Task enthielten – und zum Teil viel zu umfassend. Das anschließende Planning Poker brachte weitere Schwierigkeiten und einige Diskussionen um Zeit- und Implementierungsaufwand. Im ersten Sprint stellte sich dann auch heraus, dass wir uns in einigen Punkten verschätzt hatten, wobei insbesondere die zu großen User Stories dann im vorgegebenen Zeitfenster nicht um-

setzbar waren. Diese wurden in der Sprint Retrospektive angesprochen und für den nächsten Sprint wieder aufgenommen. Die zu Beginn des zweiten Sprints zunächst vorgenommene Neuformulierung brach nun die zu groß geratenen User Stories in kleinere auf. Bei dieser Aufgabe holten wir uns bisweilen auch Feedback bei der Lehrerin. Ein wirklicher Kunde oder eine entsprechende Rolle – diese wurde in unserem Fall dann von der betreuenden Lehrerin übernommen – sind für die Entwicklung sehr hilfreich.

Im Rahmen der weiteren Sprints konnten wir den Umgang mit den Zeremonien verinnerlichen und uns in die schnelle Entwicklung in kurzen Iterationen einarbeiten. Als motivierend wirkte das stets lauffähige Spiel, die Möglichkeit, jederzeit zu testen und, die Auswirkungen der eigenen Programmierung direkt zu sehen. Schwierigkeiten hatten wir beim unbetreuten Arbeiten insofern, als dass wir die engen Zeitvorgaben nicht konsequent einhielten – hier ist wohl ein Projektmanager hilfreich wie ihn die betreuende Lehrerin bei den Präsenzterminen darstellte. Gleiches gilt für den Rollenwechsel in den Pairs innerhalb der Implementierungsphasen. Bei weiteren Präsenzterminen galt deshalb dem Zeitplan und dem Pair-Wechsel besondere Aufmerksamkeit. Wir konnten durch den Pair-Wechsel auch von dem Teammitglied lernen, das die besten Programmierkenntnisse mitbrachte.

Nach drei Iterationen kamen wir an eine Stelle, an der wir einige Programmierprobleme nicht mehr selbständig lösen konnten und fachliche Hilfe benötigten. Der Seminarleiter zeigte uns verschiedene Möglichkeiten zur Lösung des Problems auf. Bei der Umsetzung stellte sich jedoch heraus, dass wir gleichartige Funktionalitäten an vielen verschiedenen Stellen brauchten und somit eine Überarbeitung unserer Gesamtstruktur nötig wurde. Hierfür unterbrachen wir das agile Szenario[10] und entwickelten mit Unterstützung durch die Lehrerin ein Klassendiagramm aus Metaplankarten und Post Its. Das Experimentieren mit den Karten, aus denen wir das Klassendiagramm zusammenbauten, war einfach und schnell und man konnte gut sehen, wie die einzelnen Programmteile zusammenhängen. Indem wir Klassen und Methoden verschoben und verschiedene Strukturen diskutierten, fanden wir eine sinnvolle Struktur. Die Umsetzung des Refactorings übernahm ein Pair, während das andere sich mit einem weiteren identifizierten Problem befasste: der einheitlichen und standardkonformen Benennung von Klassen und Methoden. Als relativ einfach nahmen wir das Zusammenführen des Codes wahr, wobei die zwei Pairs beim Implementieren versuchten, jeweils in verschiedenen Klassen zu arbeiten. So mussten nur die einzelnen Klassen zusammenkopiert werden. Insgesamt war es spannend, den agilen Ansatz kennenzulernen. Bessere Vorkenntnisse in der Programmiersprache und der -umgebung hätten uns allerdings den Einstieg sicher erleichtert.

[10] In einer Iteration wurde kein Produktinkrement erstellt sondern ein Refactoring geplant und umgesetzt.

3 Teil 2: Das Henne-Ei Problem – Wie Programmieranfänger/innen ein Projekt stemmen können

Eine zentrale Frage ist, was wir mit Projekten im Informatikunterricht erreichen wollen. Grundsätzlich finde ich es schwierig, im gegebenen zeitlichen Rahmen Vorgehensweisen bei der Entwicklung größerer Softwaresysteme für Lernende erlebbar zu machen und sie über die Phasen hinweg zu motivieren. Ein typisches Problem ist, dass sie gar keine Analyse machen können, bevor sie nicht ein Projekt durchlaufen haben und wissen, was die eine oder andere Modellierungsentscheidung später bewirkt, was es bedeutet, etwas als Klasse, Interface oder Entität zu modellieren. Ohne ein ordentliches Modell kann ich kein sinnvoll strukturiertes Produkt erstellen und ohne einmal ein Produkt gesehen zu haben, kann ich nicht sinnvoll modellieren – ein Henne-Ei-Problem. Unzufrieden bin ich auch damit, dass Tests aus Zeitgründen zurücktreten müssen und für eine Reflexion trotzdem viel zu wenig Zeit bleibt. Bis aber die Schüler/innen überhaupt über die Voraussetzungen verfügen, um linear verlaufende Projekte angehen zu können, ist oft schon bei vielen ihr Interesse für das Fach der Auffassung gewichen, dass Informatik viel zu schwer ist.

Wichtig ist mir deshalb, ihnen am Anfang zu zeigen, was man mit Informatik machen kann und wie große Informatiksysteme entstehen. Ich möchte ihnen zeigen, dass sie etwas schaffen können, sie für das Fach begeistern und ihnen ein inspirierendes Bild vom Beruf eines Informatikers / einer Informatikerin vermitteln. Gerade im Anfangsunterricht möchte ich erlebbar machen, dass es in diesem Beruf um Gestalten und um Teamwork geht und dass man sich in der Informatik gut organisieren und zusammen tolle Sachen machen kann. Deshalb finde ich ein iteratives Vorgehen sehr verlockend. Man hat kurze Zyklen, sodass die Schüler/innen in zwei Wochen in einem Miniprojekt alles kennenlernen und dabei ein interessantes Zahnrädchen einer größeren Lösung entwickeln. In der nächsten Iteration kommt ein weiteres Zahnrädchen dazu. Es ist mehr als die Aneinanderreihung von Umsetzungen kleiner, unabhängiger Probleme, denn es fügt sich zu etwas Größerem zusammen. Wenn daran sechs Leute in Kooperation arbeiten, entsteht schnell etwas Spannendes. Das Endprodukt muss nicht perfekt sein, es muss nicht jede Idee umgesetzt sein. Trotzdem haben die Schüler/innen einen lauffähigen Prototypen, den sie selbst gemeinsam entwickelt haben und sie wissen, dass und wie es weitergehen könnte. Auch der Test verändert sich dadurch, dass er an einem inkrementell wachsenden, realen Produkt stattfinden kann, nicht auf einer Kommandozeilenebene weit weg vom eigentlichen Produkt.

3.1 Rahmenbedingungen

Das Projekt fand in einer 9. Klasse in einem Wahlpflichtkurs mit einer Doppelstunde pro Woche statt. Die Gruppe bestand aus 20 Schüler/innen, die zum größten Teil nichtdeutscher Herkunftssprache waren, weshalb sich die Kommunikation in kooperativen Arbeitsformen auch immer zur Förderung der Sprachkompetenz im Sinne einer Sprachbil-

dung im Fachunterricht anbietet. Für die Schüler/innen war es ihr erstes Schuljahr mit Informatikunterricht. Das Projekt wurde in der frühen Phase des Kurses mit Scratch durchgeführt. Aufgabe war es, exemplarische Komponenten eines Jump-and-Run-Spiels zu entwickeln. Das Projekt musste umständehalber zwei Mal unterbrochen werden. Die Vorübungen fanden im Herbst statt, die Formulierung der User Stories im Januar und die Realisierung im Frühjahr.

3.2 Einbettung in den Unterricht und Anpassungen an das konkrete Projekt

Vor dem Projekt wurden Kollisionen und Variablen in Scratch behandelt. Die Schüler/innen konnten Zustände und Zustandsveränderungen speichern und wussten, wie eine Kollision als Ereignis erfasst werden kann. Auf dazugehörige Arbeitsblätter mit erklärenden Beispielen konnten sie im Projekt zurückgreifen. Außerdem wurde vorbereitend auf das Projekt geübt, wie einzelne Objekte in Scratch exportiert und importiert werden, wobei deutlich wurde, dass eine Codeintegration relativ einfach abläuft, solange man nicht in gleichen Sprites arbeitet. Diese Übung haben die Schüler/innen als Pair Programming durchgeführt und damit diese Praktik bereits vorbereitend geübt.

Die agile Vorgehensweise sollte eine effektive Produktentwicklung mit Scratch unterstützen und wurde entsprechend angepasst: Da die User Stories sehr klein waren und nur einige wenige Aufgaben umfassten, bedurfte es in Verbindung mit Scratch keiner zusätzlichen Formulierung von Teilaufgaben aus Entwicklersicht. Deshalb wurde nicht zwischen User Stories und Tasks unterschieden. Die User Stories wurden notiert und am Project Board befestigt, ihr Aufwand wurde mit ein bis drei Sternchen[11] und ohne Planning Poker abgeschätzt (vgl. Abb. 2). Eine logische Bearbeitungsreihenfolge ergab sich im Verlauf von selbst, d. h. auch auf eine Priorisierung konnte verzichtet werden, wodurch die Vorarbeiten weiter verkürzt wurden. Stand-Up Meetings fanden zu Beginn jeder Doppelstunde statt und gaben den Lernenden eine Struktur vor, in der sie selbstständig den Stand ihrer Arbeit rekapitulieren und anschließend die Doppelstunde planen konnten, wobei die wesentliche Aufgabe der Planung in der Auswahl geeigneter User Stories bestand. Während der Stunde konnten in der Gruppe auftretende Fragen und Probleme auch in einem Stand-Up-Meeting vor dem Board besprochen werden (vgl. Abb. 2). Da es keinen festgelegten Funktionsumfang gab, der im gegebenen Zeitrahmen umgesetzt werden musste, genügte das Board, um den Projektfortschritt zu verfolgen, auf ein Burn Down Chart zur graphischen Darstellung konnte verzichtet werden. Die konkrete Umsetzung mit Scratch erfolgte im Pair Programming, wobei die Rolle des Navigators aufgrund der wenigen Konzepte und Datenstrukturen, die bekannt waren, eingeschränkt war. Wichtig war, dass der Driver stets seine Ideen bei der Umsetzung ausdrückte und die Schüler/innen so lernten, ihre Programmierung zu beschreiben bzw. kritisch zu hinterfragen.

Es wurden zwei Prototypen entwickelt, von denen nur der zweite zusammen mit einer

[11] Wenig, mittel bzw. viel Aufwand.

Reflexion über das im Projekt Gelernte im Plenum vorgestellt wurde, um eine längere Phase der konzentrierten Projektarbeit zu haben, nachdem das Projekt im Vorfeld zwei Mal unterbrochen worden war. Ihre individuellen Leistungen haben die Schüler/innen abschließend in den Gruppen besprochen und bewertet.

3.3 Die Projektdurchführung

Als Einstieg wurde das Ball Point Game [Gl15] mit allen 20 Schülerinnen und Schülern gemeinsam gespielt (vgl. Abb. 2). Sie hatten einen Heidenspaß und eine deutlich sichtbare Optimierung. Es war eine gelungene Motivation, die zeigte, wie wichtig Absprachen für eine erfolgreiche Kooperation sind und wie man einen iterativen Prozess durch stringentes Planen, Handeln und Reflektieren optimieren kann. In der verbleibenden Zeit dieser Doppelstunde wurde der Kurs in drei Gruppen geteilt, welche erste Ideen für ihr Jump-and-Run-Spiel sammelten, diskutierten und als User Stories formulierten. In einer weiteren Doppelstunde wurden diese User Stories konkretisiert und ergänzt. Die Umsetzung erfolgte in zwei Iterationen von jeweils zwei Doppelstunden an deren Ende jeweils ein Prototyp vorliegen sollte. Teilweise wurden in den Planungen weitere User Stories ergänzt, soweit dies die Umsetzung der Spielidee erforderlich machte. Wenn sie wollten, konnten die Schüler/innen sich ihr Projekt auf einen Stick kopieren und daran zu Hause weiterarbeiten. Die dabei umgesetzten Funktionalitäten wurden in der folgenden Doppelstunde der Gruppe vorgestellt und in das Projekt integriert. Zwei oder drei Mal sind in dieser Phase Probleme aufgetreten, die entweder alle hatten oder auf die sie in Kürze stoßen würden. In diesen Situationen habe ich das Thema zu Beginn der folgenden Doppelstunde kurz im Plenum aufgegriffen und von einzelnen Gruppen erarbeitete Lösungen für das Problem dem gesamten Kurs vorstellen lassen.

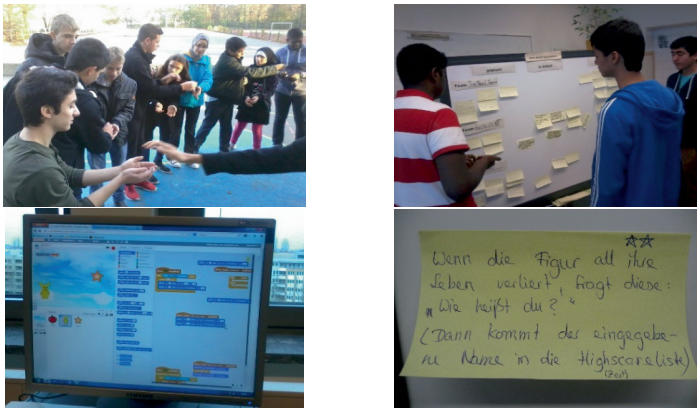


Abb. 2: Ball Point Game, am Project Board, Implementierung, User Story

In der abschließenden Präsentation sollten die Schüler/innen die Umsetzung einzelner Funktionalitäten erläutern können und jede Gruppe sollte eine Zusammenfassung ihrer

individuellen Reflexionen zur der Frage „Was habe ich im Projekt gelernt?“ vorstellen.

Eine Herausforderung stellt die Bewertung von Leistungen dar, die von der Gruppe bzw. individuell in einem Projekt erbracht werden, da zwei unterschiedliche Interessen kollidieren: Zum einen will ich sie befähigen, etwas zu tun und gleichzeitig messe ich, was sie tun. Ein weiterer Punkt ist die Frage, welchen Wert eine individuelle Idee hat und welchen Wert die Fähigkeit der Gruppe, sie erfolgreich umzusetzen. Nachdem die Schüler/innen ihre Projekte selbstorganisiert durchgeführt haben, war es mir wichtig, sie in den Prozess der Bewertung einzubeziehen. Gemäß dem Prinzip einer Poolnote bekam jede Gruppe von mir entsprechend ihrer Gesamtleistung eine Anzahl von Punkten. Die Aufgabe der Gruppe bestand nun darin, sich auf eine Verteilung der Punkte auf die Gruppenmitglieder zu einigen und sich so mit der Frage auseinandersetzen, welche Fähigkeiten und Fertigkeiten jedes einzelnen Mitgliedes die Gruppe im Projekt besonders gut vorangebracht haben und welche eher hemmend waren. Die so erzielte Note umfasste sowohl die Leistung bezüglich des Produkts als auch den Beitrag zur Zusammenarbeit.

3.4 Beobachtungen und Erfahrungen

Die Entscheidung, das Projekt früh im Schuljahr durchzuführen war gut, weil es bei den Lernenden eine positive Einstellung zum Fach erzeugte. Auch das Thema würde ich so wiederwählen, zum einen waren sie von Anfang an interessiert und konnten ihre Erfahrung mit Spielen einbringen, zum anderen entstand im Laufe der Zeit eine Art Wettbewerbssituation. Sie interessierten sich mehr und mehr für die Produkte der anderen. Diese Wettbewerbssituation wirkte zusätzlich anregend, ebenso wie das Peerfeedback. Motivation und Engagement führten dazu, dass schnell gute Prototypen da waren. Die positive Bestärkung der Selbstwirksamkeit durch die iterative Vorgehensweise und die Prototypen sehe ich sehr positiv. Und da sich der Ablauf wiederholt und eine klare Struktur hat, musste ich nicht wie früher moderieren und den nächsten Schritt vorgeben. Die Schüler/innen arbeiteten eigenständiger als in Projekten nach dem Wasserfallmodell und zunehmend selbstbewusster. Mir hat gefehlt, dass sie während des Projekts in fachlicher Hinsicht nicht nur bekannte Fähigkeiten ausbauen und vertiefen, sondern sich im Rahmen des Projekts auch neue Konzepte erarbeiten. Im nächsten Projekt werde ich daher Anregungen bereitstellen, die die Erarbeitung neuer Fachkonzepte initiieren.

Für mich sehr entlastend war, dass von Anfang an klar kommuniziert war: Wir werden nur ein Stück des Spiels implementieren, es wird funktionieren, aber nicht ausgereift sein. Es bestanden weder Anspruch noch Notwendigkeit, in der gegebenen Zeit alle Funktionalitäten umsetzen zu müssen, um ein sinnvolles Produkt zu haben. Darüber, wie sich das bei Oberstufenprojekten umsetzen lässt, muss ich noch nachdenken, hier jedenfalls war ich entspannt und die Schüler/innen waren mit ihren Ergebnissen zufrieden.

Verantwortlich für ihr Projekt haben sich fast alle Schüler/innen gefühlt. Einzelne, die sich herausnehmen, gibt es wahrscheinlich immer, aber hier ist ein solches Verhalten

deutlich aufgefallen und es war für diejenigen unangenehmer als gewöhnlich. In einer Gruppe hat es mit der Teamarbeit nicht gut funktioniert. Auch dort haben sich alle bemüht, etwas beizutragen. Allerdings war in diesem Team eine sehr leistungsstarke Schülerin mit einer schnellen Auffassungsgabe. Sie fand schnell Lösungen und setzte sie um. Leider hat es die Gruppe aber nicht geschafft, dieses Wissen zu transferieren und gemeinsam zu nutzen, obwohl ich dies wiederholt angeregt habe. Diese Schülerin hatte in der Gruppe keinen leichten Stand. Ich vermute, dass sich die anderen Gruppenmitglieder nicht von ihr unterstützt fühlten bzw. es als besondere Herausforderung empfanden, neben ihr auch gute Leistungen zeigen zu können.

Bei der Bewertung haben sich zwei Gruppen sehr schnell geeinigt, die Punkte gleich aufzuteilen. An dieser Stelle war es mir nicht wichtig, einzugreifen oder mir die Überlegungen darlegen zu lassen, da sich meiner Beobachtung nach alle für ein gutes Arbeitsergebnis engagiert hatten. Interessant war wiederum die Gruppe, in der es Spannungen gegeben hatte. In dieser Gruppe gab es entsprechend große Diskussionen: Die leistungsstärkere Schülerin meinte, dass ihr Beitrag mehr Anerkennung finden müsste, die anderen Gruppenmitglieder betonten, dass erst die Tatsache, dass sie der Gruppe eine Überarbeitung zugesagt, aber nicht eingehalten hat, ein besseres Gruppenergebnis verhindert hat. Nach langer Diskussion mit wenig Verständnis für die jeweils andere Position, haben sie sich als Kompromiss auf die gleiche Punktzahl für alle Gruppenmitglieder geeignet. Die Schüler/innen haben damit eine sehr ambivalente Situation bewältigt, die ihnen auch im späteren Leben begegnen wird. Am Ende formulierten sie dann auch in der Reflexion: Absprachen treffen ist wichtig, aber mühsam und teilweise schwierig. Aber ohne Absprachen kann eine fruchtbare Zusammenarbeit nicht funktionieren. Neben dieser Erfahrung war die Präsentation der Arbeitsergebnisse aber vor allem von Stolz auf das Erreichte geprägt. Auch wenn es sich dabei nur um kleine Funktionen handelte: Die Schüler/innen haben erfahren, dass sie gemeinsam selbst Informatiksysteme erschaffen und nach ihren eigenen Wünschen und Interessen gestalten können und waren nun hoch motiviert, weitere Themengebiete der Informatik zu erarbeiten.

Literaturverzeichnis

[Gl15] Glogler B.: Ball Point Game. http://borisgloger.com/scrum/materialien/tools/, 24.04.2015.

[Me14] Meyer, B.: Agile! The good, the hype and the ugly. Springer, 2014.

[RG12] Romeike, R.; Göttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In (Knobelsdorf, M.; Romeike, R. Hrsg.): Proceedings of the 7th Workshop in Primary and Secondary Computing Education. ACM, New York, NY, USA, 2012; S. 48–57.

[Ru12] Rumpe, B.: Agile Modellierung mit UML. Codegenerierung, Testfälle, Refactoring. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 93

Einstieg in das Thema Datenkollision am Beispiel des ALOHA-Protokolls

Patrick Dyrauf[1]

Abstract: Im Rahmen eines Blockseminars zur Didaktik der Informatik wurde ein Unterrichtsentwurf zum Thema Datenkollision erstellt und anschließend in einer 12. Klasse getestet. Hierbei wurde das ALOHA-Protokoll als Beispiel eines nicht kollisionsfreien Protokolls gewählt, da es zum einen sehr einfach aufgebaut ist und zum anderen trotzdem eine bessere Leistung bringt, als man intuitiv erwartet hätte. Des Weiteren existieren viele Erweiterungen, welche als Vertiefung besprochen werden können. Da die theoretischen Berechnungen hierzu bereits die Fähigkeiten eines Leistungskurses Mathematik an seine Grenzen bringen würde, wurde versucht durch eine Simulation in Greenfoot einige Eigenschaften des Protokolls sichtbar zu machen.

1 ALOHA-Protokoll

Das ALOHA-Protokoll wurde 1970 von Norman Abramson entwickelt, um die Nutzer auf den Inseln um Hawaii mit dem Hauptsystem von Honolulu zu verbinden. Obwohl das ursprüngliche ALOHA System für den Kurzstreckenrundfunk entwickelt wurde, lässt sich die grundlegende Idee auf jedes beliebige System von unkoordinierten Sendern, welche sich einen Kommunikationskanal teilen, anwenden. Später wurde es als Grundlage für das Ethernet-Protokoll verwendet.
Steht mehreren Sendern nur ein Kommunikationsmedium zur Verfügung, so ist es notwendig, den Zugriff auf dieses Medium an die Sender zu verteilen, denn durch gleichzeitiges Versenden zweier Nachrichten können diese beschädigt und damit unbrauchbar werden. Solche Verfahren, die es möglich machen über ein Medium mehrere Signale zu übertragen nennt man Multiplexverfahren, wobei in dem hier betrachteten Fall ein Multiplexen über die Zeit stattfindet. Hierzu macht sich das Protokoll als randomisiertes Verfahren explizit den Zufall zu nutze. In der einfachsten Version des Protokolls genügen bereits zwei Regeln, um eine Kommunikation zwischen beliebig vielen Sendern über das gleiche Übertragungsmedium möglich zu machen:

- Jeder darf jederzeit senden.
- Ist ein Sender an einer Kollision beteiligt, so sperrt sich dieser selbstständig für eine gewisse Zufallszeit.

Der theoretisch mögliche Durchsatz, der mit diesen beiden Regeln erreichbar ist, wird mit etwa 18% angegeben unter der Annahme, dass unendliche viele Sender poissonverteilt senden.

[1] Bleichstraße 2, 55232 Alzey, pdyrauf@students.uni-mainz.de

Im Rheinland-Pfälzischen Lehrplan Informatik wird das ALOHA-Protokoll sowohl im Grund- als auch im Leistungsfach als „Hinweis für eine mögliche Umsetzung“ neben CSMA/CD zum Stichwort Datenkollision im Themenblock „Kommunikation in Rechnernetzen erläutern und am Beispiel des Internet verdeutlichen“ genannt. Hierbei wird gefordert, dass die Schüler und Schülerinnen diese Kommunikation in Rechnernetzen in ihren Grundlagen erklären, also analysieren und beschreiben, sowie am Beispiel des Internets erklären können. In dem in dieser Arbeit vorgestellten Unterrichtsbeispiel wurde hierbei der Fokus auf die Analyse und Beschreibung des ALOHA-Protokolls gelegt.[**?**]

2 Simulation

Um den Schülern und Schülerinnen die Möglichkeit zu geben, die Eigenschaften des Protokolls zu entdecken, wurde eine einfache Simulation des Protokolls in Greenfoot erstellt. Im bisherigen Aufbau werden bis zu 64 Sender unterstützt und es lassen sich Werte für den Durchsatz, sowie das Verhältnis von erfolgreich gesendeten zu gesamten Sendeversuchen anzeigen. Es ist natürlich möglich, mit kleinen Änderungen beliebig viele Sender auf einem Spielfeld zu platzieren, jedoch verlangt dies den dafür genutzten Computern ein hohes Maßan Rechenleistung ab, welche eventuell nicht in jedem Unterricht vorhanden ist, weshalb sich hierbei mit niedrigeren Anzahlen begnügt wurde. Des Weiteren wird eine hohe Zahl an Sendern unübersichtlich. Aufgrund dessen wird es umso schwieriger, die Effizienz des Protokolls auch bei hohen Anzahlen zu begründen. Generell muss man sich jedoch darüber im Klaren sein, dass egal wie weit man die Simulation auch treibt, man trotzdem niemals unendlich viele Sender erreichen wird. Zur technischen Realisierung wurde hierbei eine Sende- und Sperrrate eingeführt, welche eine Gleichverteilung auf $[0,1)$ erzeugt. Mit Hilfe dieser ist es dann möglich, über einen induktiven Ansatz zu erkennen, dass um den gleichen Durchsatz bei mehr Sendern zu erhalten, lediglich die Sperrrate angepasst werden muss. Hierbei bezeichnen Sende- und Sperrrate direkte Parameter aus der Simulation. In jedem Schleifendurchgang wird mit Hilfe der eingestellten Sende- und Sperrraten entschieden welche Sender in dieser Runde ein Signal senden und ob bereits gesperrte Sender in der nächsten Runde wieder freigeschaltet werden oder nicht. Dies wurde in diesem Falle insofern implementiert, dass ein Sender mit Senderate Null niemals sendet, während ein Sender mit Senderate Eins in jeder Runde ein Signal abschickt. Unter der Annahme, dass man diese nach Belieben anpassen kann, gelingt dann die Begründung der Lauffähigkeit des Protokolls auch bei beliebig vielen Sendern.

Erinnert man sich an die theoretische Herleitung aus Tanenbaum, so ist dieser Schritt relativ ähnlich zu der Annahme, dass man sein G, welches die erwartete Anzahl an Sendeversuchen beschreibt, auch bei unendlich vielen Sendern so wählen kann, sodass dieses genau $\frac{1}{2}$ beträgt.[**?**]
Des Weiteren lassen sich Situationen erstellen, in denen ein oder mehrere Sender die Regeln des Protokolls verletzen, was dann bezüglich der Punkte Fairness und Angreifbarkeit Diskussionsmaterial bieten könnte. Mit Hilfe der Senderate lässt sich erkennen, dass bei statisch gewählten Sperrzeiten das Protokoll stark von dem jeweiligen Sendeverhalten abhängig ist. Somit ist es hiermit möglich, eine Erweiterung des Protokolls in Richtung dynamischer Sperrzeiten, welche sich an die Netzlast anpassen, zu erarbeiten. Außerdem

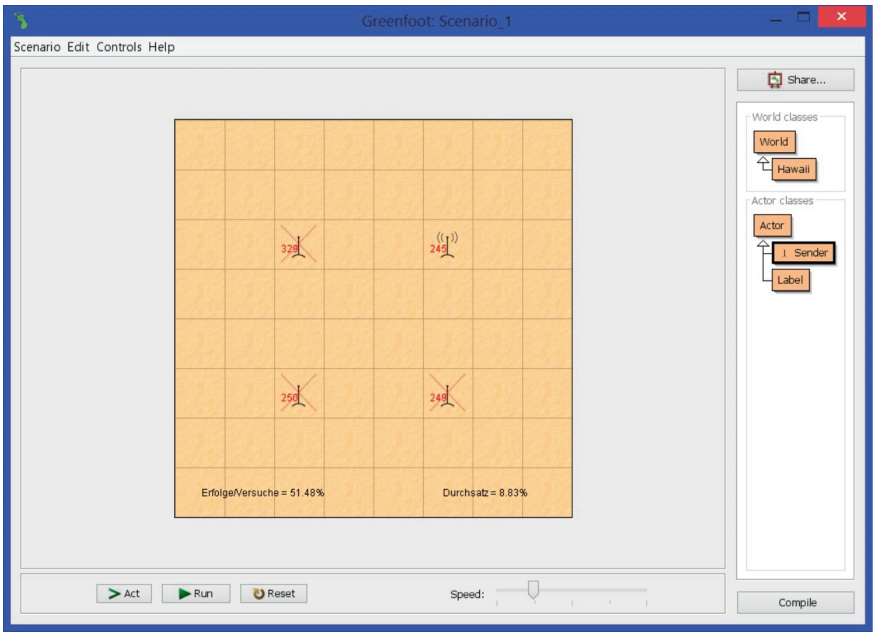


Abb. 1: Simulation in Greenfoot mit vier Sendern von denen drei gesperrt sind

kann mit Hilfe des Zeitbalkens von Greenfoot die Simulation in verschiedenen Geschwindigkeiten durchlaufen werden. Durch langsam laufende Simulationen lässt sich z.B. erkennen, dass während ein Sender sein Signal verbreitet, oftmals ein anderer spontan zu senden beginnt und somit beide Signale verworfen werden. Durch gezielte Betrachtung solcher Situationen wäre eine Erweiterung in Richtung Slotted ALOHA möglich, bei dem jeder Sender nur zu festen Zeitrahmen senden darf.

3 Praxistest

Die Simulation wurde in zwei Unterrichtsstunden einer zwölften Klasse, die aus 19 Schülern bestand, verwendet, um den Einstieg in das Thema Datenkollision zu gestalten. Hierbei wurde zunächst diskutiert, wann Datenkollision auftritt und warum es wichtig ist den Zugriff auf ein Medium zu managen. Anschließend wurde mit Hilfe der Simulation Aufgabe 1 des Arbeitsblattes in Partnerarbeit bearbeitet. Als Hausaufgabe sollte Aufgabe 2 bearbeitet werden. In der folgenden Stunde wurden die Ergebnisse von Aufgabe 1 genutzt, um im Plenum Vor- und Nachteile des Protokolls zu sammeln, sowie zu diskutieren, ob das Protokoll beliebig viele Sender unterstützt oder nicht. Danach wurde Aufgabe 3 des Arbeitsblattes mit Hilfe der Simulation in Partnerarbeit bearbeitet und die Ergebnisse dann im Plenum gesammelt. Zum Abschluss wurde in den verbliebenen acht Minuten ein Fragebogen ausgefüllt.

Die erste Frage des Fragebogens wurde von acht Schülern komplett beantwortet, indem sie die beiden Regeln des Protokolls angaben. Drei Schüler gaben nur eine der beiden Regeln an, zwei gaben keine Antwort und ein Schüler war in der ersten Stunde, in der die Regeln des Protokolls besprochen wurden, nicht anwesend. Die restlichen fünf Schüler gaben als Vereinbarungen Sende- und Sperrrate an.
Bei der zweiten Frage gaben zwölf Schüler genau einen Vorteil und einen Nachteil an, zwei Schüler gaben drei Eigenschaften an, zwei weitere vier Eigenschaften und drei Schüler sogar fünf Eigenschaften, welche allesamt korrekt zugeordnet wurden.
Bei der dritten Frage antworteten 17 Schüler mit ja und elf davon begründeten ihre Antwort korrekt. Ein Schüler antwortete mit nein und einer gab keine Antwort ab.
Bei den abschließenden Multiple Choice Fragen kreuzten lediglich drei Schüler die zweite Frage falsch an, ansonsten wurden alle Fragen der Aufgabe 4 korrekt beantwortet.

4 Fazit

Nach der Durchführung und Auswertung der Fragebögen denke ich, dass das ALOHA-Protokoll durchaus im Unterricht zum Einstieg in das Thema Datenkollision genutzt werden kann. Wie anhand der Auswertung der Fragebögen zu erkennen ist, waren ein Großteil der Schüler in der Lage diesen erfolgreich zu beantworten. Allerdings konnten in den zwei Stunden nur einige Eigenschaften des Protokolls herausgearbeitet und diskutiert werden. Dies liegt unter anderem daran, dass das Verwenden der beschriebenen Simulation sehr viel Zeit in Anspruch nehmen kann, je nachdem für welche Zwecke man diese benutzt.

In Folge dessen schließe ich, dass die Simulation durchaus sinnvoll im Unterricht eingesetzt werden kann, jedoch ist diese an einigen Stellen verbesserungswürdig. Zunächst ist auffallend, dass einige Schüler die Regeln des Protokolls äquivalent zu den Parametern der Simulation sahen. Daraus folgere ich, dass die Regeln des Protokolls noch ausführlicher diskutiert und klarer von den Parametern aus Greenfoot abgegrenzt werden müssen, damit eine Verwechslung ausgeschlossen werden kann. Des Weiteren wäre ein schnelleres und einfacheres Arbeiten mit der Simulation möglich, wenn man Methoden einführen würde, die eine beliebige Anzahl von Sendern direkt auf dem Spielfeld platzieren und ggf. die Spielfeldgröße direkt anpassen könnten. Zwar hat es den Schülern ersichtlich Spaßgemacht diese in vielen kreativen Formen auf dem Spielfeld zu platzieren, jedoch ist dies kein zu erreichendes Unterrichtsziel. Zusätzlich kann man sich überlegen, ob man bei dem eher abstrakten Begriff der Sperrrate bleibt, mit dem die Schüler in diesem Testfall keine Probleme hatten oder lieber zu der intuitiven Variante der Sperrzeit übergeht.

In den Diskussionsrunden gaben einige Schüler an, dass sie das Protokoll nicht verwenden würden, weil der erzielte Durchsatz so gering ist. Hier würde sich wie oben schon erwähnt das Suchen nach Erweiterungen des Protokolls anbieten. Gerade in Greenfoot ließe sich die Version des Slotted ALOHA z.B. so implementieren, dass jeder Sender nur alle k Runden senden darf, wobei k dann die Länge des Slots beschreiben würde. Erweiterungen

dieser Art könnten SuS meines Erachtens auch eigenständig implementieren, bzw. diese zuvor modellieren.

5 Anhang

5.1 Arbeitsblatt

Aufgabe 1

- Öffnet das Szenario_1 in Greenfoot.
- Platziert zunächst die über der Tabelle angegebene Anzahl an Sendern beliebig auf dem Spielfeld und stellt die entsprechenden Sende- und Sperrraten ein.
- Lasst die Simulation einige Zeit laufen und notiert in der Tabelle den angezeigten Wert für das Verhältnis von erfolgreichen Sendungen zu Kollisionen, nachdem dieser sich etwas eingependelt hat, sowie den Wert für den Durchsatz.
- Um nicht nach dem Ändern der Werte alle Sender neu setzen zu müssen, sollte die Resetmethode eines Senders verwendet werden.

Anzahl der Sender: 10		Senderate		
		1	0,1	0,01
Sperrate	1			
	0,1			
	0,01			

Anzahl der Sender: 40		Senderate		
		1	0,1	0,01
Sperrate	1			
	0,1			
	0,01			

Aufgabe 2

Hanna meint: „Wenn man zu viele Sender im Netz hat, dann kann keiner mehr senden, weil sich alle gegenseitig blockieren."

- Nehmt Stellung zu der Aussage und begründet eure Antwort. Verwendet hierzu eure Ergebnisse aus Aufgabe 1 und arbeitet heraus, was Hanna mit „viele Sender" meinen könnte.

Aufgabe 3

Testet, was passiert, wenn sich ein Teilnehmer nicht an die Regeln des Protokolls hält:

- Öffnet dazu das Szenario_2 in Greenfoot.
- Erstellt zunächst 2 Sender.
- In diesem Szenario kann man die Sende- und Sperrrate der Teilnehmer separat ändern. Setzt beide Raten bei einem Sender auf 1 und lasst die Simulation laufen.
- Notiert, was euch auffällt.
- Erstellt nun 9 weitere Teilnehmer von denen zwei die gleichen Parameter wie oben aufweisen. Notiert, bevor ihr die Simulation testet, was ihr erwartet.
- Testet, wie sich die oben beschriebene Konstellation in der Simulation verhält und vergleicht das Ergebnis mit euren Erwartungen.

5.2 Fragebogen

Frage 1: Nenne die Vereinbarungen, die die Sender beim ALOHA-Protokoll treffen müssen, um untereinander zu kommunizieren.

Frage 2: Nenne je einen Vor- und einen Nachteil des ALOHA-Protokolls.

Frage 3: Ist es möglich sich beim ALOHA-Protokoll einen Vorteil gegenüber den anderen Sendern zu verschaffen? Begründe!

Frage 4: Welche der folgenden Aussagen sind wahr oder falsch?

	wahr	falsch
Jeder Sender sperrt sich eigenverantwortlich selbst, wenn er an einer Kollision beteiligt war.		
Beim ALOHA-Protokoll ist sichergestellt, dass jeder irgendwann seine Nachricht vollständig senden kann.		
Beim ALOHA-Protokoll entstehen keine Kollisionen.		
Der Durchsatz beim ALOHA-Protokoll liegt über 30%.		
Das ALOHA-Protokoll funktioniert auch mit beliebig vielen Sendern.		

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 101

Gestrandet auf der Schatzinsel - Schätze heben mit Informatik in der Grundschule

Jens Gallenbacher[1], Karola Gose[2] und Dominik Heun[3]

Abstract:

Das Lernlabor Abenteuer Technik in Darmstadt hatte den Mathematik- und Informatikworkshop "Gestrandet auf der Schatzinsel" im Programm, der sich an Grundschulklassen richtete. Die Schülerinnen und Schüler sollten hier Beispiele für Informatik kennenlernen und erste Erfahrung mit Codierung sammeln.

Die Schülerinnen und Schüler stranden auf einer Insel und finden eine Flasche, in der sie die Schatzkarte von Kapitän Graubart finden. Mit einigen Tricks können sie die geheimnisvollen Zeichen auf der Schatzkarte deuten, die sie zur Schatzkiste führen. Mit einigem Nachdenken können sie sogar den Zahlencode der Schlösser knacken und den Schatz heben. Am Ende haben sie nicht nur viel Spaß gehabt, sondern auch einiges über Informatik gelernt.

Im Fokus des Workshops steht die erste Arbeit mit Codierungen innerhalb der Informatik. Die Teilnehmerinnen und Teilnehmer lernen Repräsentationen von Bildern, das Binärsystem und binäre Suche kennen. Dabei wird großen Wert auf eine anschauliche, enaktive, motivierende Aufbereitung der Materialien gelegt.

Die Materialien können mit leichter Modifizierung auch im Anfangsunterricht eingesetzt werden.

1 Einleitung

Informatikunterricht kann einen großen Beitrag zur Allgemeinbildung auch schon in der Grundschule leisten. Dafür ist eine Ausrichtung an fundamentalen Ideen der Informatik und allgemeinbildenden Kompetenzen notwendig. Die Gesellschaft für Informatik (GI) formuliert in ihren Bildungsstandards für die Sekundarstufe I jeweils fünf Prozess- und Inhaltsbereiche, die zu einem allgemeinbildenden Informatikunterricht anleiten sollen.

Das Lernlabor *Abenteuer Technik*, das in Darmstadt von Mai 2011 bis Dezember 2014 über 5000 Schülerinnen und Schüler in Tagesworkshops betreuen durfte, baute auf diesen Kompetenzen auf und ging noch einen Schritt weiter: Die Förderung von Kompetenzen, die in allen technischen Disziplinen Anwendung finden (Modellieren und Problemlösen), sollte forciert werden. Mit diesen beiden in der Informatik zentralen Kompetenzen sollte der Einstieg in Technik ermöglicht werden. Eine große Rolle spilter der Ansatz Anchored Instruction, der in einem Beitrag der INFOS 2013 detailliert vorgestellt wurde([GH13]).

[1] Technische Universität Darmstadt, Didaktik der Informatik, Hochschulstraße 10, 64289 Darmstadt, jg@di.tu-darmstadt.de

[2] Altes Kurfürstliches Gymnasium Bensheim, Wilhelmstraße 62-64, 64625 Bensheim, kgoseweb.de

[3] Technische Universität Darmstadt, Didaktik der Informatik, Hochschulstraße 10, 64289 Darmstadt, dh@di.tu-darmstadt.de

Low Tech stellt einen weiteren Grundpfeiler der Arbeit des Lernlabors dar, Repräsentation dafür, dass technische Disziplinen zwar Technologien verwenden, diese aber lediglich Werkzeuge sind. Die eigentliche Problemlösung findet meist vor der Arbeit mit dem Computer statt und lässt sich weitgehend durch Nachdenken und Kreativität gestalten. Die Verlegung des Fokus weg vom Computer hin zur Problemlösung, entgegen der teilweise widersprechenden öffentlichen Meinung, macht die Problemlösung zum neuen Schwerpunkt. Dieses Vorgehen ist aus der Informatikausstellung *Abenteuer Informatik* bekannt und erfolgreich erprobt.

2 Vorüberlegungen

In der Fachdidaktik der Informatik gibt es schon einige Ansätze, die sich mit Informatikunterricht in der Grundschule beschäftigen, etwa in Bezug auf die Arbeit mit Robotern (bspw. [Bo13]) oder der Ausstattung von Kindern mit Laptops in der Initiave *One Laptop per Child*. Alle diese Ansätze verfolgen die ebenfalls wichtige Heranführung von Lernenden an den Computer als Werkzeug.

Im Gegensatz dazu, sollen bei uns die Schülerinnen und Schüler erste Erfahrungen mit Modellen und Problemen der Informatik sammeln. Um dies im Kontext von *Abenteuer Technik* zu ermoglichen, wird auf Technologie verzichtet und auf *Pen and Paper*-Beispiele gesetzt. Dies vermittelt Informatikdenken ohne dabei auf den Computer zurückzugreifen.

Die bisherigen Workshops des Lernlabors richteten sich an Lernende ab der 6. Klasse, so dass mit Lernenden der 3. und 4. Klasse noch keine Erfahrungen gesammelt werden konnten. In Workshops mit Grundschulkindern zu der Ausstellung *Abenteuer Informatik* sowie Sonderaktionen wie dem *Maus-Türöffnertag* konnten jedoch Eindrücke zum Verhalten und Arbeitsweisen von Kindern dieser Jahrgangsstufe gesammelt werden.

Der Ansatz *Anchored Instruction* ist gerade für jüngere Jahrgangsstufen geeignet, da sich diese stark auf die Rahmenhandlung einlassen und so neben den positven Aspekten des Transfers der Problemstellungen und des anwendbaren Wissens (vgl. [GH13]) die besondere Motivierung der Lernenden hinzukommt.

Für die inhaltliche Ausrichtung des Workshops wurden zunächst Themenbereiche aus der Informatik identifiziert, die für Schülerinnen und Schüler der dritten Klasse geeignet sind sowie den Anforderungen des Lernlabors genügen. Dabei wurden Aktivitäten von *CS unplugged* (vgl. [BWF15]) und *Abenteuer Informatik* auf Tauglichkeit im Einsatz des Workshops geprüft. Zunächst wurden Inhalte aus dem Bereich “Data Represenation” gewählt, die in den Workshop integriert werden sollten. Bekannte Aufbereitungen der Ausstellungen *Abenteuer Informatik* wurden nicht gewählt, da diese im Lernlabor auch außerhalb des Workshops zur Verfügung standen.

Eine weitere Vorgabe für den Workshop war die Verwendung von Themenmodulen, so dass bei einer Weiterentwicklung des Workshops auch andere Themengebiete eingesetzt werden können, ohne den Grundaufbau des Workhops ändern zu müssen. Die Länge eines Moduls wurde auf 30 Minuten beschränkt, da die bisherigen Erfahrungen gezeigt, dass die

kontinuierliche problemorientierte Arbeit mit Lernenden der Grundschule nur mit Phasenwechseln in diesem Zeitintervall sinnvoll ist.

Zusätzlich soll durch den Workshop die Kooperation im Team aufgebaut werden, so dass die Teilnehmenden nur durch Teamarbeit das gemeinsame Ziel des Workshops erreichen können.

3 Aufbau des Workshops

Ausgehend von der vorherigen Entscheidung für das Themenfeld "Data Representation" wurden drei Module entwickelt, die jeweils auf einer Aktivität von *CS unplugged* bzw. *Abenteuer Informatik* basieren. Diese drei Aktivitäten sind:

- Bildübertragung (Colour by Numbers - Image Representation (CS unplugged))
- Binäre Suche (Schnelle Suche (Abenteuer Informatik))
- Binärzahlen (Count the Dots (CS unplugged))

Im Folgenden wird zunächst ein Überlick über die Rahmenhandlung gegeben und danach auf die einzelnen Module eingegangen.

3.1 Rahmenhandlung

Durch die Auswahl des Themenbereichs Codierung lag der Entschluss nahe, die Rahmenhandlung auf eine Schatzsuche festzulegen, da eine Schatzkarte im weitesten Sinne auch der Codierung von Informationen (hier Lageinformationen) entspricht. Außerdem erfreuen sich Schatzsuchen im Grundschulalter großer Beliebtheit. Dabei muss aber beachtet werden, dass der kooperative Gedanke nicht unberücksichtigt bleibt, da bei einer Schatzsuche sehr schnell Konkurrenzdenken aufkommen kann.

Zu Beginn des Workshops sehen die Teilnehmenden einen kurzen Film, der die Rahmenhandlung aufspannt. Darin geraten sie mit einem Kapitän und seinem Schiff in ein Unwetter, bei dem das Schiff sinkt. Die Teilnehmenden und der Kapitän können sich auf eine nahegelegene Insel retten und sind dort auf sich alleine gestellt. Die Betreuenden des Workshops nehmen ebenfalls Rollen innerhalb der Geschichte ein und begleiten die Teilnehmenden durch den Workshop.

Nach einiger Zeit findet eine Betreuungsperson eine Flaschenpost, in der sich eine Schatzkarte und eine Nachricht befindet. Die Nachricht wird von den Lernenden vorgelesen. Aus der Nachricht geht hervor, dass ein Schiffsbrüchiger auf dieser Insel bereits nach dem Schatz von Kapitän Graubart gesucht hat. Aus den beiliegenden Schatzkarten (beispielhaft in Abbildung 1) ist er jedoch nicht schlau geworden. Die beiden Schatzkarten unterscheiden sich lediglich in der Anordnung der verschiedenen Zeichen. Zusätzlich zu der Schatzkarte liegt ein Lageplan mit Koordinaten des Lernlabors bei. Damit die Lernenden sich als Schatzsuchtruppe fühlen, erhalten sie nun jeweils einen Ansteckbutton ihrem Namen.

Abb. 1: Schatzkarte

3.2 Bildübertragung

Nachdem die Teilnehmenden sich selbständig mit der Schatzkarte beschäftigt haben und dabei (meist) zu keinem Ergebnis gekommen sind, werden sie in zwei Gruppen eingeteilt, die sich jeweils einer Schatzkarte widmen. Jeweils ein Betreuender setzt sich mit einer Gruppe zusammen und hilft bei der Entschlüsselung der Karte.

Abb. 2: Bildübertragungskarten

Im ersten Spiel erhalten die Lernenden verschiedene Karten, die in Abbildung 2 abgebildet sind. Zusätzlich erhalten ein vorgegebenes Raster und Stifte. Von der Betreuungsperson werden nun nach und nach weitere Karten ausgeteilt, die ein gegebenes Bild repräsentieren. Beginnen die Lernenden, die Kästchen im Raster in der jeweiligen Farbe der Karte zu färben, erhalten sie am Ende ein Bild.

Nachdem dieser “Algorithmus” einmal vorgegeben wurde, wird eine Schülerin oder ein Schüler ausgewählt, um sich selbst ein Bild auszudenken und den anderen über die Karten zu vermitteln. Nach und nach werden die Lernenden an das Verfahren gewöhnt und lernen, welche Informationen für eine reibungsfreie Kommunikation über die Karten notwendig

sind. So erkennen sie bspw. in fehlerhaften Versuchen, dass die Information des Beginns einer neuen Zeile im Raster für den Empfänger eine wichtige Information ist, da ansonsten das Bild nicht fehlerfrei rekonstruiert werden kann.

Können die Teilnehmenden ein Bild fehlerfrei übertragen, lenkt die Betreuungsperson den Fokus erneut auf die Schatzkarte und die erste Aufgabe. Die Teilnehmenden interpretieren die Zeichen entsprechend der Karten sowie ein Leerzeichen als Anfang einer neuen Zeile und können so die erste Aufgabe dekodieren. Die erste Gruppe erhält bspw. einen Buchstaben wie in Abbildung 3 dargestellt.

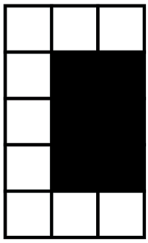


Abb. 3: Lösung der Bildübertragung

Doch mit einem Buchstaben alleine können sie nicht die Position des Schatzes bestimmen. Sie müssen mit der zweiten Gruppe zusammenarbeiten, um die exakten Koordinaten des Schatzes zu erhalten. Zusammen können Sie die Schatzkiste heben. Doch daran befindet sich eine schwere Eisenkette mit zwei dreistelligen Zahlenschlössern, deren Code sie nicht kennen. Somit wenden sie sich erneut der Schatzkarte zu.

Abb. 4: Schatzkiste

3.3 Binäre Suche

An einer Wand befindet sich ein großer Zahlenstrahl mit den Zahlen von 1 bis 64. Die Betreuungsperson spielt mit den Lernenden ein Zahlenratespiel, bei der sie sich eine Zahl ausdenkt und die Lernenden sie erraten müssen. Ist ein Rateversuch falsch, antwortet sie mit “höher” oder “tiefer”, je nachdem, was auf die ausgedachte Zahl zutrifft.

So können die Schülerinnen und Schüler mit dem Zahlenstrahl an der Wand und zwei Klebezetteln mit aufgemalten Pfeilen die möglichen Zahlen einschränken. Anhand ausgewählter Beispiele führt die Betreuungsperson den Lernenden vor Augen, wie sie am geschicktesten raten können. Dies führt zur binären Suche, wobei für Lernende im Grundschulalter das Finden der Mitte zwischen zwei Zahlen nicht trivial ist. Doch auch hierfür eignet sich der Zahlenstrahl als Veranschaulichung. Zwei Lernende stellen sich jeweils an den beiden Zahlen auf, zu denen die Mitte gefunden werden soll. Gleichmäßig gehen sie Zahl für Zahl aufeinander zu und treffen sich schließlich bei der mittleren Zahl.

Übertragen die Lernenden diese Strategie auf die Schatzkarte und interpretieren die Pfeile als Angabe "höher" oder "tiefer", können sie eine Ziffer des Schlosses herausfinden. Doch diese Ziffer reicht noch nicht aus, um das Schloss zu öffnen.

3.4 Binärzahlen

Bei diesem Modul wird anhand der Aktivität "Count the Dots" aus *CS unplugged* bzw. "Binäruhr" aus *Abenteuer Informatik* das Umwandeln von Binärzahlen in Dezimalzahlen geübt.

Vier der Lernenden stellen sich nebeneinander auf und erhalten (von rechts nach links) Karten, auf denen sich auf einer Seite ein, zwei, vier oder acht Punkte befinden. Auf der Rückseite sind keine Punkte abgebildet. Die Betreuungsperson geht mit den Teilnehmenden die Leitfragen der zugrundeliegenden Exponate durch und führt so das Zählen und Umwandeln von Binärzahlen ein.

Erneut wird das Gelernte auf die Schatzkarte übertragen und es können zwei weitere Ziffern entschlüsselt werden. Damit können die Teilnehmenden alle Stellen des Schlosses und damit die Schatzkiste öffnen.

3.5 Abschluss

Haben beide Gruppen die jeweiligen drei Ziffern herausgefunden, können sie die Schlösser von der Kiste entfernen und an den Inhalt gelangen. In der Kiste befinden sich sogenannte *Mitmachhefte* (beispielhaft in Abbildung 5), in denen sich die Aktivitäten des Tages in kleiner Form befinden. Diese können sie nach Hause mitnehmen, dort den Eltern, Geschwistern die neuen Erkenntnisse vorführen. Für das Aufgreifen im Unterricht eignen sich die *Mitmachhefte* ebenfalls.Zusätzlich erfüllt das *Mitmachheft* eine Sicherungsfunktion, indem die Aktivitäten noch einmal aufgelistet und erklärt werden.

4 Erfahrungen

Der Workshop wurde im Oktober 2014 in das Programm des Lernlabors aufgenommen. Bis Dezember 2014 haben 25 Schulklassen das Angebot eines Klassenworkshops wahrge-

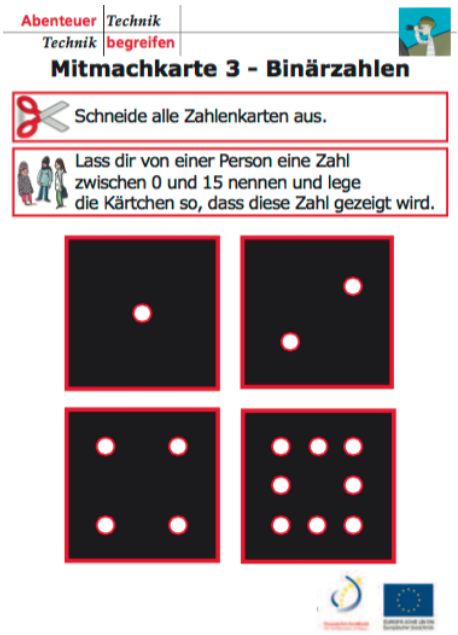


Abb. 5: Auszug aus Mitmachheft

nommen. Das unmittelbare Feedback der Schülerinnen und Schüler sowie der Lehrpersonen fiel sehr positiv aus. Gelobt wurde vor allem die sehr motivierende Rahmenhandlung, die die Kinder sehr schnell in ihren Bann gezogen hat.

Die verwendeten Aktionen aus dem Bereich *Codierung* haben sich als sehr gut für den Einstieg herausgestellt und können von den Lernenden im Grundschulalter bearbeitet werden.

Die Aufgaben werden im Workshop durch die Betreuungspersonen frontal eingeführt und im Lehrer-Schüler-Gespräch erarbeitet. Danach erfolgt eine freie Berarbeitung durch die Lernenden, die jedoch von der Betreuungsperson beobachtet und im Zweifel angeleitet wird. Dieses Vorgehen, als Mittelweg zwischen freier Bearbeitung und strikter Anleitung, stellt für diese Altersgruppe einen guten Kompromiss dar. Mit einer zu großen Freiheit können einige Lernenden in diesem Alter noch nicht umgehen, während andere durch die Anleitung eigener Erkenntnisse beraubt werden. Hier wäre eine Weiterentwicklung im Bereich der Binnendifferenzierung sinnvoll.

Die *Mitmachhefte* als Schatz wurden von den Teilnehmenden ebenfalls sehr positiv angenommen. Nach einigen Durchführungen wurde das Mitmachheft um eine Urkunde ergänzt, in die der Button der Teilnehmenden eingefügt werden kann. Dies wertete den Umgang mit den *Mitmachheften* enorm auf.

Zum Abschluss des Workshops wurde beim Austeilen der *Mitmachhefte* auch der Bezug zur Informatik hergestellt, etwa durch die Information, dass schwarz und weiß bzw. “an”

Abb. 6: Urkunde

und “ausëbenfalls im Computer eingesetzt werden können, um Zahlen darzustellen. Momentan handelt es sich dabei jedoch m um einen durch Bilder unterstützte Präsentation durch den Betreuenden. Auch dies sollte besser in die einzelnen Module direkt integriert werden.

5 Einsatz im Schulunterricht

Da die Materialien durch den Einsatz in *CS unplugged* und *Abenteuer Informatik* auch losgelöst von einem Workshop durchgeführt wurden, kann der Workshop in Teilen auch in den Grundschulunterricht integriert werden. Die Lernenden sollte schon einmal mit offenen Aufgaben gearbeitet haben, damit sie mit der Aufgabenform nicht überfordert sind. Um gut in Kleingruppen arbeiten zu können, sind zwei Lehrpersonen vorteilhaft.

Beim Einsatz in der Schule muss der Ansatz der *Anchored Instructions* in einer anderen Weise umgesetzt werden. Die Methode des Geschichte-Erzählens ist in der Grundschule nicht ungewöhnlich und lässt sich mit der Geschichte des Workshops ebenfalls umsetzen. Bei entsprechender Ausstattung der Schule, kann der Einstiegsfilm über einen Projektor gezeigt werden.

Der Lageplan mit Koordinaten ist auf den Einsatz im Lernlabor zugeschnitten, so dass er nicht im Schulunterricht verwendet werden kann. Hier ist eine Anpassung an die jeweilge Umgebung erforderlich. Die Schatzkiste muss mit mindestens zwei Schlössern verschließbar sein und wird mit mehr als zwei Gruppen gearbeitet, sind entsprechend mehr Schlösser notwendig. Werden die Schlösser auf die Ergebnisse der Aufgaben eingestellt, können die Aufgabenstellungen direkt übernommen werden.

Für interessierte Lehrerinnen und Lehrer ist bei den Autoren eine Lehrerhandreichung erhältlich, die die Hintergründe der Aufgaben sowie die Schatzkarten enthält.

6 Ausblick

Da die Finanzierung des Lernlabors zum 31. Dezember 2015 eingestellt wurde, kann der Workshop aktuell nicht weiterentwickelt werden, obwohl eine große Nachfrage seitens der Grundschulen besteht. Die dargelegte Durchführung zeigt jedoch, dass auch ein Informatikunterricht abseits der Einbeziehung von Computern in der Grundschule möglich ist.

Die spielerische Behandlung von Codierungen hat den Lernenden einen ersten Eindruck von Informatik vermittelt und gezeigt, dass eine Beschäftigung mit Informatik Spaß machen kann.

Literaturverzeichnis

[Bo13] Borowski, Christian: Kinder auf dem Weg in die Informatik: Roboter in der Grundschule. In (Norbert Breier, Peer Stechert, Thomas Wilke, Hrsg.): INFOS 2013, 15. GI-Fachtagung İnformatik und Schule". Kiel Computer Science Series 2013-03. Department of Computer Science, CAU Kiel, S. 21–28, 2013.

[BWF15] Bell, Tim; Witten, Ian; Fellows, Michael: , CS unplugged - An enrichment and extension programme for primary-aged students, April 2015.

[GH13] Gallenbacher, Jens; Heun, Dominik: Ein moderner Ansatz für Anchored Instruction im Informatikunterricht. In (Breier, Norbert; Stechert, Peer; Wilke, Thomas, Hrsg.): Informatik erweitert Horizonte, INFOS 2013, 15. GI-Fachtagung Informatik und Schule, 26.-28. September 2013, Kiel, Germany. Jgg. P-219 in LNI. GI, S. 87–96, 2013.

Jubel, Trubel, Informatik - Ein Schülerworkshop für den Klassenraum

Jens Gallenbacher, Dominik Heun und Wiebke Kothe[1]

Abstract:

Im Lernlabor *Abenteuer Technik* konnten bis Dezember 2014 Schulklassen den Informatikworkshop "Jubel, Trubel, Informatik" besuchen. Ziel dieses Workshops sind die Vermittlung der allgemeinbildenden Kompetenzen Problemlösen und Modellieren.

Die Schülerinnen und Schüler planen einen Jahrmarkt, der in Darmstadt stattfinden wird. Dabei müssen sie zahlreiche Aufgaben meistern, zum Beispiel die Verkabelung einer Geisterbahn, die Planung von Routen für Reinigungskräfte oder die Zusammenstellung von konfliktfreien Liefergruppen der Aussteller. Alle diese Aufgaben haben eines gemeinsam: Sie können mit Methoden der Informatik gelöst werden.

Im Vordergrund des Workshops steht die konzeptionelle Arbeit mit informatischen Problemen, die ohne den Computer gelöst werden sollen. Frei nach dem Motto der INFOS "Informatik allgemeinbildend begreifen" unterstützen die Materialien eine enaktive Auseinandersetzung mit den Problemen.

Nach einer Überarbeitung der Materialien kann der Workshop nun auch in den Unterricht integriert werden. Dieser Beitrag stellt die zugrundeliegenden Konzepte sowie die Unterrichtsmaterialien vor.

1 Einführung

Die Vermittlung allgemeinbildender Kompetenzen ist eines der höchsten Ziele des Informatikunterrichts. Neben dem Aufbau sinnstiftender Kommunikation und Kooperation kann die Informatik vor allem Beiträge zum Problemlösen und Modellieren leisten. Dieser Aspekt findet sich ebenfalls in den Bildungsstandards der Gesellschaft für Informatik [2] wieder, genauer im Prozessbereich *Modellieren und Implementieren*.

Im Zeitraum von Mitte 2010 bis Dezember 2014 entstanden diesbezüglich im Lernlabor *Abenteuer Technik* in Darmstadt eine Vielzahl von Workshops, die im Klassenverband besucht werden konnten. Über 5000 Schülerinnen und Schüler konnten bis zu diesem Zeitpunkt im Lernlabor in Tagesworkshops betreut werden. Workshops, die für dieses Lernlabor konzipiert wurden, zeichneten sich durch verschiedenen Kriterien aus:

- Kompetenzorientierung
- Anchored Instruction

[1] Technische Universität Darmstadt, Didaktik der Informatik, Hochschulstraße 10, 64289 Darmstadt, jg@di.tu-darmstadt.de, dh@di.tu-darmstadt.de, wk@di.tu-darmstadt.de

[2] [**?**]

- Genetischer Aufbau
- Hands-On
- Lowtech

Eine große Besonderheit liegt in der Verwendung des Konzepts Anchored Instruction, das für das Lernlabor entsprechend weiterentwickelt wurde[3]. Die Lernenden erleben den ganzen Workshop als Rollenspiel, in dem die auftretenden Probleme aus dem Rollenspiel heraus entstehen. Die Lernenden können so die Herausforderungen wirklich erleben und die Relevanz der Lösung erkennen.

In [**?**] wurde ein Modul aus dem Lernlabor beschrieben. In diesem Bericht soll ein weiterer Workshop dargestellt werden: Jubel, Trubel, Informatik. Dieser Workshop ist nun für die Schule überarbeitet worden. Materialien wurden für den Unterricht geändert oder erstellt. Diese können bei den Autoren angefragt werden.

Nachfolgend wird der Workshop zunächst beschrieben und die Materialien dargestellt. Danach erfolgt eine didaktisch-methodische Betrachtung sowie eine Beschreibung der bisherigen Erfahrungen.

2 Überblick

In Darmstadt fand im letzten Jahr ein bekannter Jahrmarkt statt, für dessen Planung drei Personen angestellt wurden. Ein Magier, ein Schatzmeister und ein Jurist, alle drei Spezialisten auf ihrem Fachgebiet, schafften es jedoch nicht den Jahrmarkt erfolgreich durchzuführen. Verschwenderische Kabelverlegung in der Geisterbahn, Vertragsstrafen oder fernbleibende Aussteller, führten zu empfindlichen finanziellen Verlusten.

Alle drei dürfen den nächsten Jahrmarkt erneut planen, erhalten dabei aber Unterstützung durch die Lernenden. Auf einer Pressekonferenz sollen sie ihre Pläne für die verschiedenen Probleme darlegen.

Der Workshop besteht aus drei unterschiedlichen Aufgabenbereichen, die von verschiedenen Gruppen bearbeitet werden können: Eulerkreise/-wege, Konfliktgraphen und minimale Spannbäume. Im Rahmen des Unterrichts bietet sich ebenfalls die Möglichkeit, dass die verschiedenen Bereiche von allen Lernenden nacheinander bearbeitet werden.

Zielgruppe des Workshops sind Schülerinnen und Schüler ab der 8. Klasse bis zur gymnasialen Oberstufe. Er kann im Rahmen der algorithmischen Problemlösung oder dem Thema Graphen durchgeführt und danach im Unterricht aufgegriffen werden.

Für die Durchführung ist kein Computerraum erforderlich. Die Materialien müssen entweder vorher von der Lehrperson ausgeschnitten oder gemeinsam mit den Schülerinnen und Schülern produziert werden. Die in der Geschichte eingebettete Pressekonferenz kann als Abschlusspräsentation im Unterricht genutzt werden.

[3] [**?**]

Die Aufgaben können sowohl in Einzel-, Partner- und Gruppenarbeit bearbeitet werden.

Im Rahmen des Workshops wurden fünf (Zeit-) Stunden Bearbeitungszeit eingeplant. Im Rahmen der Vereinfachung für den Unterricht gehen wir von einer Bearbeitungszeit von vier bis sechs Unterrichtsstunden aus.

2.1 Eulerkreise und -wege

Dieser Abschnitt ist dem Juristen zugeordnet. Er schickt den Lernenden einen Brief, in dem er die Situation schildert. Nach der Parkplatzverordnung der Stadt Darmstadt müssen bei größeren Festen, Parkplätze für die Besucher zur Verfügung gestellt werden. An Seitenrändern von Straßen sind diese schnell gefunden. Das Sicherheitsunternehmen fordert nun aber die beste Patrouillienplanung für ihr Personal, damit dieses keine unnötigen Wege doppelt gehen muss und trotzdem jede Straße gleichmäßig kontrolliert wird.

In dieser ersten Aufgabe machen sich die Schülerinnen und Schüler mit dem grundlegenden Problem vertraut. Auf Stadtkarten (siehe Abbildung 1) können sie zunächst ausprobieren, wo ein entsprechender Rundweg zu finden ist.

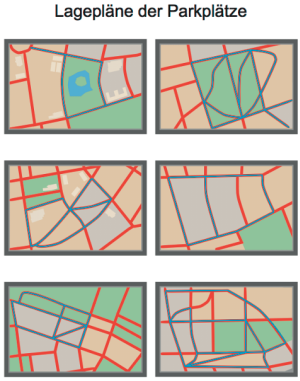


Abb. 1: Arbeitsblatt zu Eulerkreisen

In der zweiten Aufgabe muss ein ähnliches Problem gelöst werden. Für die Kehrmaschinen der Stadt müssen erneut Rundwege identifiziert werden. Dazu liegt ein stilisierter Stadtplan Darmstadts vor. Die Schülerinnen und Schüler sollen möglichst wenige Straßen sperren, um einen Rundweg möglich zu machen.

Für diese Aufgabe ist die Kenntnis des Eulerkriteriums sehr hilfreich. In einem ungerichteten, zusammenhängenden Graphen existiert ein Kantenzug, der jede Kante genau einmal enthält, genau dann, wenn jeder Knoten eine gerade Anzahl an adjazenter Kanten besitzt. Bei der Erforschung dieses Kriteriums helfen Zeitschriften, die sich diesem Thema widmen. Diese sind ebenfalls im Rahmen des Workshops entstanden und erklären humorvoll sowie schülergerecht verschiedene Themen.

Um vergleichbare Probleme in Zukunft schneller lösen zu können, verlangt die letzte Aufgabe eine Dokumentation der Problemlösung. Diese muss an dieser Stelle explizit nicht als formaler Algorithmus angegeben werden, es reicht eine prosaische Beschreibung aus.

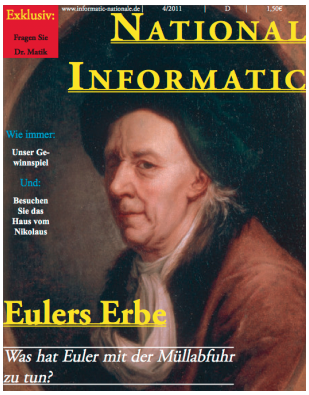


Abb. 2: Zeitschrift zum Thema Eulerkreise

2.2 Minimale Spannbäume

Bei dieser Aufgabe, die dem Schatzmeister zugeordnet ist, geht es um die Verkabelung einer Geisterbahn (siehe Abbildung 3). Verschiedene Attraktionen müssen mit dem Stromkasten verbunden werden, wobei jedoch nur starre, gerade Kabel verwendet werden können. Zusätzlich sorgen Hindernisse, wie bspw. Wände, Schienen der Bahn oder Attraktionen selbst, für höhere Kosten bei der Kabelverlegung.

Zwar können auch alle Attraktionen direkt mit dem Stromkasten verbunden werden, dies stellt sich aber als sehr teuer heraus. Ziel ist deshalb alle Attraktionen direkt oder über andere Attraktionen kostengünstig mit dem Stromkasten zu verbinden?

In der ersten Aufgabe werden die Schülerinnen und Schüler aufgefordert, verschiedene Wege auf der Karte auszumessen. Dabei kann die Entfernung zwischen zwei Punkten mit einem Lineal bestimmt werden. Dazu müssen noch die Kosten durch Hindernisse gerechnet werden, um die Gesamtkosten einer Leitung zu erhalten.

Schnell fällt den Lernenden auf, dass die Karte, so schön und bunt sie ist, doch sehr von der eigentlichen Aufgabe ablenkt. Mit einer darüber liegenden Folie markieren sie die einzelnen Steckdosen sowie die Verbindungen zwischen ihnen. Es entsteht ein Graph, wie in Abbildung 4 dargestellt.

Die Lernenden erhalten nun Karten, auf denen die verschiedenen Kosten für die Leitungen ersichtlich sind. Auf Blanko-Karten können die Schülerinnen und Schüler ihre, in der vorherigen Aufgabe herausgefundenen, Werte eintragen und somit auch in dieser Aufgabe verwenden. Es ist denkbar, dass die Lernenden für alle Leitungen die jeweiligen Kosten bestimmen. Da dies jedoch viel Zeit in Anspruch nimmt, bietet sich eine Abkürzung durch das Austeilen der verbleibenden Karten an.

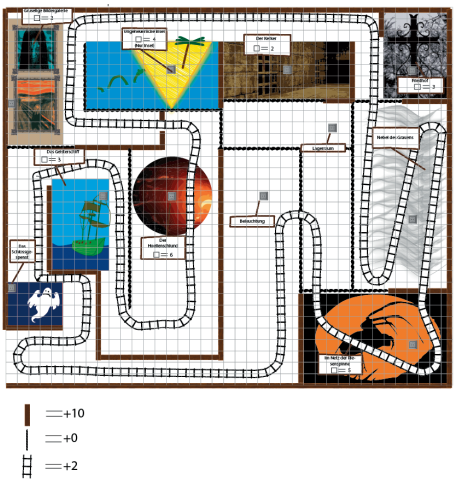


Abb. 3: Minimaler Spannbaum: Plan der Geisterbahn

Die günstigste Verkabelung lässt sich durch die Karten auch von Lernenden niedriger Jahrgangsstufen herausfinden. Zunächst werden die Karten aufsteigend sortiert. Es ist intuitiv, erst die günstigste Leitung zu verbauen, danach die zweitgünstigste usw. Wird ein Knoten, der schon verbunden ist, erneut verbunden (ein Zyklus entsteht), ist ersichtlich, dass diese Leitung nicht benötigt wird. Der Algorithmus von Kruskal wird auf diese Weise selbständig erarbeitet.

Auch hier gibt es wieder die Unterstützung durch eine Zeitung (Informatical Times), die Hilfestellungen für die Algorithmussuche liefert sowie Anwendungen und Probleme darstellt.

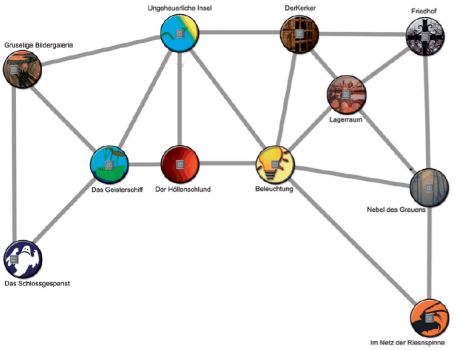


Abb. 4: Minimaler Spannbaum: Plan der Geisterbahn

Zum Abschluss dieser Aufgabe sollen die Lernenden einen Algorithmus formulieren. Um den Formalisierungsgrad weiter zu erhöhen, erhalten sie vorgefertigte Bausteine, mit denen sie eine Anweisung zur Lösung des Problems formulieren sollen. Ein paar Beispiele sind in Abbildung 5 zu sehen. Die Unterscheidung in Objekt und Merkmal des Objekts

wurde farblich vorgenommen. Durch Aneinanderreihung der Karten entsteht ein Ablaufdiagramm, das das Problem der Suche nach einem minimalen Spannbaum löst. Die Karten können laminiert und mit einem Magneten versehen werden, so dass sie auch an der Tafel verwendet werden können.

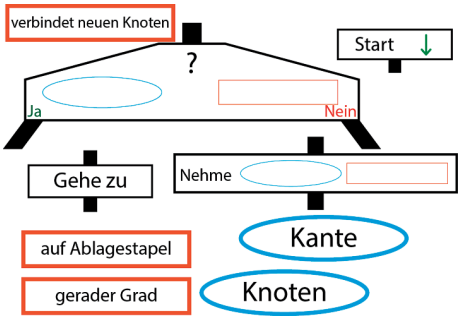


Abb. 5: Beispiele für Algorithmuskarten

2.3 Konfliktgraphen

Der Jurist schließt mit jedem Betreiber eines Standes einen Vertrag. Leider haben einige Standbesitzer Vorbehalte gegenüber anderen Ständen und wollen nicht neben diesen stehen. Sie befürchten Warenverwechslungen bei Anlieferungen und die generelle Konkurrenz.

In der ersten Aufgaben sollen die Schülerinnen und Schüler jeweils Dreiergruppen erstellen, die gemeinsam beliefert werden können. Dazu haben sie für jeden Standbesitzer eine Karte, auf der die Beziehungen zu den anderen Ständen vermerkt ist.

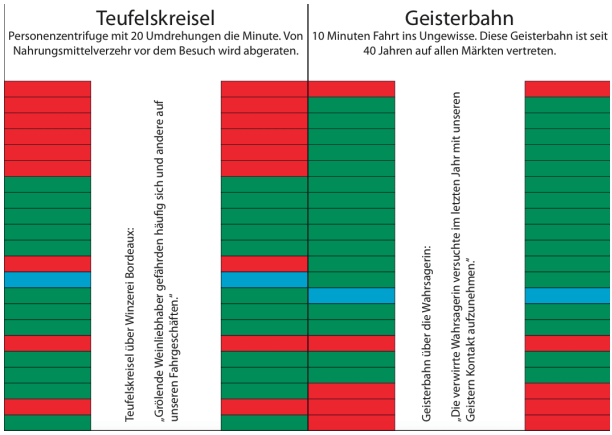


Abb. 6: Beispiele zweier Konfliktkarten

Jede Zeile der Randmarkierungen steht für einen Stand. Die Zeile, die die aktuelle Karte darstellt, ist blau eingefärbt. Zeilen, die grün eingefärbt sind, repräsentieren konfliktfreie Kombinationen; rote Zeile zeigen an, dass mit demjenigen Stand ein Konflikt besteht.

Werden zwei Karten nebeneinander gelegt, kann erkannt werden, ob zwischen diesen beiden Karten ein Konflikt besteht. Im Beispiel der Abbildung 6 gibt es keine Streitigkeiten zwischen den beiden Ständen, da jeweils die blaue Markierung auf eine grüne trifft.

Mit dieser Kodierung können durch Ausprobieren Dreiergruppen gebildet werden. Diese Vorgehensweise ist jedoch sehr mühsam, da die Vergleiche aufwändig sind. Aus diesem Grund stellt der Jurist in der zweiten Aufgabe eine weitere Darstellungsform vor: Stände werden erneut von Karten repräsentiert, auf denen die befreundeten Standbesitzer verzeichnet sind. Die Karten können laminiert und mit einem Magneten bestückt werden, so dass sie an der Tafel haften.

Mit Kreide können auf der Tafel die Freundschaftsbeziehungen durch einen Graph dargestellt werden. Die Suche nach Dreiergruppen ändert sich so in eine Suche nach vollständig verbundenen Cliquen der Größe 3. Diese Suche lässt sich mittels Backtracking realisieren. Für die Lösungssuche haben wir verschiedene Bearbeitungsmethoden. Karten, die schon einer Clique zugeordnet wurden, können aus dem Graphen herausgenommen, umgedreht oder markiert werden. Es ist jedoch wichtig, dass man diese Aktion wieder zurücknehmen kann, da dies das Kernelement des Backtrackings ist.

Der Graph kann auch invertiert werden: eine Kante repräsentiert nicht mehr eine Freundschaftsbeziehung, sondern einen Konflikt. Nun kann das Problem mittels Graphenfärbung gelöst werden.

In der dritten Aufgaben ist eine Vorschrift gefordert, mit der ein vergleichbares Problem gelöst werden kann. Wieder werden Bausteine vorgegeben, die aber diesmal von den Lernenden selbst beschriftet werden müssen.

2.4 Abschluss

Die Bearbeitung aller Aufgaben endet jeweils mit der Vorbereitung einer Pressekonferenz. Die Lehrperson kann sich Leitfragen für die Konferenz überlegen, um auf bestimmte Aspekte der Bearbeitung einzugehen. Die Pressekonferenz kann als Abschlusspräsentation dieser Unterrichtsreihe genutzt werden, um die gelernten Inhalte zu sichern.

3 Didaktisches Vorgehen und Erfahrungen

Die Unterrichtsreihe setzt auf selbstorganisiertes Lernen. Alle Materialien sind so aufgebaut, dass die Lernenden die nötigen Informationen aus den Materialien erlangen können. Generell sollte die Klasse mit vergleichbaren offenen Methoden schon einmal gearbeitet haben, um nicht durch die für sie neue Arbeitsweise überfordert zu werden.

Die Unterrichtseinheiten setzen gezielt auf algorithmische Problemlösung ohne Computerunterstützung. Dies hat zwei Gründe: Zum einen soll der wahrgenommene Fokus der Informatik vom Computer weg zum menschlichen Denken hin verschoben werden. Andererseits soll deutlich gemacht werden, dass Problemlösestrategien der Informatik auch weit

über die Anwendung im Computerkontext genutzt werden können. Ein positiver Effekt dieses Vorgehens ist die veränderte Wahrnehmung von Informatik und das Aktivieren von Lernenden, die Informatik bisher für sich als Interessensziel ausgeschlossen haben. Im Rahmen einer *Schnupperstunde Informatik* ist der vorgestellte Workshop ebenfalls denkbar.

Die Unterrichtssequenz wird von motivierenden und humorvollen Unterrichtsmaterialien getragen, die auch in einer reduzierten Version im Unterricht nichts von ihrem Motivationspotential verlieren. Im Rahmen eines Lehrerworkshops an der ETH Zürich konnten die neuen Materialien bereits erprobt werden.

Die Rahmenhandlung, die sich in allen Aufgaben wiederfindet, unterstützt eine spätere Reproduktion der Inhalte. Die Rahmenhandlung wird sehr häufig sehr gut behalten und mit ihr die vermittelten Kompetenzen des Workshops. Die Geschichte von Jurist, Schatzmeister und Magier dient also als Anker.

Die gemeinsame Arbeit an einem Problemkomplex (in diesem Fall der Rettung des Jahrmarkts) hat sich ebenfalls als motiverender Aspekt herausgestellt. Die gesamte Klasse arbeitet zusammen, um die Geschichte zu einem guten Ende zu bringen.

Die Wirksamkeit von Anchored Instruction ist nicht auf Jahrgangsstufen der Grundschule oder Sekundarstufe I beschränkt. Natürlich ist Schülerinnen und Schülern aller Jahrgangsstufen klar, dass die Rahmenhandlung nur erfunden ist. Wird die Geschichte, aber dennoch konsequent und mit einem Augenzwinkern erzählt, lassen sich auch Lernende der Oberstufe auf die Handlung ein. Jubel, Trubel, Informatik wurde bereits in einer Oberstufenklasse erfogreich getestet.

Der Workshop wurde im Lernlabor *Abenteuer Technik* mehrfach mit unterschiedlichen Klassen und Jahrgangsstufe durchgeführt und mit den gewonnen Erkenntnissen weiterentwickelt.

Literaturverzeichnis

[JG13a] Jens Gallenbacher, Dominik Heun: Ein moderner Ansatz für Anchored Instruction im Informatikunterricht. In (Breier, Norbert; Stechert, Peer; Wilke, Thomas, Hrsg.): Informatik erweitert Horizonte, INFOS 2013, 15. GI-Fachtagung Informatik und Schule, 26.-28. September 2013, Kiel, Germany. Jgg. P-219 in LNI. GI, S. 87–96, 2013.

[JG13b] Jens Gallenbacher, Dominik Heun, Kristin Rammelt: Einsatz von Speicherprogrammierbaren Steuerungen im Lernlabor Abenteuer Technik. In (Breier, Norbert; Stechert, Peer; Wilke, Thomas, Hrsg.): INFOS 2013, 15. GI-Fachtagung Informatik und Schule, Praxisband. Kiel Computer Science Series. Department of Computer Science, CAU Kiel, 2013.

[Pu07] Puhlmann, Hermann et al.: Grundsätze und Standards für die Informatik in der Schule. LOG IN, 146/147, 2007.

Persönliche Lernumgebungen – ein Beitrag zur Individualisierung des Lernens

Stefanie Gaßmann[1] und Henry Herper[2]

Abstract: Das Web 2.0 – das heutige Internet und die Interaktionen der Nutzer darin – bietet meist kostenlos vielfältige Werkzeuge und Anwendungen an, die im Unterricht genutzt werden könnten. Wie können diese Werkzeuge zu einer persönlichen Lernumgebung zusammengefügt werden? Persönliche Lernumgebungen sollen Komponenten enthalten, die den Lernenden die Möglichkeit geben, Inhalte nach ihren eigenen Bedürfnissen zu verwalten, zu verändern und selbst Inhalte zu erstellen. Die Persönliche Lernumgebung sollte ein gewisses Repertoire an Anwendungen beinhalten, um Lernprozesse bezüglich eines selbstgesteuerten und individuellen Lernens zu unterstützen. Es werden die für den erfolgreichen Einsatz dieser Lernumgebungen notwendigen technischen Voraussetzungen diskutiert. Verschiedene Umsetzungs- und Unterstützungsmöglichkeiten mit Persönlichen Lernumgebungen werden an Beispielen demonstriert. Gemeinsam mit den Teilnehmenden sollen Vorteile aber auch Risiken herausgearbeitet werden, die mit dem Einsatz einer Persönlichen Lernumgebung einhergehen. Im Rahmen des Workshops werden erste Ergebnisse des Projektes IMAILE - Innovative Methods for Award procedures within ICT Learning in Europe - vorgestellt, in dem Persönliche Lernumgebungen der nächsten Generation entwickelt werden.

Keywords: PLE, Lernumgebungen, Web 2.0

1 Einleitung

Digitale Medien durchdringen viele Bereiche unserer Arbeits-, Lern- und Lebenswelt immer stärker. Mobile Endgeräte sind in unterschiedlicher Form auch für Schüler zunehmend ständig verfügbar. Jugendliche zwischen dem 12. - 19. Lebensjahr nutzen täglich das Internet und stufen dieses für sich als eines der wichtigsten digitalen Medien ein. Trotz hoher Nutzungsfrequenz darf damit nicht äquivalent auf einen Zuwachs an einem rechtmäßigen und verantwortungsvollen Umgang mit diesen Medien geschlossen werden. Für Schülerinnen und Schüler ist es normal, Teil sozialer Netzwerke zu sein und Informationen zu teilen, zu bewerten und zu kommentieren. Nur sehr selten werden diese Informations- und Kommunikationstechnologien jedoch als Lernwerkzeuge für einen effizienten Wissenserwerb eingesetzt.

Dabei bietet das Web 2.0 – das heutige Internet und die Interaktionen der Nutzer darin – vielfältige Werkzeuge und Anwendungen an, die im Unterricht genutzt werden

[1] Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, PF 4120, 39016 Magdeburg, henry.herper@ovgu.de
[2] Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, PF 4120, 39016 Magdeburg, stefanie.gassmann@st.ovgu.de

könnten. Aus der großen Anzahl der angebotenen Werkzeuge und Inhalte können sich heute schon versierte Internetnutzer eine Persönliche Lernumgebung schaffen, die es ihnen erlaubt, aus der Informationsflut die geeigneten Informationen herauszufiltern und aufzubereiten sowie die vielfältigen Web-Anwendungen zu strukturieren. Mit Blick auf die informatische Kompetenz und Medienkompetenz muss hinterfragt werden, ob nicht alle Schüler die Grundkompetenzen erwerben sollten, gezielt mit ihrer Lernumgebung in die Informationsflut einzutauchen und für sich relevante Fakten herausfiltern zu können.

Die Persönliche Lernumgebung soll für Lernende, aber auch für Lehrende einsetzbar sein. Mit der Zunahme digitaler Unterrichtsmittel, wie interaktiver Displays und digitalen Endgeräte sowie der ständigen Verfügbarkeit von Netzwerkverbindungen ergeben sich auch für die Lehrenden neue Anforderungen, den Unterricht zu gestalten und ihre Inhalte zu verwalten und gemeinsam zu nutzen.

Besonders für strukturschwache Regionen, wie sie beispielsweise im Norden Sachsen-Anhalts vorzufinden sind, bieten Persönliche Lernumgebungen die Möglichkeit, Schülerinnen und Schülern an kleineren Schulstandorten ein breiteres Lehrangebot zu sichern und kollaboratives Arbeiten für Lehrende und Lernende zu ermöglichen.

Ein weiterer Baustein für die erfolgreiche Einführung Persönlicher Lernumgebungen ist die Verfügbarkeit geeigneter Inhalte. Die Aussage, dass im Internet ausreichend Wissen zur Verfügung steht, ist sicher richtig, aber dieses ist derzeit für die Schulen noch nicht geeignet. Im schulischen Bereich stellen die Schulbuchverlage mit den Lehrbüchern und Arbeitsheften die meisten Inhalte zur Verfügung. Beim Übergang von der Papierform zur digitalen Form sind Lizenzmodelle erforderlich, die es den Lernenden ermöglichen, die geeigneten Inhalte in ihre Persönliche Lernumgebung zu integrieren und mit dieser zu arbeiten.

2 Persönliche Lernumgebungen

Für den Begriff „Persönliche Lernumgebung“ gibt es viele voneinander abweichende Definitionen. Nach Schaffert und Kalz [Sch09] stellt eine Persönliche Lernumgebung eine Lernanwendung dar, in die der Lernende die Möglichkeit hat, verteilte Online-Informationen, -Ressourcen, oder -Kontakte zu integrieren. Zudem ermöglicht die Lernumgebung, dass mit ihr digitale Produkte erstellt und mit anderen Internetnutzern geteilt werden können. Dies spiegelt im Wesentlichen das Verhalten der heutigen Internetnutzer wider, welche sich zunehmend von Konsumenten hin zu Produzenten entwickeln. Vor allem Web-Anwendungen zur Kommunikation und Kollaboration prägen daher jene Lernumgebung. Eine weitere wesentliche Eigenschaft ist, dass in dieser Lernumgebung der Lernende im Zentrum steht. Es gibt demnach keine einheitlich vorgefertigte Lernumgebung, sondern jeder Lernende erstellt individuell seine eigene Persönliche Lernumgebung.

Im Projekt IMAILE[3] - Innovative Methods for Award procedures within ICT Learning in Europe – wird eine persönliche Lernumgebung wie folgt definiert:

„Persönliche Lernumgebungen (PLE) sind Systeme, die Lernende dabei unterstützen, ihren eigenen Lernprozess zu steuern und zu organisieren. Darin ist eine angebotene Unterstützung der Lernenden enthalten, um

- ihre individuellen Lernziele festzulegen (mit Unterstützung durch die Lehrenden),
- ihren Lernprozess und die notwendigen Inhalte zu organisieren und
- mit den anderen am Lernprozess beteiligten Personen zu kommunizieren.“ [IM15]

Worin unterscheiden sich die Persönlichen Lernumgebungen von derzeit verwendeten Lernumgebungen? Als digitale Lernplattformen werden heute oft Lernmanagementsysteme (LMS) eingesetzt, die häufig in den Landesinstituten für Lehrerbildung und Schulentwicklung und an Universitäten betrieben werden. Diese LMS werden häufig als „Inseln“ im Internet bezeichnet, weil sie oft nur einer geschlossenen Lerngruppe zur Verfügung stehen.

Die Persönliche Lernumgebung sollte ein gewisses Repertoire an Anwendungen beinhalten, um Lernprozesse bezüglich eines selbstgesteuerten und individuellen Lernens zu unterstützen. Dies können unter anderem Wikis sein. Sie ermöglichen den Schülern ein kollaboratives Erstellen von Beiträgen, welche mit Videos und Fotos ausgestaltet werden können. Insbesondere die Hypertextfunktion bietet den Schülern die Möglichkeit, ihr Gelerntes in der Gesamtheit zu verorten. Zum Erstellen von Lerntagebüchern eignen sich E-Portfolios und Blogs, die es den Lernenden ermöglichen, ihre Lernfortschritte zu dokumentieren. Chats und soziale Netzwerke können das Unterrichtsgespräch sinnvoll ergänzen, indem Diskussionsbeiträge, Anmerkungen oder offene Fragen schriftlich gesammelt werden und so für die folgende Unterrichtsstunde zur Verfügung stehen. Wird beispielsweise mit einem HTML-Editor von den Schülern eine Webseite erstellt, kann diese über die URL in die Persönliche Lernumgebung eingebunden werden und ggf. als ein weiteres Werkzeug genutzt werden, was zur Wertschätzung des Schülerproduktes führt.

Durch die vielfältigen multimedialen Elemente, die den Lernenden zur Integration in ihre Persönliche Lernumgebung angeboten werden können, ergeben sich im Unterricht größere Handlungs-, Gestaltungs- und Entscheidungsräume. Beispielsweise kann ein Musiklehrer den Schülern eine Sammlung verschiedener Links zur Verfügung stellen, in der Arbeitsblätter, Nachschlagewerke und Texte zum einem bestimmten Thema zu finden sind, aber ihnen auch die Möglichkeit geben, jene Musik anzuhören. Durch dieses breite Spektrum an Inhalten wird die Lernumgebung der Lernenden vielfältig. In ihr können sie individuell für sich interessante Inhalte und Werkzeuge heraussuchen und

[3]IMAILE – 619231: This project has been funded with support from the European Commission in the context of the Seventh Framework Program. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

eigenständig bearbeiten. Dadurch wird die Möglichkeit eines schülerzentrierten Unterrichts geboten, in dem die Lehrperson die Rolle des Beraters einnimmt.

Für Persönliche Lernumgebungen der nächsten Generation ist vorgesehen, dass neben den Möglichkeiten der Verwaltung von Werkzeugen und Inhalten auch Komponenten entwickelt werden, die den einzelnen Lernenden individuell in Abhängigkeit von seinem Lernfortschritt gezielt unterstützen. Der Lernfortschritt wird individuell erfasst. Es ist jedoch auch klar, dass für eine solche Unterstützung eine sehr umfangreiche Erfassung und Speicherung aller Aktivitäten der Lernenden personalisiert notwendig ist. Dabei sind die Aspekte des Datenschutzes und der Datensicherheit zu berücksichtigen.

Im Rahmen des Workshops werden Beispiele für derzeit verfügbare Persönliche Lernumgebungen vorgestellt.

3 Technische Voraussetzungen für den Einsatz

Die Umsetzung digitaler Persönlicher Lernumgebungen im schulischen und im häuslichen Umfeld erfordern entsprechende technische Voraussetzungen. Die Schulen, aber auch die Elternhäuser der Lernenden müssen über eine geeignete IT-Infrastruktur verfügen, die den störungsfreien Einsatz der Lernumgebungen ermöglicht. Die erste zu diskutierende Frage ist: „Wo befindet sich die Lernumgebung?“ Als Möglichkeiten stehen der Schülerrechner, die Schulcloud, eine Landescloud oder ein externer, ggf. weltweit agierender Anbieter zur Auswahl. Da umfangreiche persönliche Daten der Lernenden erfasst werden, ist eine sorgsame Auswahl eines vertrauenswürdigen und zuverlässigen Anbieters erforderlich.

Für den schulischen Einsatz ist eine geeignete Infrastruktur vorzuhalten. Diese beginnt mit einer ausreichend breitbandigen Anbindung der Schule an das Internet. Da als Endgeräte Tablets, Laptops und Smartphones eingesetzt werden, ist von der Schule eine geeignete WLAN-Infrastruktur zu betreiben, die den Zugriff in allen Klassenräumen und in allen Unterrichtsfächern ermöglicht.

Soll die Persönliche Lernumgebung auf allen Endgeräten verwendbar sein, so ist derzeit nur eine browserbasierte Lösung möglich. Dieser Ansatz würde dem Konzept BYOD – bring your own device – entgegenkommen.

Bei der technischen Umsetzung sind auch die Aspekte der Datensicherheit zu beachten. Zugriffsrechte müssen individuell, aber auch für Lerngruppen oder Klassen vergeben werden können. Die Daten müssen ständig verfügbar sein und gesichert werden. Es ist zu klären, welche Rechte der Schule und dem Lehrenden eingeräumt werden, auf die Persönliche Lernumgebung zuzugreifen.

Im Rahmen des Workshops wird der aktuelle Entwicklungsstand des IMAILE-Projektes mit den Teilnehmern diskutiert. Die gewonnenen Ergebnisse fließen in die weitere Entwicklung des Projektes ein.

4 Fazit

Der erfolgreiche Einsatz von Persönlichen Lernumgebungen erfordern von Lehrenden und Lernenden umfangreiche neue Kompetenzen. Es reicht nicht aus, einen Computer bedienen zu können, sondern er muss bewusst zur Gewinnung und Festigung von Kompetenzen eingesetzt werden. Eine fundierte informatische Bildung und Medienbildung sind wesentliche Voraussetzungen zum erfolgreichen Einsatz dieser Systeme. Lehrende und Lernende müssen die Regeln beachten, die sich beispielsweise aus Urheber-, Verwertungs- und Persönlichkeitsrechten ergeben.

Eine Akzeptanz dieser Systeme bei den Lehrenden erfordert eine umfangeiche Weiterbildung. Die einzelnen Fachdidaktiken sind aufgefordert, Unterrichtsmethoden zu entwickeln, die den erfolgreichen Einsatz digitaler Lernwerkzeuge und Persönlicher Lernumgebungen unterstützen. Die technischen und inhaltlichen Rahmenbedingungen sind von den Schulträgern und Kultusbehörden sicherzustellen.

Die Befürchtung, dass Persönliche Lernumgebungen der nächsten Generation den Lehrenden ersetzten sollen, können wir nicht teilen. Die Rolle des Lehrenden wird sich verändern, aber er wird weiterhin mit seiner Ausbildung und seinen Erfahrungen die wesentliche Komponente in Lehr- und Lernprozessen sein.

Durch die Gestaltung der Persönlichen Lernumgebung erwerben die Lernenden Kompetenzen, die einen wesentlichen Beitrag für das lebenslange Lernen leisten.

Literaturverzeichnis

[IM15] IMAILE, http://www.imaile.eu/ple-personal-learning-environments/, Stand: 27.04.2015.

[SK09] Schaffert, S.; Kalz, M.: Persönliche Lernumgebungen: Grundlagen, Möglichkeiten und Herausforderungen eines neuen Konzepts, http://dspace.ou.nl/bitstream/1820/1573/1/schaffert_kalz_ple09_dspace.pdf, 2009, Stand: 27.04.2015.

Jens Gallenbacher (Hrsg.): INFOS 2015 – Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 125

Big Data im Informatikunterricht: Motivation und Umsetzung

Andreas Grillenberger und Ralf Romeike[1]

Abstract: Das Sammeln und Auswerten von Daten ist heute allgegenwärtig: In vielen Bereichen des täglichen Lebens nimmt die Bedeutung von Daten und datenbezogenen Anwendungen immer mehr zu, z. B. bei der Nutzung sozialer Medien oder bei der Verwaltung großer Mengen an eigenen Daten. Während Daten früher hauptsächlich konsumiert wurden, wird heute auch jeder zum Produzenten immer umfangreicherer Datenmengen. Dabei werden immer größere und vielfältigere Datenmengen verwaltet und verarbeitet. Im Informatikunterricht wird Big Data jedoch bisher kaum thematisiert: Die fachlichen Grundlagen dafür scheinen auf den ersten Blick zu komplex und kaum auf schulischem Niveau verständlich zu sein. In diesem Beitrag werden daher zuerst die wesentlichen Entwicklungen vorgestellt, die sich derzeit im Datenmanagement ereignen, sowie die sich dadurch ergebenden fachlichen Herausforderungen. Um zu demonstrieren, dass solche fachliche Innovationen oft grundlegende Konzepte enthalten, die im Informatikunterricht thematisiert werden können, wird anhand von zwei Unterrichtsszenarien aus diesem Themenbereich exemplarisch vorgestellt, wie Informatik es mittels moderner Ansätze zur Datenverarbeitung ermöglicht, Big Data beherrschbar zu machen.

Keywords: Daten, Datenmanagement, Datenbanken, Kompetenzen, Big Data, NoSQL, Datenanalyse, Datenstromsysteme, Datenverarbeitung, Datenvisualisierung, Alltag, Datenschutz

1 Einleitung

Big Data gehört derzeit wohl zu den maßgeblichen Erscheinungen der Informatik, die alles und jeden betreffen. Nicht nur in Informatik, Wirtschaft und Gesellschaft spielt dieses Thema, die Verarbeitung von großen und vielfältig strukturierten Datenmengen in hoher Geschwindigkeit, in den letzten Jahren eine immer größere Rolle. Auch die Möglichkeiten und Konsequenzen werden heute in der Gesellschaft ausführlich diskutiert: So bestimmen die negativen Seiten dieser Entwicklungen, z. B. die Möglichkeiten der Geheimdienste zur Sammlung und Analyse umfangreicher Datenmengen, seit geraumer Zeit die Nachrichten. Andererseits eröffnen öffentlich verfügbare Datenquellen willkommene Möglichkeiten, die bis vor kurzem noch undenkbar waren. Beispielsweise analysiert *Google Flu Trends* Suchanfragen und prognostiziert dadurch Grippewellen oft genauer und auch wesentlich schneller als herkömmliche Vorhersagen. Analog nutzen Herausgeber von Kreditkarten zunehmend Datenanalysen um Bezahlvorgänge in Echtzeit auf mögliche Betrugsfälle zu prüfen und betroffene Transaktionen abzulehnen [MC13].

Während in einigen Fällen die Datenerfassung und -auswertung für den Nutzer offensichtlich ist, beispielsweise bei intelligenten Stromzählern oder Smart Watches, die ihre Auswertungen dem Nutzer zur Verfügung stellen, werden jedoch auch große Datenmengen im Verborgenen generiert. Smartphones erfassen kontinuierlich die Bewegungsdaten

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, Martensstr. 3, 91058 Erlangen
andreas.grillenberger@fau.de, ralf.romeike@fau.de

ihrer Nutzer und teilen diese sogar dem Hersteller mit, Webseiten (und zunehmend auch Desktop-Anwendungen) protokollieren genaue Nutzeraktionen, selbst Taxifahrten werden heute detailliert erfasst und ausgewertet [Bu14]. Diese Datensammlungen bieten oft einen direkten Mehrwert für den Nutzer: Das Smartphone kann bei Verlust oder Diebstahl geortet werden, die Nutzererfahrung von Anwendungen wird verbessert und Taxis können schon vorab zu stark frequentierten Orten gesendet werden, um die Wartezeiten zu verkürzen. Die Bereitschaft, an datenbasierten Anwendungen zu partizipieren, ergibt sich daher aus Faktoren wie dem Vertrauen in den Anbieter, dem persönlich erfahrenen Nutzen oder dem Umfang der Datenerfassung [Wo14]. Um diese Aspekte, insbesondere das Vertrauen in einen Anbieter und die Möglichkeiten, die sich durch die Datenerfassung ergeben, einschätzen zu können, ist jedoch ein Verständnis der Grundlagen, Möglichkeiten und Risiken moderner Datenverarbeitung nötig.

Trotz sichtbarer Bemühungen, Themen wie Datenschutz im Unterricht zu berücksichtigen, konzentriert sich Informatikunterricht zum Thema *Daten* bisher vor allem auf den Kontext *Datenbanken* [GR14]. Themen die heute allgegenwärtig sind, z. B. Echtzeit-Datenanalysen, Datenqualität, Metadaten oder die Datennutzung im Alltag, werden bisher kaum betrachtet, wie eine Analyse verschiedener Lehrpläne und Bildungsstandards zeigt [GR14]. Auch in der Fachdidaktik werden diese Entwicklungen bisher kaum thematisiert. Obwohl diese Themen auf komplexen Grundlagen wie Statistik beruhen, zeigen sich die Auswirkungen und Vorgehensweisen aus dem Bereich Big Data jedoch in einer Vielzahl von Gebieten und werden auch außerhalb der Informatik, für hochkomplexe aber auch relativ einfache Zwecke, eingesetzt. Selbst Datenwissenschaftler wie Halevy et. al. [HNP09] schließen, dass einfache mathematische Modelle in Kombination mit großen Datenmengen oft bessere Ergebnisse liefern als komplexe Modelle mit wenigen Daten. Auch im Kontext von Big Data werden viele der bereits seit Jahren eingesetzten Methoden weiterhin verwendet, beispielsweise die Datenanalysemethoden „Klassifikation“, „Clusteranalyse“ und „Assoziation“. Insbesondere die Assoziationsanalyse gewinnt heute stark an Bedeutung. Aufgrund der Verbreitung dieses Themas, dem Einsatz auf verschiedenen Komplexitätsstufen und der zeitlichen Beständigkeit vieler der Grundlagen, kann angenommen werden, dass die weitgreifenden Innovationen im Datenmanagement informatische Grundlagen im Sinne der fundamentalen Ideen [Sc93] beinhalten, die den Informatikunterricht auch über das Thema „Datenbanken“ hinaus wesentlich bereichern können. Damit zeigt sich unmittelbar, dass es eine wichtige Herausforderung für den Informatikunterricht ist, schülergerechte Zugänge zu diesen Themen zu finden. Im Folgenden skizzieren wir daher die grundlegenden Entwicklungen im Datenmanagement und zeigen anhand von zwei Beispielen exemplarisch, wie solche Innovationen aus der Informatik für den Informatikunterricht didaktisch reduziert und somit zugänglich gemacht werden können.

2 Big Data: Grundlagen und aktuelle Entwicklungen

Big Data ermöglicht vielfältige Innovationen im Bereich Datenverwaltung und -verarbeitung, die Voraussetzung für Informatiksysteme sind, wie wir sie heute täglich benutzen: Soziale Netzwerke nutzen *nicht-relationale Datenbanken*, z. B. zur Speicherung von Freundschaftsbeziehungen oder zur Verbesserung der Suchmöglichkeiten [ER13], Daten werden zu verschiedensten Zwecken in *Echtzeit analysiert*, immer häufiger in der *Cloud* gespeichert

und dabei auf eine große Anzahl von *Systemen verteilt*. Diese technischen Innovationen führen zu neuartigen Auswertungsmöglichkeiten von Daten, die immer öfter eine Grundlage für den wirtschaftlichen Erfolg von Unternehmen darstellen: Beispielsweise prognostiziert Amazon mittlerweile gut genug welche Artikel ein Kunde möglicherweise kaufen wird, um diese schon vorher in ein naheliegendes Versandzentrum zu verlegen [He14]. Der Umgang mit Daten hat sich dabei in den letzten Jahren stark gewandelt: Während die Verarbeitung und Speicherung von Daten bisher insbesondere auf der Nutzung von relationalen Datenbanken basierte, ist heute eine Ausweitung dieses Fachgebiets erkennbar. Neben Datenbanken bzw. der Verwaltung strukturierter Daten gewinnen im nun oft als *„Datenmanagement"* bezeichneten Fachgebiet auch weitere Aspekte an Bedeutung, z. B. die Verwaltung un- bzw. wenig strukturierter Daten, Datensicherheit und Datenschutz, gemeinsame Datennutzung, Metadaten und auch Datenqualität (vgl. [DA09]).

Eine grundlegende Innovation stellt dabei die Möglichkeit zur Verarbeitung von *Big Data* dar. Während dieser Begriff auf den ersten Blick insbesondere auf große Datenmengen (*volume*) hindeutet, wird er nach Laney [La01] noch durch zwei weitere „V"s charakterisiert (vgl. Abb. 1): Die unterschiedliche Strukturierung von Daten (*variety*) sowie deren immer schnellere Erzeugung und Verarbeitung (*velocity*). Diese Entwicklung stellt die Verarbeitung und Speicherung von Daten in Datenbanken vor neue Herausforderungen: Neben traditionellen Aspekten, wie Konsistenz, gewinnt heute die verteilte Datenspeicherung stark an Bedeutung. Steigende Datenmengen und Verarbeitungsgeschwindigkeit verhindern die Verarbeitung auf einem einzelnen Datenbankserver (selbst wenn dieser hoch performant ist), eine *vertikale Skalierung* reicht daher für den Umgang mit Big Data nicht aus. Stattdessen muss *horizontale Skalierung* eingesetzt werden: Die Daten werden auf viele (im Einzelnen weniger performante) Server verteilt und parallel verarbeitet [FU14].

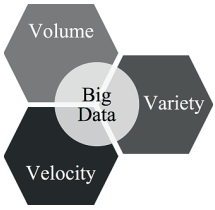


Abb. 1: Drei-V-Modell für Big Data

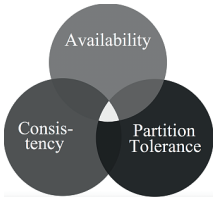


Abb. 2: CAP-Theorem

Bei der verteilten Speicherung von Daten auf mehreren Datenbankservern muss jedoch, wann immer eine Anfrage einen konsistenten Zustand benötigt, dieser erst durch Synchronisation und Prüfung verschiedener Bedingungen (z. B. constraints und das definierte Datenschema) sichergestellt werden. Konsistenz sorgt daher bei der parallelen Datenspeicherung für eine wesentliche Verlangsamung. Dieser inhärente Widerspruch wird durch das *CAP-Theorem* [Br12] beschrieben: Nur zwei der Eigenschaften Konsistenz (***C****onsistency*), hohe und schnelle Verfügbarkeit (***A****vailability*) sowie Partitionstoleranz (***P****artition tolerance*, Möglichkeit zur verteilten Speicherung von Daten) können gleichzeitig sichergestellt werden (vgl. Abb. 2). Bei vielen modernen Datenbankanwendungen steht insbesondere die Verfügbarkeit und verteilte Speicherung im Vordergrund, sodass die bisher meistgenutzten relationalen Datenbanken nicht mehr ausreichen: Diese stellen insbesondere Konsistenz und Verfügbarkeit, nicht aber Partitionstoleranz, sicher. Daher entstehen derzeit vielfältige neue Datenbankmodelle wie Graphdatenbanken (z. B. *Neo4J*) oder dokumentenorientierte Datenbanken (z. B. *CouchDB*), die unter dem Begriff *NoSQL*[2] zusammengefasst

[2] *NoSQL* wird heute als „Not only SQL" interpretiert [Ed11] und als Sammelbegriff für nicht-relationale Datenbanksysteme verwendet.

werden. Diese Datenbanken heben üblicherweise eine wesentliche Einschränkung des relationalen Modells auf: Es wird seitens der Datenbank kein Schema erzwungen, sodass die Konsistenz der Daten nicht durch die Datenbank sichergestellt werden kann, sondern in den Verantwortungsbereich der Anwendung verlagert wird. Dadurch wird jedoch eine wesentlich schnellere Datenverarbeitung ermöglicht. Im Gegensatz zum *ACID*-Paradigma[3] bei relationalen Datenbanken, liegt den NoSQL-Datenbanken daher das *BASE*-Paradigma zugrunde: **B**asically **A**vailable, **S**oft-state, **E**ventually consistent [Ed11].

3 Big Data als Unterrichtsgegenstand

Die Betrachtung der fachlichen Innovationen im Datenmanagement zeigt sich eine Vielzahl neuartiger Konzepte und Vorgehensweisen, die grundlegend für das Verständnis von Phänomenen wie Cloudcomputing und Big Data sind. Es kann vermutet werden, dass diese Konzepte prototypisch für fundamentale Ideen stehen, die zum Erschließen der im Kontext von Big Data entstehenden Phänomene wichtig sind. Im Folgenden stellen wir daher anhand zweier Unterrichtsszenarien die Möglichkeit dar, grundlegende Aspekte von Big Data auf schulischem Niveau zu vermitteln.

3.1 Schürfen nach Daten: Data Mining in Big Data

Um Entscheidungen darüber treffen zu können, ob man Informatiksysteme nutzt, die Daten in großem Umfang sammeln und analysieren, ist es nötig, das Potential und die Gefahren der Big-Data-Verarbeitung einschätzen zu können. Eine der größten Herausforderungen bei der Verarbeitung von Big Data stellt heute die Analyse der Daten dar, wobei sich die Herangehensweise oft deutlich von klassischen Datenanalysen unterscheidet: Statt Daten für eine spezielle Analyse gezielt zu sammeln, wird heute versucht, aus auf Vorrat gesammelten Daten wertvolle Informationen zu gewinnen [DN13]. Zum Zeitpunkt der Datensammlung ist dabei typischerweise weder die Existenz der zu gewinnenden Informationen, noch der Analysezweck bekannt. Diese Analysen werden daher oft, in Analogie zum Goldbergbau, als *Data Mining* bezeichnet: Das Schürfen nach wertvollen Informationen im großen Datenberg[4]. In diesem Zusammenhang werden Daten auch als das neue Öl des 21. Jahrhunderts beschrieben [Wo11]. Die zur Datenanalyse eingesetzten Methoden setzen dabei andere Schwerpunkte als bisher: Durch eine andere Sichtweise auf Daten und die primäre Suche nach Korrelationen statt Kausalitäten können vielfältige Ergebnisse gewonnen werden, die bisher kaum möglich waren.

Unterrichtsgegenstand Data-Mining In diesem Beispiel stellen wir im Kontext der Gewinnung von Informationen aus verschiedenen *Open-Data*[5]-Datensätzen die Thematisierung einiger grundlegender Data-Mining-Methoden im Informatikunterricht beispielhaft dar. Obwohl Datenbanken dabei als Werkzeug eingesetzt werden und daher auch Kompetenzen

[3] ACID: **A**tomicity, **C**onsistency, **I**solation, **D**urability [KE13]

[4] Der Begriff *Data Mining* ist in Bezug auf die Analogie jedoch missverständlich, da diese eher den Begriff *Information Mining* nahelegt, da es um den Abbau von Informationen geht.

[5] Unter *Open Data* werden Datensätze verstanden, die frei zugänglich zur Verfügung gestellt werden.

in diesem Bereich erworben werden können, wird der Schwerpunkt hier auf die Grundlagen und Möglichkeiten von Datenanalysen gelegt und Schrittweise an diese herangeführt[6]. Als Datenquelle für dieses Beispiel wählen wir zwei Open-Data-Sätze, die alle Vorfälle, die der Polizei von San Francisco im Jahr 2014 gemeldet wurden[7], sowie alle Anrufe, die dort bei der Servicenummer „311“ seit 2008 eingingen[8], beinhalten. Moderne Datenanalyse-Tools (z. B. „Tableau Public“, „Microsoft PowerQuery“ oder „CartoDB“) ermöglichen einen direkten Zugriff auf solche Datensätze. Eine wichtige Grundlage aller Data-Mining-Prozesse sind die Datenanalysemethoden *Klassifikation*, *Assoziation* und *Clusterbildung*. Während die Klassifikation von Daten auch im traditionellen Datenbankunterricht in Form von Gruppierungen vorkommt, stellen die Clusterbildung (Zusammenfassen von Datensätzen, die bezüglich der zu analysierenden Eigenschaft gleich oder ähnlich sind) und das Prinzip der Assoziation (d. h. von einem Attributwert oder einer Menge von Attributwerten auf weitere Eigenschaften zu schließen, die nicht explizit im Datensatz genannt sind) neue Vorgehensweisen dar.

Für viele Analyseziele reicht es dabei heute aus, Korrelationen zwischen Daten zu erkennen anstatt nach Kausalzusammenhängen zu suchen: *Google Flu Trends* sucht beispielsweise nach Suchbegriffen, die als Indikator für eine Grippeerkrankung dienen können. Für die Prognose der Ausbreitung von Grippewellen ist es nebensächlich, ob ein Sinn-Zusammenhang zwischen einem Suchbegriff und einer Grippeerkrankung erkennbar ist, eine hohe Korrelation tatsächlicher Grippeerkrankungen und der Suche nach einem Begriff aus. Dies wird möglich, da heute (idealerweise) alle verfügbaren Informationen zum Gegenstand der Untersuchung gespeichert und somit analysiert werden können, sodass wesentlich geringere Stichprobenfehler vorliegen als bei klassischen Analysen, die sich auf kleine Datenmengen oder beschränkte Stichproben konzentrieren. Durch die meist ungeprüfte Speicherung der verfügbaren Daten besteht jedoch die Gefahr, dass durch ungenaue oder fehlerbehaftete Daten auch fehlerhafte Ergebnisse entstehen. Bei ausreichend großen Datenmengen kann jedoch davon ausgegangen werden, dass die Fehler nur einen geringen Anteil ausmachen und somit das Ergebnis, wenn überhaupt, nur geringfügig negativ beeinflussen.

Ziel des Unterrichts ist es, dass die Schülerinnen und Schüler …

- einfache Datenanalysen an vorgegebenen Datensätzen durchführen
- die Datenanalysemethoden Klassifikation und Assoziation am Beispiel nachvollziehen und erklären
- die Möglichkeiten und Gefahren von Big-Data-Analysen erkennen
- verstehen, dass die Qualität der gewonnenen Information nicht nur von der Analyse der Daten sondern insbesondere auch von deren Interpretation abhängt
- erkennen, dass der Einfluss der Datenqualität mit ansteigender Datenmenge abnimmt
- den Unterschied zwischen Kausalität und Korrelation erkennen und am Beispiel erklären

[6] Je nach organisatorischen Gegebenheiten (z. B. Lehrplan) kann der Schwerpunkt jedoch auch stärker in Richtung relationaler Datenbanken verschoben werden.

[7] https://data.sfgov.org/Public-Safety/SFPD-Incidents-from-1-January-2014/tmnf-yvry (abgerufen: 20.04.2015)

[8] https://data.sfgov.org/City-Infrastructure/Case-Data-from-San-Francisco-311-SF311-/vw6y-z8j6 (abgerufen: 20.04.2015)

Ein möglicher **Unterrichtsablauf** gliedert sich in folgende Phasen:

1. **Kennenlernen von Big Data:** Durch Untersuchung eines großen Datensatzes erkennen die Lernenden den Aufbau von Datensätzen sowie die Bedeutung der verschiedenen Attribute und können Ideen zu den darin implizit enthaltenen Informationen sammeln.
2. **Klassifizierung von Daten:** Beim Filtern des Kriminalfälle-Datensatzes nach Stadtteilen und Bestimmung der Anzahl gemeldeter Ereignisse in diesem lernen die Schülerinnen und Schüler die Klassifizierung von Daten nach gegebenen Merkmalen kennen. Nebenbei kann die Aussagekraft von Analyseergebnissen am Beispiel diskutiert werden: Eine mögliche Fehlinterpretation der gewonnenen Information wäre, dass ein Stadtteil aufgrund einer höheren Zahl von Kriminalfällen in diesem gefährlicher ist.
3. **Assoziationen zwischen Daten:** Es kann festgestellt werden, dass die Adresse auch den Stadtteil festlegt. Während diese Assoziation offensichtlich ist, sind es andere weniger: Auch die Beschreibung eines Vorfalls (anscheinend ein vordefinierter Text) legt die Kategorie fest. Im Unterricht kann anhand dieser beiden Beispiele das Weglassen der redundanten Attribute Stadtteil bzw. Kategorie diskutiert werden. Dabei ist es wichtig, den Lernenden bewusst zu machen, dass anhand des Datensatzes nur eine Korrelation zwischen den beiden Attributen gefolgert werden kann, aber keine Kausalität. Durch das Weglassen können daher Fehler entstehen. Weitere Informationen, wie zum Beispiel beim Zusammenhang zwischen Adresse und Stadtteil vorhanden, können jedoch einen Kausalzusammenhang untermauern, sodass derartige Fehler ausgeschlossen werden.
4. **Verknüpfung von Datensätzen:** Aus dem Datensatz können verschiedene Informationen auf einfache Weise gewonnen werden, beispielsweise die Abhängigkeit der Anzahl an Delikten vom Stadtteil: Die Daten wurden in geeigneter Weise gesammelt, um solche Auswertungen durchführen zu können. Falls jedoch weitergehende Informationen gewonnen werden sollen, müssen weitere Daten herangezogen werden. Mit dem Datensatz ist es beispielsweise nicht direkt möglich zu analysieren, ob eine Korrelation zwischen Verkehrsunfällen und Ausfällen der Straßenbeleuchtung vorliegt. Es existiert jedoch auch kein Datensatz, in dem diese Information direkt enthalten ist. Durch Annahme einer Assoziation, nämlich dass im Fall einer ausgefallenen Beleuchtung eine Meldung bei der zuständigen Behörde eingeht, kann diese Information jedoch gewonnen werden: Die Anrufe bei der Servicenummer 311 der Stadt San Francisco liegen als Datensatz vor und können unter der Annahme dieser Assoziation in einen neuen Datensatz, der die benötigten Informationen beinhaltet, überführt werden. Indem diese Informationen und der ursprüngliche Datensatz im Rahmen eines Mash-Up[9] zusammengefasst werden, kann untersucht werden, ob die gesuchte Korrelation vorliegt.
5. **Rückblick:** Im Rückblick kann erkannt werden, dass selbst wenige zusätzliche Daten dazu beitragen können, wesentlich umfangreichere Informationen zu gewinnen. Dabei zeigt sich, dass Fehler in den Daten, wie sie beispielsweise auch durch die ungenaue Assoziation „*keine Beschwerde → Beleuchtung funktionsfähig*“ produziert werden, sich zwar im Einzelfall negativ auswirken, bei großen Datenmengen jedoch nur noch einen geringen Einfluss haben.

[9] Unter einem *Mash-Up* wird allgemein die Verknüpfung verschiedener Datensätze verstanden. In relationalen Datenbanken kann dies mit einen Join verglichen werden.

Ein wichtiges Merkmal von Data Mining, das durch dieses Beispiel verdeutlicht wird, ist der geänderte Umgang mit Daten: Statt für bestimmte Zwecke gesammelte Daten auszuwerten, können heute auch große Datenmengen auf unbekannte oder bei der Sammlung der Daten noch nicht angestrebte bzw. vermutete Zusammenhänge untersucht und dadurch wertvolle Informationen gewonnen werden.

3.2 Fischen im Unendlichen: Datenstromsysteme

Eine grundlegende Herausforderung stellt heute die Verarbeitung von Big Data (nahezu) in Echtzeit dar: Viele Anwendungen, wie die Live-Erkennung von Kreditkartenbetrug oder die Auswertung von Sensordaten (u. a. bei der Tsunami-Erkennung), benötigen eine schnelle Datenauswertung, müssen aber zugleich umfangreiche Datenmengen verarbeiten.

Unterrichtsgegenstand Datenströme Obwohl es heute möglich ist, immer größere Datenmengen zu speichern und zu analysieren, ist dieses Vorgehen nicht in allen Fällen zielführend: Bei der Betrachtung von kontinuierlichen Datenströmen, die u. a. durch soziale Medien oder Sensoren erzeugt werden, enthält in der Regel nur ein Teil der verfügbaren Daten relevante Informationen (beispielsweise, dass ein definierter Grenzwert überschritten wurde). Gleichzeitig wird die schnelle Verfügbarkeit der Analyseergebnisse, idealerweise in Echtzeit, immer grundlegender (z. B. im automatisierten Börsenhandel). Der bisherige Ansatz, alle erzeugten Daten in einer Datenbank zu speichern, ist daher in diesen Fällen weniger geeignet. Stattdessen reicht es aus, nur die für den konkreten Zweck benötigten Daten auszuwählen und direkt zu verarbeiten, sodass eine weitere Speicherung nicht nötig ist. Im Vergleich zu einer Datenbank, bei der die Daten permanent gespeichert und verschiedene Abfragen auf diese angewendet werden (vgl. Abb. 3), verarbeiten daher sog. *Datenstromsysteme* den Datenstrom direkt, indem vorher definierte Abfragen auf den Datenstrom angewendet werden, ohne dass eine dauerhafte Datenspeicherung nötig ist (vgl. Abb. 4). Anschaulich kann das Prinzip einer Datenbank mit einem Hamster verglichen werden, der sein Futter auf Vorrat sammelt, während ein Datenstromsystem dagegen einem Bären ähnelt, der nur Fische aus einem Fluss fischt um diese direkt zu fressen. In einem Datenbanksystem können daher jederzeit weitere Anfragen auf bereits analysierte Daten angewendet werden, während in einem Datenstromsystem die Daten nach der Verarbeitung verloren sind[10]. In Zusammenhang mit Big Data gewinnen diese Systeme immer stärker an Bedeutung: Während die schnelle Verarbeitung sehr großer Datenmengen mit bewährten Systemen kaum möglich ist[11], stellen Datenstromsysteme eine

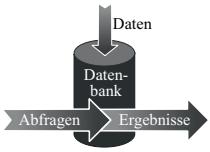


Abb. 3: Funktionsweise eines DBMS

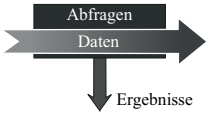


Abb. 4: Funktionsweise eines DSS

[10] Häufig wird dem entgegengewirkt, indem alle oder ein Ausschnitt der Daten neben der Echtzeitverarbeitung im Datenstromsystem in einem anderen System (z. B. Datenbank) für längere Zeit gespeichert werden.

[11] Um den Twitter-Nachrichtenstroms zu analysieren müssten ca. 6000 Tweets pro Sekunde abgespeichert und verarbeitet werden [Kr13]. Nimmt man als Größe eines Tweets inklusive aller Metadaten, wie GPS-Position oder Hashtags, durchschnittlich 500 Byte an, wächst die Datenmenge sekündlich um 3 MB an, täglich daher um über 250 GB. Die Speicherplatzgrenzen typischer Datenbanksysteme wären damit nach wenigen Tagen erreicht.

effiziente Lösung dar und ermöglichen durch direkte Verarbeitung schnelle Analyseergebnisse, aber auch eine hohe Speicherplatzeffizienz. Im Kontext von Big Data können daher die Grenzen und Funktionsweise moderner und im Alltag oft präsenter Echtzeitsysteme verdeutlicht werden.

Ziel des Unterrichts ist es, dass die Schülerinnen und Schüler ...

- erkennen, dass selbst große Datenmengen mit geeigneten Informatiksystemen in nahezu Echtzeit verarbeitet werden können
- die Grenzen der Verarbeitung großer Datenmengen mit Datenbanken und Datenstromsystemen erkennen und erklären
- Datenstromsysteme als Lösung für die Herausforderungen bei der Verarbeitung von Big Data erläutern
- mögliche Probleme und Fehlerquellen moderner Echtzeit-Datenanalysen erkennen
- die Alltagsrelevanz von Echtzeitanalysen erkennen und Beispiele für deren Nutzung im Alltag nennen

Ein möglicher **Unterrichtsablauf** gliedert sich in folgende Phasen:

1. **Echtzeit-Datenverarbeitung im Alltag:** Als Kontext werden verschiedene Beispiele diskutiert, bei denen Echtzeitdatenverarbeitung benötigt wird, z. B. die Analyse von Sensordaten (z. B. Tsunamierkennung), von Kreditkartentransaktionen (z. B. Ablehnung verdächtiger Kreditkartentransaktionen) oder von Social-Media-Daten (z. B. Beurteilung des Erfolgs eines Produkts).
2. **Grenzen der Datenverarbeitung mit Datenbanksystemen:** Am Beispiel der Analyse von Beiträgen in sozialen Medien anhand bestimmter Stichwörter (z. B. Produktnamen, Marken) werden die Grenzen beim Einsatz einer (relationalen) Datenbank, insbesondere hinsichtlich der Komplexität des Datenmodells und des Speicherplatzbedarfs, diskutiert.
3. **Skalierung und Datenstromsysteme:** Vor diesem Hintergrund kann ausgelotet werden, wie diese Grenzen überwunden werden können: Neben den Grenzen der vertikalen Skalierung kann horizontale Skalierung als Möglichkeit – die aber die Echtzeitfähigkeit einschränkt – diskutiert werden. Als innovativer Ansatz wird die direkte Verarbeitung des Datenstroms mit Datenstromsystemen vorgestellt.
4. **Kennenlernen von Datenstromsystemen:** Hierzu kann ein einfaches Datenstromsystem herangezogen werden. Für diesen Zweck entwickeln wir derzeit eine Anwendung, die es erlaubt, den Datenstrom von Twitter in Echtzeit mit durch den Nutzer (in gewissem Maße) anpassbaren Abfragen zu analysieren. Bei der eigenen Implementierung einer solchen Anwendung können die Lernenden die Prinzipien und Konzepte von Datenstromsystemen selbst erkennen und umsetzen.
5. **Grenzen von Datenstromsystemen:** Beim „Versuch“ nachträglich Abfragen zu bearbeiten, erkennen die Schüler die Grenzen eines Datenstromsystems. Sie erkennen, dass die Nutzung von Datenstromsystemen kaum möglich ist, wenn die Kriterien zur

Formulierung einer Abfrage vorher nicht genau bekannt sind. Somit sind mit Datenstromsystemen z. B. Ausreißeranalysen nur möglich, wenn ein üblicher Wertebereich definiert werden kann. In Datenbanken hingegen kann der übliche Bereich und Ausreißer aus diesem im Nachhinein statistisch analysiert werden. Je nach Anwendungszweck (z. B. Tsunamivorhersage) ist jedoch die Analyseverzögerung von Nachteil.

Weder DBMS noch DSS können in allen Anwendungsfällen gleichermaßen eingesetzt werden. Sie stellen unterschiedliche Herangehensweisen an die Datenverarbeitung dar, an denen gut erkannt werden kann, dass für verschiedene Anwendungszwecke verschiedene Lösungen nötig sind. Gerade im Zusammenhang mit immer größeren Hauptspeichermengen gewinnen Datenstromsysteme an Bedeutung: Die flüchtige Speicherung und schnelle Verarbeitung erlaubt zeitnah zur Verfügung stehende Resultate, die jedoch durch genauere (da auf einem größeren Datenbestand basierende), aber langsamer erzeugte Ergebnisse, aus datenbankbasierten Analysen unterstützt werden können.

4 Diskussion

Die beiden beschriebenen Beispiele greifen zwei wesentliche Aspekte von Big Data auf, die auch im Alltag immer stärker an Bedeutung gewinnen, und zeigen an ihnen innovative Vorgehensweisen der Informatik auf: Durch die immer größer werdenden Datenmengen werden einerseits immer umfangreichere Analysen der Daten möglich, andererseits werden Daten auch immer schneller verarbeitet, idealerweise in Echtzeit. Während heute die Gefahren von und kritische Sichtweisen auf die Erhebung großer Datenmengen ausführlich diskutiert werden, u. a. auch im Zusammenhang mit dem NSA-Skandal, nutzt gleichzeitig jeder die vielfältigen Möglichkeiten moderner Datenverwaltung und generiert dabei umfangreiche Datenmengen. Dabei kommt auch dem dargestellten Data Mining und der Echtzeitanalyse, möglicherweise unter Nutzung von Datenstromsystemen, eine wesentliche Bedeutung zu. Ein grundlegendes Verständnis der Grundlagen, Herangehensweisen und Ideen dieser Systeme wird dabei immer wichtiger, da sie heute allgegenwärtig sind. Insbesondere ist auch die Teilnahme am gesellschaftlichen Diskurs und die Einschätzung solcher Anwendungen und Dienste nur möglich, wenn eine entsprechende Grundlage durch den Informatikunterricht gelegt wurde. Durch die zuvor dargestellten Beispiele konnte exemplarisch gezeigt werden, dass Big Data Aspekte beinhaltet, die für den Informatikunterricht auf einem geeigneten Niveau didaktisch reduziert unterrichtet werden können, und somit entgegen des ersten Eindrucks nicht grundsätzlich zu komplex für den Schulunterricht ist. In beiden Beispielen können dabei Aspekte erkannt werden, die als potentielle fundamentale Ideen [Sc93] gehandelt werden können: Neben den Datenanalysemethoden sind unter anderem auch das Selektieren geeigneter Daten, der Umgang mit Schnittstellen/APIs, das Erkennen von Möglichkeiten und Grenzen von Informatiksystemen, Parallelisierung bzw. Aufsplittung von Daten zur effizienteren Verarbeitung, die Betrachtung von Stichproben (Sampling), u. v. m. in diesen Beispielen enthalten. Diese Ideen können nur in einem Informatikunterricht vermittelt werden, der sich zwar auf die Grundlagen konzentriert, dabei aber moderne Entwicklungen und Beispiele miteinbezieht. Durch die Beschäftigung mit Themen wie moderner Datenverarbeitung und Datenmangement bzw. Big Data, kann der Informatikunterricht wesentlich zu einer überlegten Positionierung im Spannungsfeld, das sich durch diese neuen Möglichkeiten ergibt, beitragen: Jeder muss heute zwischen dem

Schutz der eigenen Privatsphäre durch Vermeidung der Weitergabe von Daten und dem Komfort, den viele moderne Systeme bieten, abwägen – eine Herausforderung, die ohne entsprechendes (informatisches) Hintergrundwissen kaum bewältigt werden kann.

Literaturverzeichnis

[Br12] Brewer, Eric: CAP twelve years later: How the "rules" have changed. Computer, 45(2):23–29, 2012.

[Bu14] Burns, Will: Taxi Stockholm Shows Us That Big Data Needs A Big Idea. 2014. http://onforb.es/1shlLLL, abgerufen: 20.04.2015.

[DA09] DAMA International: The Dama Guide to the Data Management Body of Knowledge . Take It With You. Technics Publications Llc, 2009.

[DN13] Dorschel, Joachim; Nauerth, Philipp: Big Data und Datenschutz – ein Überblick über die rechtlichen Herausforderungen. Wirtschaftsinformatik & Management, (2):32–38, 2013.

[Ed11] Edlich, Stefan; Friedland, Achim; Hampe, Jens; Brauer, Benjamin; Brückner, Markus: NoSQL. Hanser, Carl Gmbh + Co., 2011.

[ER13] Eifrem, Emil; Rathle, Philip: Why the most important part of Facebook Graph Search is "Graph". 2013. http://neo4j.com/?p=81, abgerufen: 20.04.2015.

[FU14] FUJITSU Technology Solutions GmbH: White Paper Lösungsansätze für Big Data. 2014. http://globalsp.ts.fujitsu.com/dmsp/Publications/public/wp-bigdata-solution-approaches-de.pdf, abgerufen: 20.04.2015.

[GR14] Grillenberger, Andreas; Romeike, Ralf: A Comparison of the Field Data Management and its Representation in Secondary CS Curricula. In: Proceedings of WiPSCE 2014. ACM, Berlin, 2014.

[He14] Heuzeroth, Thomas: Amazon verschickt Waren schneller, als Sie kaufen. Die Welt, 2014. http://www.welt.de/wirtschaft/webwelt/article123990975, abgerufen: 20.04.2015.

[HNP09] Halevy, Alon; Norvig, Peter; Pereira, Fernando: The unreasonable effectiveness of data. Intelligent Systems, IEEE, 24(2):8–12, 2009.

[KE13] Kemper, Alfons; Eickler, André: Datenbanksysteme. Oldenbourg Wissensch.Vlg, 2013.

[Kr13] Krikorian, Raffi: New Tweets per second record, and how! 2013. https://blog.twitter.com/node/2845, abgerufen: 20.04.2015.

[La01] Laney, Douglas: 3D Data Management: Controlling Data Volume, Velocity, and Variety. Bericht, META Group, 2001.

[MC13] Mayer-Schönberger, Viktor; Cukier, Kenneth: Big Data - Die Revolution, die unser Leben verändern wird. FinanzBuch Verlag, München, 2013.

[Sc93] Schwill, Andreas: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik, 1993.

[Wo11] World Economic Forum: Personal Data: The Emergence of a New Asset Class. 2011. http://www3.weforum.org/docs/WEF_ITTC_PersonalDataNewAsset_Report_2011.pdf, abgerufen: 20.04.2015.

[Wo14] World Economic Forum: Rethink Personal Data: Trust and Context in User-Centred Data Ecosystems. 2014. http://www3.weforum.org/docs/WEF_RethinkingPersonalData_TrustandContext_Report_2014.pdf, abgerufen: 20.04.2015.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 135

Big-Data-Analyse im Informatikunterricht mit Datenstromsystemen: Ein Unterrichtsbeispiel

Andreas Grillenberger und Ralf Romeike[1]

Abstract: Big Data stellt heute ein zentrales Thema der Informatik dar: Insbesondere durch die zunehmende Datafizierung unserer Umwelt entstehen neue und umfangreiche Datenquellen, während sich gleichzeitig die Verarbeitungsgeschwindigkeit von Daten wesentlich erhöht und diese Quellen somit immer häufiger in nahezu Echtzeit analysiert werden können. Neben der Bedeutung in der Informatik nimmt jedoch auch die Relevanz von Daten im täglichen Leben zu: Immer mehr Informationen sind das Ergebnis von Datenanalysen und immer häufiger werden Entscheidungen basierend auf Analyseergebnissen getroffen. Trotz der Relevanz von Daten und Datenverarbeitung im Alltag werden moderne Formen der Datenanalyse im Informatikunterricht bisher jedoch allenfalls am Rand betrachtet, sodass die Schülerinnen und Schüler weder die Möglichkeiten noch die Gefahren dieser Methoden erfahren können.

In diesem Beitrag stellen wir daher ein prototypisches Unterrichtskonzept zum Thema Datenanalyse im Kontext von Big Data vor, in dem die Schülerinnen und Schüler wesentliche Grundlagen von Datenanalysen kennenlernen und nachvollziehen können. Um diese komplexen Systeme für den Informatikunterricht möglichst einfach zugänglich zu machen und mit realen Daten arbeiten zu können, wird dabei ein selbst implementiertes Datenstromsystem zur Verarbeitung des Datenstroms von Twitter eingesetzt.

Keywords: Big Data, Datenanalyse, Data Mining, Assoziation, Clusterbildung, Klassifikation, Datenstromsysteme, Echtzeitverarbeitung, Snap!, Twitter, Unterrichtsbeispiel

1 Einleitung

Die Bedeutung der Verarbeitung von Daten nimmt im Kontext von *Big Data* rapide zu: Nicht nur in der Informatik kommt dem Fachgebiet *Datenmanagement*, das sich durch die Innovationen im Kontext von Big Data aus dem Gebiet *Datenbanken* entwickelt hat, eine steigende Bedeutung zu, sondern auch in Wirtschaft, Politik und Gesellschaft. Gleichzeitig zeichnet sich der Stellenwert dieser Entwicklungen beispielsweise auch dadurch ab, dass in diesem Zusammenhang eine völlig neue Berufsgruppe entsteht: der Datenwissenschaftler ("*Data Scientist*") [DP12]. Neben diesen Einflüssen halten Daten und Datenanalysen jedoch auch immer häufiger Einzug in den Alltag, oft ohne dass dies bewusst wahrgenommen wird: Kreditkartenanbieter analysieren ständig Transaktionen auf mögliche Betrugsfälle, Sensordaten werden ausgewertet um Extremwetterereignisse schnell erkennen zu können und im Produktmarketing werden Echtzeitanalysen eingesetzt, um bei der Einführung eines neuen Produkts einem gegebenenfalls auftretenden unerwünschten Image

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, Martensstr. 3, 91058 Erlangen
andreas.grillenberger@fau.de, ralf.romeike@fau.de

möglichst bald entgegenwirken zu können. Heute begegnet daher jeder andauernd den Ergebnissen verschiedener Datenanalysen. Andererseits werden jedoch auch immer größere Datenmengen produziert, nicht nur im Bereich von Wissenschaft, Wirtschaft und Politik, sondern wiederum auch im Alltag: Mobile Geräte wie Smartphones erzeugen unaufhörlich große Datenmengen, beispielsweise durch Übermittlung ihres Standortes (nicht nur aus GPS- sondern auch aus WLAN- und Funkzellendaten) oder indem jedes mit dem Gerät aufgenommene Foto u. a. mit GPS-Daten versehen wird. Durch aktuelle Innovationen wie das *Internet der Dinge* („Internet of Things", IoT), werden voraussichtlich bald noch reichhaltigere Datenquellen aus vielfältigeren Bereichen des täglichen Lebens zur Verfügung stehen: In der Vision des IoT werden jegliche Geräte und Gegenstände mit denen der Mensch interagiert mit Sensoren und eingebetteten Systemen versehen und erhalten somit eine eigene digitale Identität [Br09]. Durch die tiefe Integration solcher Systeme in das tägliche Leben – die Vision des IoT sieht vor, dass diese Geräte nahezu unbemerkt den Alltag unterstützen – entstehen sowohl Gefahren, beispielsweise für die eigene Privatsphäre, aber auch vielfältige Möglichkeiten mit diesen Daten umzugehen, sie für eigene Zwecke zu nutzen und somit das eigene Leben zu unterstützen.

Obwohl sich das IoT und ähnliche Entwicklungen noch in den Grundzügen befinden, kann heute schon erkannt werden, dass eine immer stärkere *Datafizierung*[3] des gesamten Lebens stattfindet. Gleichzeitig fehlen jedoch einem Großteil der Menschen selbst grundlegende Kenntnisse darüber, welche Möglichkeiten und Gefahren sich insbesondere durch die zunehmenden Möglichkeiten zur schnellen Analyse selbst großer Datenmengen eröffnen, und somit auch wesentliche Kompetenzen, die zum Umgang mit (eigenen und fremden) Daten nötig sind. Obwohl derartige komplexe Entwicklungen sicherlich nicht vollumfänglich im Informatikunterricht abgebildet werden können und sollen, liegt jedoch trotzdem die Vermutung nahe, dass dabei grundlegende Aspekte im Sinne der fundamentalen Ideen nach Schwill [Sc93] enthalten sind (vgl. auch [GR15b]). Die Bedeutung und Tragweite von Big Data und der zugrundeliegenden Konzepte wurden auch von Berendt et. al. [Be14] in einer von ihnen beschriebenen Unterrichtsreihe ausführlich dargestellt. Der dabei genutzte Ansatz, im Unterricht eigene Datenanalysen – bei Berendt et. al. insbesondere Assoziationsanalysen – durchzuführen, scheint dabei für das Verständnis vielversprechend zu sein. Im vorgeschlagenen Unterrichtskonzept wurde dazu der Apriori-Algorithmus diskutiert, was sich jedoch in der Erprobung als problematisch erwiesen hat: „Weniger erfolgreich verlief ein für das Reihenziel bzw. dessen Verständnis unabdingbarer Baustein: der Apriori-Algorithmus" [Be14]. Dabei liegt die Vermutung nahe, dass der Algorithmus aufgrund seiner Komplexität zu Verständnisschwierigkeiten geführt hat. In diesem Beitrag stellen wir daher eine Unterrichtsidee vor, die am Beispiel der grundlegenden Datenanalysemethoden *Assoziation*, *Clusterbildung* und *Klassifikation* zeigt, wie die Grundlagen von aktuellen Innovationen der Informatik für den Informatikunterricht zugänglich gemacht werden können, indem eine Beschränkung auf die grundlegenden und fundamentalen Aspekte dieser Entwicklungen stattfindet, aber trotzdem der innovative Charakter gewahrt wird. Diese Unterrichtsidee soll insbesondere Lehrerinnen und Lehrern als Anregung und Einstieg in das komplexe Thema *Big Data* dienen und stellt daher kein detailliert ausgearbeitetes und erprobtes Unterrichtsbeispiel dar. Zur Umsetzung der dargestellten Ideen wird

[3] Datafizierung bezeichnet die zunehmende Erfassung vielfältiger Aspekte der Realität in Form von Daten.

zusätzlich ein Werkzeug vorgestellt, mit dem Schülerinnen und Schülern durch eine Erweiterung der universellen Programmierumgebung Snap! [HM14] eigene Datenstromanalysen durchführen und dafür auf den Datenstrom des sozialen Netzwerkes Twitter zugreifen können. Bevor dieses Werkzeug in Kapitel 3 dieses Beitrags vorgestellt wird, skizzieren wir in Kapitel 2 die fachlichen Grundlagen der aktuellen Entwicklungen im Datenmanagement und von Datenstromsystemen. Auf dieser Basis stellen wir in Kapitel 4 die konkrete Unterrichtsidee vor.

2 Unterrichtsgegenstand: Big Data und Datenanalysen

Allen aktuellen Innovationen auf dem Gebiet Datenmanagement liegt die Verarbeitung immer größerer Mengen verschiedenartiger Daten in kurzer Zeit zu Grunde: *Big Data*. Dieser Überbegriff für die aktuellen Entwicklungen wird durch die sog. drei „V"s charakterisiert: große Datenmengen (*volume*), schnelle Datenerzeugung und -verarbeitung (*velocity*) sowie unterschiedliche Strukturierung oder gar Unstrukturiertheit dieser Daten (*variety*). Bei der Verarbeitung von Big Data ist die Gewinnung von neuen Informationen und Erkenntnissen aus oft schon vorhandenen umfangreichen Datensätzen ein wesentliches Ziel: Im Kontext von *Data Mining*, dem „Datenbergbau"[4], werden Daten oft als das Gold des 21. Jahrhunderts bezeichnet, in anderen Kontexten auch als das neue Öl. Die Innovationen im Datenmanagement sind dabei vielfältig einsetzbar und eröffnen verschiedene neue Möglichkeiten, stellen aber gleichzeitig den Datenschutz vor neue Herausforderungen und verursachen neue Gefahren für die Privatsphäre. In diesem Zusammenhang bezeichnet Dittrich [Di13] Daten auch als das neue Uran: Natürlich können Daten sinnvoll und gewinnbringend eingesetzt werden, sie sind aber auch kaum löschbar, können in falsche Hände gelangen, angereichert werden und vieles mehr.

Während zur Speicherung und Verarbeitung von Daten bisher insbesondere Datenbanken zum Einsatz kamen, gewinnen im Kontext von Big Data weitere Ansätze zur Datenverarbeitung deutlich an Bedeutung, wie beispielsweise *Datenstromsysteme* (DSS): Im Gegensatz zu Datenbanken werden die Daten in DSS nicht dauerhaft gespeichert, sondern sofort verarbeitet, wodurch eine wesentlich höhere Geschwindigkeit der Datenverarbeitung ermöglicht wird. Durch die sofortige Verarbeitung kann auf geänderte Daten (beispielsweise Sensor-Messwerte) schnell reagiert werden, wie es z. B. bei der Messung von seismischen Wellen in einem System zur Tsunamierkennung und -vorwarnung nötig ist. Diese Systeme betonen daher insbesondere den Aspekt der *velocity* von *Big Data*. Einen wichtigen Einsatzzweck von DSS stellt daher die Überwachung von Datenquellen („*Monitoring*") dar: Gerade im Kontext des Internets der Dinge und auch bei der Heimautomation ist dieses Prinzip allgegenwärtig, beispielsweise indem Beleuchtung und Heizung eines Gebäude automatisch in Abhängigkeit davon geregelt werden, ob Personen anwesend sind oder Jalousien sich automatisch dem Sonnenstand anpassen.

Eine wichtige Basis stellen in diesem Zusammenhang die grundlegenden Datenanalysemethoden dar. An diesen zeigt sich auch die Möglichkeit, die Grundlagen von komplexen

[4] Obwohl sich der Begriff *Data Mining* durchgesetzt hat, sollte dabei möglicherweise eher vom Informationsbergbau gesprochen werden, da es dabei um die Gewinnung von Informationen, nicht von Daten, geht.

und im Ganzen schwer erfassbaren Datenanalysen für den Unterricht didaktisch reduziert greifbar zu machen. In der in diesem Artikel vorgestellten Unterrichtsidee werden wir uns insbesondere auf die drei wichtigsten Datenanalysemethoden [ES00] „Klassifikation“, „Clusterbildung“ und „Assoziation“ beschränken:

- **Clusterbildung** dient dazu, Gemeinsamkeiten zwischen verschiedenen Daten zu analysieren und Daten nach diesen Merkmalen zusammenzufassen. Für die automatisierte Clusterbildung werden bekannte Verfahren wie beispielsweise der „k-Means-Algorithmus“ eingesetzt.
- **Klassifikation** bezeichnet das Einsortieren von Daten in vordefinierte Klassen anhand vorgegebener Merkmale. Von der Clusterbildung unterscheidet sich diese Methode daher dadurch, dass vordefinierte Klassen verwendet werden anstatt unbekannte zu finden. Zur Klassifikation von Daten werden beispielsweise Bayes-Klassifikatoren oder Entscheidungsbäume eingesetzt.
- **Assoziationen** werden genutzt um Zusammenhänge zwischen verschiedenen Merkmalen eines Datensatzes auszudrücken. Diese typischerweise in der Form „Wenn ... dann ...“ formulierten Zusammenhänge werden häufig eingesetzt, um aus bekannten Merkmalen unbekannte vorherzusagen. Aus der der Assoziationsanalyse stammt auch der von Berendt et. al. [Be14] eingesetzte Apriori-Algorithmus.

Im Unterricht können solche Methoden und die damit einhergehenden Möglichkeiten nur unter Nutzung umfangreicher Datenquellen erprobt werden. Solche Datenquellen stehen heute jedoch aus verschiedenen Bereichen zur Verfügung: Für relativ statische Datenanalysen, beispielsweise mit Datenbanken, können Datensätze herangezogen werden, die im Rahmen von *Open-Data*-Initiativen[5] zunehmend veröffentlicht werden. Gleichzeitig stellen insbesondere soziale Netzwerke einen umfangreichen und vielfältig einsetzbaren Strom von Daten bereit, der gut für dynamische Analysen, beispielsweise im Kontext von DSS, genutzt werden kann, durch den starken Bezug zur Lebenswelt aber meist auch einen motivierenden Charakter aufweist. Daher werden wir uns in diesem Unterrichtsbeispiel auf die Analyse des Twitter-Datenstroms mit Hilfe eines einfachen Datenstromsystems konzentrieren.

3 Datenstromanalyse mit Snap!

In diesem Kapitel werden wir kurz das angesprochene und im folgenden Unterrichtskonzept genutzte Werkzeug zur Analyse des Twitter-Datenstroms mit Snap! beschreiben[6]. Obwohl es bereits verschiedene Datenstromsysteme gibt, war für den Unterricht eine Eigenentwicklung auf diesem Gebiet unverzichtbar, da bekannte Systeme auf diesem Gebiet für den Unterricht kaum geeignet sind: Insbesondere ist durch die umfangreichen Möglichkeiten, die Einstiegshürde hoch, während zugleich die zugrundeliegenden Prinzipien nur

[5] Open Data bezeichnet Daten, die der Allgemeinheit zur freien Nutzung und Weiterverbreitung zur Verfügung gestellt werden. Sie stammen oft aus der öffentlichen Verwaltung (vgl. z. B. http://www.govdata.de).

[6] Eine detailliertere Beschreibung kann [GR15a] entnommen werden.

schwer erkennbar sind. Hinzu kommen die derzeitigen Entwicklungen auf diesem Gebiet, die die Auswahl eines Systems deutlich erschweren: Im Gegensatz zu Datenbanken gibt es bei DSS bisher keine universelle Anfragesprache, sodass das erworbene Sprachwissen und die Kenntnisse über die Bedienung sehr anwendungsspezifisch wären. Diese Nachteile können durch die Erweiterung der Programmierumgebung Snap! vermieden werden: Den Schülern ist häufig dieses oder ein ähnliches Werkzeug schon bekannt. Durch die blockorientierte Programmierung wird die Einstiegshürde gesenkt und zugleich wird vermieden, sich auf eine bestimmte Anfragesprache zu spezialisieren, indem die Anfragen auf höherer Abstraktionsebene betrachtet werden.

Das implementierte Werkzeug basiert dabei auf zwei Bausteinen: Einerseits musste Snap! um entsprechende Blöcke für die Datenstromanalyse erweitert werden, andererseits musste eine Anbindung an den zu analysierenden Datenstrom geschaffen werden: Eine direkte Anbindung von Snap! an die von Twitter zur Verfügung gestellte API ist aufgrund von Sicherheitsmaßnahmen zur Vermeidung von Cross–Site–Scripting–Attacken in allen üblichen Browsern nicht möglich. Um die Anbindung dennoch zu ermöglichen, musste daher ein Programm implementiert werden, das als Proxy zwischen der Twitter-API und Snap! dient und den Zugriff durch Snap! explizit zulässt. Um die im Folgenden beschriebenen Blöcke in Snap! nutzen zu können, muss daher diese Hilfsanwendung im Hintergrund laufen – es reicht jedoch aus, die Anwendung auf einem PC pro Klassenraum zu starten und Snap! so zu konfigurieren, dass die Daten von diesem PC abgefragt werden.

Zur Erweiterung von Snap! um die für Datenstromanalysen benötigten Blöcke, wurden als Basis die 

und 

Blöcke herangezogen. Die Snap!-Erweiterung ist damit auch auf der offiziellen Snap!-Installation lauffähig, da nur Snap!-eigene Funktionalitäten verwendet werden. Um Daten aus dem Twitter-Datenstrom einzulesen, wurde der Block 

implementiert, der jeweils einen Tweet von der Hilfsanwendung einliest. Innerhalb von Snap! wird das gesamte Tweet-Objekt als JSON-formatierter[7] Text gespeichert. Zum Zugriff auf die einzelnen Attribute ist daher eine weitere Funktion nötig, die den JSON-Text interpretiert und das gewünschte Attribut ausliest. Diese Funktion wird durch den Block 

repräsentiert. Zur ansprechenden Darstellung der Analyseergebnisse wurde zusätzlich die Möglichkeit einer Kartendarstellung bzw. einer Darstellung als Balkendiagramm mit den Blöccken 

bzw. 

implementiert. Von einigen dieser Blöcke, beispielsweise dem Block zum Auslesen von Attributen eines Tweets, wurden weitere Varianten implementiert, beispielsweise 

zum Auslesen der Geokoordinaten.

4 Unterrichtsidee

Im Alltag können Schülerinnen und Schüler Datenanalysen und deren Auswirkungen heute an vielfältigen Stellen begegnen, beispielsweise bei der Verwendung von sozialen Medien oder beim Einkauf im Online-Versandhandel. Um die Möglichkeiten, das Potential und auch die Risiken solcher Analysen erkennen und verstehen zu können, zielt das im

[7] JSON bezeichnet die JavaScript Object Notation, ein gebräuchliches und einfach lesbares Format zum Austausch strukturierter Daten.

Folgenden vorgestellte Unterrichtskonzept darauf ab, die drei grundlegenden Datenanalysemethoden „Klassifikation“, „Clusterbildung“ und „Assoziation“ didaktisch reduziert zu vermitteln. Um einen handlungsorientierten Unterricht zu diesen Themen zu ermöglichen, in dem die Lernenden von Anfang an die Möglichkeit zur Nutzung realer Daten sowie zur interaktiven Entwicklung und Anpassung ihrer Datenanalysen haben, wird das zuvor beschriebene Snap!-basierte Werkzeug eingesetzt, das es den Schülerinnen und Schülern ermöglicht, selbst eigene Datenstromanalysen durchzuführen.

4.1 Randbedingungen und Einsatzkontext

Die im Folgenden beschriebene Unterrichtsidee orientiert sich an keinem speziellen Lehrplan, sondern stellt einen universellen Ansatz dar, Datenanalysemethoden didaktisch reduziert im Unterricht einzusetzen. Damit können Elemente dieser Sequenz in unterschiedlicher Tiefe und Breite in verschiedenen Kontexten eingesetzt werden, wie beispielsweise den Möglichkeiten von Datenanalysen, zu Datenschutz und Privatsphäre oder als Motivation für die Speicherung großer Datenmengen, eingesetzt werden. Das dazu nötige Vorwissen beschränkt sich auf die grundlegende Unterscheidung von Informationen und Daten sowie idealerweise grundlegende Kenntnisse bzw. Erfahrungen mit einer blockbasierten Programmiersprache. Kenntnisse in der Datenanalyse, zu Datenstromsystemen, Datenbanken oder ähnlichem werden dabei zwar nicht zwingend benötigt, können aber förderlich eingebunden werden.

4.2 Unterrichtsverlauf

4.2.1 Einführung

Zur Schaffung eines lebensweltlichen Bezugs wird in diesem Unterrichtsbeispiel exemplarisch der Einsatz von Personalisierungsmöglichkeiten im Online-Versandhandel herangezogen. Jedem Schüler, der bereits öfter im Internet eingekauft hat, ist sicherlich aufgefallen, dass sich die Portale der Online-Versandhändler an die eigenen Einkaufsgewohnheiten anpassen, insbesondere in Bezug auf empfohlene Produkte. Während schon für diese Anpassungen grundlegende Datenanalysen genutzt werden, wird es jedoch wesentlich spannender, wenn miteinbezogen wird, dass solche Empfehlungen oft nicht nur mit einem Benutzerkonto, sondern auch mit Merkmalen wie IP-Adresse, Region oder Browser-Identifikation verknüpft werden können und somit auch ohne Benutzerkonto wertvolle Informationen über den Nutzer zur Verfügung stehen oder ermittelt werden können. Das Wissen über den Wohnort der Besucher kann im Zeitalter von Big Data nicht mehr nur eingesetzt werden um regionale Produkte anzubieten, sondern beispielsweise auch um vorherzusagen, wie viel Geld der Kunde zur Verfügung hat oder welchen Beruf er ausübt – wenn auch nur mit einer bestimmten Wahrscheinlichkeit. Dass diese Wahrscheinlichkeitsanalyse für viele Zwecke ausreicht zeigt ein Patent, dass Amazon 2014 zugesprochen wurde: Der Versandhändler plant, Produkte schon in ein naheliegendes Versandzentrum zu senden wenn ein Kunde Interesse daran zeigt, aber noch vor einer konkreten Bestellung

[Sp14]. Um verschiedene Zusammenhänge aufzudecken, reichen einzelne Datensätze oft nicht aus: Weitere Informationsquellen müssen herangezogen werden. Als ideale Datenquellen für Marketingzwecke dienen dabei oft soziale Medien wie Twitter oder Facebook: Durch die große Menge an zur Verfügung stehenden Daten können gute Resultate relativ einfach gewonnen werden.

Ziel des im Folgenden vorgestellten Unterrichts ist es daher, dass die Schülerinnen und Schüler die Gewinnung solcher Zusammenhänge nachvollziehen, solche Zusammenhänge an einfachen Beispielen selbst gewinnen und auch kritisch hinterfragen können.

4.2.2 Datenanalysemethoden

Nach einer kurzen Einführung in das Thema Datenanalysen können die Schülerinnen und Schüler mit Hilfe des zuvor erwähnten Werkzeugs einfache Aanalysen selbst durchgeführen und die Funktionsweise der zugrundeliegenden Methoden am Beispiel nachvollziehen.

Klassifikation Anhand einfacher Klassifikationsaufgaben können die Lernenden das zugrunde liegende Prinzip erkennen: die Einteilung von vorliegenden Daten in feste und zuvor definierte Kategorien. Beispielsweise kann dafür die Aufgabe gestellt werden, mit dem zur Verfügung gestellten Tool alle eingehenden Tweets anhand bestimmter Stichworte oder der Sprache in der sie verfasst wurden zu klassifizieren. Unter Nutzung des angesprochenen Werkzeugs kann als erstes Resultat beispielsweise ein Diagramm, wie in Abbildung 1 dargestellt, gewonnen werden.

Abb. 1: Klassifikation der Tweets nach Sprache, dargestellt als Balkendiagramm in Snap!

Bei der Diskussion der möglichen Aussagen, die aus der Klassifikation gewonnen werden können, stellen die Schülerinnen und Schüler die Grenzen dieser Methode fest: Beispielsweise könnte problemlos analysiert werden, welches Produkt beliebter ist als ein anderes, woher die meisten Käufer kommen, und ähnliches. Die zuvor beschriebenen Schlussfolgerungen, die heute aus Daten gewonnen werden können, gehen darüber weit hinaus: Es geht nicht nur darum zu entdecken, welches Produkt am beliebtesten ist, sondern welches Produkt in welcher Region bevorzugt wird – es wird eine zweite Dimension eingeführt. Eine Kategorisierung nach mehreren Dimensionen wäre zwar möglich, die Anzahl der Kategorien explodiert dabei jedoch, da jede Kategorie mit jeder anderen kombiniert werden kann. Zusätzlich können nicht in allen Fällen klare Kategorien schon vor der Analyse festgelegt werden, z. B. muss die Region hier nicht zwingend administrativen Regionen entsprechen, sodass in diesem Fall die Klassifikation an ihre Grenzen gerät.

Clusterbildung An dieser Stelle setzt die Clusterbildung an: Ähnliche Daten werden zusammengefasst und als Gruppe betrachtet. Trotz komplexer mathematischer Grundlagen kann eine einfache Clusteranalyse bereits mit dem zur Verfügung gestellten Werkzeug bewältigt werden: Nach der Visualisierung einer bestimmten Eigenschaft auf der Karte, können Cluster intuitiv bestimmt und so erkannt werden, welche der Ausprägungen dieser Eigenschaft in verschiedenen Regionen vorherrscht. Durch die intuitive Herangehensweise kann die Clusterbildung somit nachvollzogen werden, ohne die mathematischen Grundlagen betrachten zu müssen. In Abbildung 2 wird dieser Vorgang beispielhaft dargestellt.

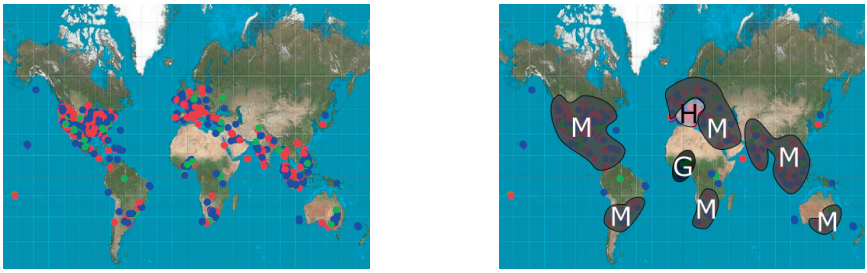


Abb. 2: Visualisierung aller Tweets auf einer Karte in Snap!. Die Farbe der Punkte entspricht der Follower-Zahl (rot: über 500, blau: über 100, grün: unter 100). Die rechte Grafik stellt beispielhafte Cluster mit **g**eringer, **m**ittlerer und **h**oher Followerzahl dar. Karte ©2011 Strebe, CC BY–SA 3.0

Dieses einfache Beispiel zeigt einige der grundlegenden Fragestellungen bei der Verwendung von Clusterbildung: Wie groß sollen die erzeugten Cluster sein? Ab wann wird eine Ausprägung einer Eigenschaft als vorherrschend charakterisiert? Welche Fehler bin ich bereit in Kauf zu nehmen?

Assoziation Gerade wenn Vorhersagen anhand geclusterter Daten getroffen werden sollen, spielen diese Fragen eine entscheidende Rolle: Je größer die Cluster, umso größer ist bei heterogenen Daten der Fehler, der eingegangen werden muss, um diese als ein Cluster betrachten zu können. Dieser Fehler wirkt sich auch auf die Assoziationsanalyse aus, da möglicherweise falsche oder nur teilweise zutreffende Assoziationen gebildet werden. In diesem Kontext sollten daher verschiedene mögliche Assoziationen, die sich aus den obigen Betrachtungen ergeben, mit den Schülern auf ihre Aussagekraft hin untersucht werden: Mit Blick auf die in Abbildung 2 dargestellte Karte scheint die Assoziation „Wer Twitter in Westeuropa, Südostasien oder den USA nutzt, hat mindestens 100 Follower“ mit relativ geringem Fehler zutreffend zu sein. Diese Aussagen vernachlässigen aber jegliche Kausalität: Es wird nicht weiter überlegt, warum beispielsweise in Afrika, Australien oder Russland Twitter anscheinend kaum genutzt wird. Da es sich bei der Auswertung jedoch um eine nicht-repräsentative Stichprobe handelt (es wurden nur Tweets betrachtet, die Geodaten offenbaren), kann kein Rückschluss auf alle Twitter-Nutzer getroffen werden. Möglicherweise hängt die Bereitschaft die eigenen Geodaten zu offenbaren mit der eigenen Aktivität – und damit möglicherweise auch der Follower-Anzahl – zusammen, sodass die getroffene Schlussfolgerung durch die Stichprobenwahl beeinflusst wurde. Diese Grenzen der Aussagekraft von Datenanalysen muss man sich daher unbedingt bewusst machen, um falsche Schlüsse und somit auch das Vorurteil der Allwissenheit und Unfehl-

barkeit von Big Data zu vermeiden, aber auch um die Bedeutung eines möglichst vollständigen Datensatzes bei Big-Data-Analysen nachvollziehen zu können.

4.2.3 Möglichkeiten und Risiken von Datenanalysen

Die obigen Beispiele zeigen deutlich, wie aus einfachen und bedeutungslos wirkenden Daten interessante und je nach Einsatzgebiet sehr wertvolle Daten gewonnen werden können. Übertragen auf den Online-Versandhandel zeigt sich damit die Möglichkeit, eine starke Personalisierung von Produktempfehlungen, Werbung (eigener sowie fremder) etc. durchzuführen, was für beide Seiten – Kunde und Betreiber – wünschenswert ist: Einerseits wird dem Kunden das Auffinden interessanter Produkte erleichtert, andererseits steigert sich der Absatz des Händlers. Neben diesen Möglichkeiten die durch die Prinzipien der Datenanalyse eröffnet werden, müssen jedoch auch immer die damit einhergehenden Gefahren betrachtet werden, um auf diese geeignet reagieren und angemessen damit umgehen zu können. Dieselben Onlinehändler, die derartige Datenanalysen zum beidseitigen Vorteil von Kunde und Händler einsetzen, können diese Daten beispielsweise auch zur Personalisierung des Preises eines Artikels nutzen: Wer aus einem reichen Umfeld kommt, ist sicherlich eher bereit, hohe Preise zu bezahlen, als jemand aus einer ärmeren Wohngegend. Dies zeigt auch die Gefahr einer Stigmatisierung aufgrund von Big-Data-Analysen: Durch die reine Betrachtung von Korrelationen anstatt von Kausalitäten wird ein wichtiger Aspekt vernachlässigt. Es wird nicht mehr überlegt, warum jemand möglicherweise in einer eher ärmeren Wohngegend lebt, sondern daraus beispielsweise nur noch direkt seine Kreditunwürdigkeit gefolgert. Im schlimmsten Fall geschieht dies, ohne dass derjenige auch nur geringen Einfluss nehmen kann, da möglicherweise keinerlei konkrete Daten über diese Person zur Verfügung stehen, sondern nur statistische Analysen der Entscheidung zugrunde liegen.

5 Fazit

Wie in der Unterrichtsidee dargestellt wurde, können die grundlegenden Datenanalysemethoden geeignet didaktisch reduziert betrachtet werden, um sie in der Schule diskutieren zu können. Damit wird das Potential geschaffen, auf dieser Grundlage auch komplexere Beispiele, wie z. B. den Apriori-Algorithmus verstehen zu können. Die drei Methoden *Klassifikation*, *Clusterbildung* und *Assoziation* stellen dabei nicht nur in der Datenanalyse wichtige Grundlagen dar, sondern können auch in anderen Gebieten der Informatik, aber auch in anderen Wissenschaften und im täglichen Leben sinnvoll eingesetzt werden. Diese Methoden werden beispielsweise im Kontext von Datenbanken, Data Warehouses und Datenstromsystemen verwendet, aber auch in der Mustererkennung oder der angewandten Informatik (z. B. medizinische Informatik). Gerade Klassifikationen und Assoziationen können aber z. B. auch beim objektorientierten Entwurf wiedergefunden werden. Obwohl diese Methoden im Zusammenhang mit Big Data und Data Mining als relativ moderne Entwicklungen erscheinen, betonen sie exemplarisch ein wichtiges Prinzip der Informatik: die Reduzierung von Komplexität, indem gleichartige Objekte zusammengefasst (*klassi-*

fiziert), die Ähnlichkeit von Objekten erkannt (*Clusterbildung*) und Regeln/Zusammenhänge zwischen verschiedenen Objekten und deren Verhaltensweisen aufgedeckt werden (*Assoziation*). Auch im Zusammenhang mit Datenanalysen waren diese Methoden schon lange vor dem Schlagwort *Big Data* etabliert und wurden beispielsweise schon 2000 von Ester & Sander [ES00] in der Art beschrieben, in der sie auch heute eingesetzt werden. Auch in Zukunft ist nicht zu erwarten, dass die Bedeutung dieser Methoden abnehmen wird, im Gegenteil ist aufgrund der beschriebenen Entwicklungen aktuell ein Ansteigen ihrer Relevanz zu beobachten und auch zukünftig weiter zu erwarten.

Durch die Thematisierung dieser Methoden im Informatikunterricht bekommen die Schülerinnen und Schüler außerdem die Möglichkeit, Datenanalysen selbst durchzuführen, was durch das dargestellte Werkzeug unterstützt wird. Die Analyse des Twitter-Datenstroms stellt dabei nur eine der vielfältigen Einsatzmöglichkeiten dar: Mit geringem Aufwand kann Snap! beispielsweise auch so erweitert werden, dass statt des Twitter-Datenstroms RSS-Feeds, andere RDF-formatierte Daten und Ähnliches eingesetzt werden können. Durch die Nutzung weiterer Werkzeuge, beispielsweise von *import.io*, können auch beliebige Webseiteninhalte für die Datenanalyse vorbereitet werden (*Data Scraping*).

Literaturverzeichnis

[Be14] Berendt, Bettina; Dettmer, Gebhard; Demir, Cihan; Peetz, Thomas: Kostenlos ist nicht kostenfrei. LOG IN, (178/179), 2014.

[Br09] Brand, Leif; Hülser, Tim; Grimm, Vera; Zweck, Axel: Internet der Dinge - Perspektiven für die Logistik. 2009. https://www.vdi.de/fileadmin/vdi_de/redakteur/dps_bilder/TZ/2009/Band%2080_IdD_komplett.pdf, zuletzt geprüft: 27.04.2015.

[Di13] Dittrich, Jens: NSA, Überwachung, Big Data und warum Daten wie Uran sind. 2013. https://youtu.be/gwz6u8kqvSo, zuletzt geprüft: 27.04.2015.

[DP12] Davenport, Thomas H.; Patil, D. J.: Data scientist: the sexiest job of the 21st century. Harvard business review, 90(10):70–77, 2012.

[ES00] Ester, Martin; Sander, Jörg: Knowledge Discovery in Databases:. Springer Berlin, 2000.

[GR15a] Grillenberger, Andreas; Romeike, Ralf: Analyzing the Twitter Data Stream Using the Snap! Learning Environment (in Druck). In: Proceedings of ISSEP 2015, Lecture Notes in Computer Science. Springer International Publishing, 2015.

[GR15b] Grillenberger, Andreas; Romeike, Ralf: Big Data im Informatikunterricht: Motivation und Umsetzung. In: INFOS 2015. Lecture Notes in Informatics (LNI). Köllen Druck+Verlag, Bonn, 2015.

[HM14] Harvey, Brian; Mönig, Jens: Snap! Reference Manual. 2014. http://snap.berkeley.edu/SnapManual.pdf, zuletzt geprüft: 27.04.2015.

[Sc93] Schwill, Andreas: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik, 1993.

[Sp14] Spiegel Online: Neues Patent: Amazon will schon vor der Bestellung liefern. 2014. http://www.spiegel.de/article.do?id=944252, zuletzt geprüft: 27.04.2015.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 145

Benutzen – Analysieren – Gestalten – Verankern als didaktische Schrittfolge im Informatikunterricht

Lutz Hellmig[1] und Tino Hempel[2]

Abstract: In den Rahmenplänen des Landes Mecklenburg-Vorpommern findet das didaktische Prinzip Benutzen, Analysieren, Gestalten [**?**] stete Erwähnung, wird jedoch nur in sehr komprimierter Form beschrieben. Mit dem vorliegenden Artikel soll eine Interpretation und Erweiterung durch eine vierte Phase, das Verankern des Wissens, vorgenommen werden. Anhand von Unterrichtsbeispielen wird die Eignung des Prinzips für die Gestaltung eines anregenden Informatikunterrichts illustriert.

1 Einleitung

Informatikunterricht ist wohl wie kaum ein anderes Schulfach durch das Zusammenspiel zwischen der Aneignung konzeptionellen Wissens und dem handelnden Umgang mit Werkzeugen geprägt. Die technische Entwicklung führt mittlerweile dazu, dass das erforderliche Vorwissen für das Interagieren mit Informatiksystemen sinkt und für einige Informatiksysteme schon unterhalb der Nachweisbarkeitsgrenze liegt. Damit verliert auch das klassische und zu häufig praktizierte Instruktionsdesign nach Gagnė [GB74], bei dem erst die Theorie präsentiert werden muss, bevor praktische Übungen unter Nutzung von Informatiksystemen folgen können (Konzepte im Dienste der Werkzeuge), immer deutlicher an Wert. Trotz allem lohnt ein Blick auf Gagnė. Interessanterweise wird der vierte Punkt seiner 9 Events of Instruction im Deutschen häufig als Präsentation des zu erlernenden Wissens verstanden, entsprechend übersetzt und in die pädagogische (Un-)Tat umgesetzt. Im Original ist bei Gagnė aber von „Present stimulus material“ die Rede. Wenn tatsächlich stimulierendes, also eine Handlung anregendes Material angeboten wird, sind die Voraussetzungen für die weitgehend selbständige Erschließung der informatischen Konzepte mithilfe der Nutzung konkreter Artefakte durch die Schüler[3] gegeben. Die Rollen von Unterrichtsziel und -mittel werden getauscht: Der Umgang mit Werkzeugen steht nun im Dienste des Begreifens informatischer Konzepte. Damit ist das Befolgen der verbliebenen acht Events of Instruction erst recht nicht mehr ein didaktisches Mittel erster Wahl – mehr Potential bieten handlungsorientierte Ansätze wie der im Folgenden grob umrissene: „In der ersten Phase, dem Benutzen, verwenden die Schüler Informatiksysteme. Mit den in dieser Phase gesammelten Erfahrungen wird in der zweiten Phase, dem Analysieren, der Aufbau und die Arbeitsweise des Informatik- bzw. seines konkreten Anwendungssystems

[1] Universität Rostock, Institut für Informatik, A.-Einstein-Str. 22, 18051 Rostock, lutz.hellmig@uni-rostock.de
[2] Richard-Wossidlo-Gymnasium Ribnitz-Damgarten, Schulstraße 15, 18311 Ribnitz-Damgarten, feedback@tinohempel.de
[3] Aus Gründen der besseren Lesbarkeit wird durchgängig das generische Maskulinum verwendet, welches männliche und weibliche Personen einschließt.

untersucht. Dadurch werden die Voraussetzungen geschaffen, dass die Schüler in der dritten Phase Gestalten, Lösungen selbstständig finden und die untersuchten Informatik- und deren Anwendungssysteme in neuen Zusammenhängen sinnvoll und effizient einsetzen." [**?**, S. 2]

Die erkundende und experimentelle Untersuchung von Strukturen und Zusammenhängen is kein speziellen Domäne der Informatik. Der Wissenserwerb in der Informatik kann oftmals in ähnlicher Weise wie in den Naturwissenschaften erfolgen, in denen sich mit dem erstmalig 1976 beschriebenen forschend-entwickelnden Unterricht eine Methode für den weitgehend selbständigen, handelnden Wissenserwerb etabliert hat [**?**, **?**]. Eine Zuordnung der Schritte Benutzen, Analysieren und Gestalten zu den Phasen des forschend-entwickelnden Unterrichts offenbart, dass der didaktische Gang mit einem Dreischritt nur unzureichend beschrieben wird und die notwendigen Phasen der Abstraktion der gewonnenen Erkenntnisse und der Wissenssicherung nicht beinhaltet. Wir schlagen darum vor, das Prinzip mit einer vierten Stufe, dem Verankern, zu vervollkommnen.

Tab. 1: Zuordnung der Stufen des Prinzips Benutzen-Analysieren-Gestalten-Verankern zu den Phasen des forschend-entwickelnden Unterrichts

Forschend-Entwickelnder Unterricht		**B-A-G-V**
Problemgewinnung	Problemgrund	Benutzen
	Problemfindung	Benutzen/Analysieren
	Problemformulierung	Analysieren
Überlegungen zur Problemlösung	Analyse des Problems	Analysieren
	Vorschläge zur Problemlösung	Gestalten
	Entscheidung für einen Lösungsweg	Gestalten
Durchführung des Problemlösevorschlags	Planung der konkreten Handlungen	Gestalten
	Umsetzen des Lösungswegs	Gestalten
	Erörterung und Zusammenfassung des Ergebnisses	*Verankern*
Abstraktion der gewonnenen Erkenntnisse	Ikonische Abstraktion	*Verankern*
	Verbale Abstraktion	*Verankern*
	Symbolhafte Abstraktion	*Verankern*
Wissenssicherung	Anwendungsbeispiele (Transfer)	Gestalten/Verankern
	Wiederholung Inhalt/Denkphasen	*Verankern*
	Lernzielkontrolle	*Verankern*

Der zeitliche Umfang der Phasen Benutzen, Analysieren, Gestalten und Verankern im Informatikunterricht sollte jeweils etwa gleich lang sein. In Tabelle 1 werden die unterschiedlichen Gewichtungen im didaktischen Gang zwischen Naturwissenschaft und Informatik sichtbar. Bevor die einzelnen Phasen des Prinzips allgemein charakterisiert werden, wird die Schrittfolge anhand eines prototypischen Unterrichtsbeispiels verdeutlicht.

2 Erstes Beispiel: Formeln in Tabellenkalkulationen

Das zentrale Merkmal, das die Tabelle eines Kalkulationsprogramms von einer Tabelle in einer Textverarbeitung unterscheidet, ist die Möglichkeit, funktionale Abhängigkeiten zwischen Werten abzubilden und die abhängigen Werte beim Verändern der Parameter automatisch aktualisieren zu lassen. Daraus lässt sich für die Schüler das **Lernziel** ableiten, Formeln in Tabellenkalkulationen unter Verwendung von Zellbezügen und Grundrechenarten aufstellen zu können. Während in den **Bildungsstandards Informatik** das Aufstellen von Formeln in einer Tabellenkalkulation überraschenderweise keine explizite Erwähnung findet, rechtfertigen landesspezifische **Curricula** (Tabellen analysieren; Formeln zur Problemlösung entwickeln und an die Tabellenstruktur anpassen) dieses Themengebiet [**?**]. Zur Einordnung der Unterrichtseinheit sind die **Lernvoraussetzungen** der Schüler zu berücksichtigen. Für die Realisierung dieses Unterrichtsbeispiels sollten die Schüler schon das Eintragen von Zahlen und Zeichenketten in die Kalkulationstabelle beherrschen (motivierbar beispielsweise durch die Diagrammerstellung aus statischen Daten), numerische Werte als Währung formatieren und die Feldbezeichnung in A1-Notation angeben können. Die Schüler erhalten als **Material** eine präparierte Kalkulationstabelle, anhand derer sie die Aufgaben des Arbeitsblatts bearbeiten.

	A	B	C	D	E	F	G
1	Vorstellung		öffentliche Generalprobe	Premiere	2. Vorstellung	3. Vorstellung	4. Vorstellung
2	Eintrittspreis		2	5	4	4	
3	Zuschauer	Lehrer	7	10			
4	Zuschauer	Schüler	8	30			
5							
6	Einnahmen		30	200	0		

Abb. 1: Datenansicht der Kalkulationstabelle

Würde man die Formeln der Tabelle einblenden, ergäbe sich das folgende Bild.

	A	B	C	D	E	F	G
1	Vorstellung		öffentliche Generalprobe	Premiere	2. Vorstellung	3. Vorstellung	4. Vorstellung
2	Eintrittspreis		2	5	4	4	
3	Zuschauer	Lehrer	7	10			
4	Zuschauer	Schüler	8	30			
5							
6	Einnahmen		=C2*(C3+C4)	200	=E2*(E3+E4)		

Abb. 2: Formelansicht der Kalkulationstabelle

Die **Aufgaben** des Arbeitsblatts beziehen sich auf einen vorangestellten Kontext – in diesem Falle den Betrieb eines Schülertheaters. Die Mitglieder einer Theatergruppe haben im Wahlpflichtfach „Darstellendes Spiel“ ein Stück einstudiert, das sie im Schultheater mit 40 Zuschauerplätzen aufführen wollen, solange ausreichend Besucher kommen. Um den Überblick über die Einnahmen zu behalten, möchten sie eine Tabellenkalkulation nutzen. Von einer anderen Theatergruppe haben sie dazu die Datei `eintrittsgelder.ods` erhalten – leider ohne Informationen darüber, wie die Tabelle zu benutzen ist.

Benutzen

1. In der Datei `eintrittsgelder.ods` befinden sich einige Daten.
   Formatiere dort, wo es sinnvoll ist, die Daten mit Währungssymbolen.

2. Ändere Eintrittspreise und Zuschauerzahlen. Beobachte die Inhalte in Zeile 6.
3. Beschreibe Probleme, die Dir beim Benutzen der Tabelle auffallen.

Analysieren

Den unformatierten Inhalt der markierten Tabellenzelle kann man in der Bearbeitungszeile über den Spaltenbeschriftungen sehen.

Betrachte den tatsächlichen Inhalt der Zellen in Zeile 6 in der Bearbeitungszeile und versuche, die Ursache für die in Aufgabe 1.3 gefundenen Probleme zu beschreiben.

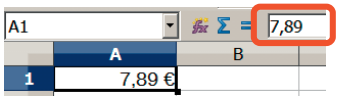


Abb. 3: Anzeige eines unformatierten Zelleninhalts in der Bearbeitungszeile

Gestalten

1. Behebe die erkannten Probleme in Zeile 6.
2. Für die Statistik ist auch die Gesamtzahl aller Zuschauer interessant. Die automatisch ermittelte Gesamtzahl der Zuschauer jeder Vorstellung soll in Zeile 5 zu sehen sein. Löse diese Aufgabe.
3. Mittlerweile hat die Theatergruppe beschlossen, dass Lehrer den doppelten Eintrittspreis bezahlen sollen. Verändere die Tabelle entsprechend.
4. Entwickle weitere Ideen, um die Tabelle zu verbessern und realisiere diese.

Verankern

1. Du hast das vorstehende Problem durch die Verwendung von Formeln in einer Tabellenkalkulation gelöst. Formuliere Deine Erkenntnisse dazu und halte sie im Heft fest. Kannst Du…
   - erklären, welche Vorteile der Einsatz von Formeln mit sich bringt?
   - beschreiben, was beim Eingeben von Formeln zu beachten ist?
   - verschiedenartige Beispiele für korrekte Formeln angeben?
   - fehlerhafte Formeln angeben und aufzeigen, was nicht beachtet wurde?
2. Welche offenen oder weiterführenden Fragen sind durch die Beschäftigung mit den Formeln entstanden?

3 Allgemeine Beschreibung der Methode

Anhand des ersten Beispiels lassen sich wesentliche Merkmale der didaktischen Schrittfolge allgemein beschreiben. Um alle Phasen der Schrittfolge für die Schüler plausibel

zu motivieren, Anregungen für eine kreative und ergebnisoffene Auseinandersetzung mit dem Thema zu geben und Möglichkeiten für eine kritische Betrachtung der verwendeten Modelle zu eröffnen, sollte eine **kontextuelle Einbindung** erfolgen.
Das **Benutzen** trägt motivierenden Charakter und dient den Schülern zur Erfassung der Situation und dem Bewusstmachen einer Problemstellung. Dies kann durch den Umgang mit einem unvollständigen oder fehlerhaften Informatiksystem bzw. durch die Bearbeitung der gleichen Aufgabe mit verschiedenen Informatiksystemen geschehen. Ein Nebeneffekt dieser Phase ist, dass die Schüler Routine im Umgang mit Informatiksystemen erwerben. Einen vertieften und modellhaften Einblick in die Problemlage können die Schüler gewinnen, wenn sie zuvor einen Sachverhalt ohne Zuhilfenahme von Informatiksystemen nachvollziehen – etwa durch das Nachspielen und Protokollieren einer Situation oder die Lösung eines Problems mit althergebrachten Mitteln wie dem Anfertigen und Ausfüllen einer Tabelle mit Stift und Papier. Dies eröffnet die Möglichkeit, informatische Vorgänge auf der enaktiven Ebene im wahrsten Wortsinne zu begreifen und die Beziehungen zwischen Prozessen aus der Lebensumwelt der Schüler und der Informatik zu erkennen.
Zur zielgerichteten Anregung des Erkenntnisprozesses bei den Schülern bedarf es geeigneter Aufgabenstellungen, die die innewohnenden Probleme des abstrakten Modells oder dessen konkreter Umsetzung offenbaren. Zielloses Interagieren mit Informatiksystemen kostet Zeit und garantiert nicht die Entdeckung eines Problems, das Aufdecken von Grenzen und Fehlern, was die notwendige Voraussetzung für die nächste Phase ist.
Im zweiten Schritt, dem **Analysieren**, stellen die Schüler Vermutungen zu den Ursachen und zu Möglichkeiten der Behebung des Problems auf. Das gelingt umso besser, je mehr die Schüler über methodische Kompetenzen verfügen – einschließlich der Bereitschaft, selbstständig forschend zu arbeiten. Bei Bedarf können hier unterstützende Impulse gegeben werden, die den Schülern Wege weisen, so selbstständig wie möglich die fachlichen Hintergründe des Problems zu entdecken und die zugrundeliegenden Modelle zu entwickeln. Für die Analyse können beispielsweise Informatiksysteme verglichen, Bezüge zwischen Informatiksystemen und nichtinformatischen Konzepten und Verfahren hergestellt, Systemmeldungen und Code untersucht sowie ggf. geeignete Literatur verwendet werden. Das verwendete Informatiksystem muss zwei wichtige Merkmale aufweisen.
(1) Um die Zahl der Instruktionen zu minimieren, sollten die zur Analyse nötigen Informationen im Informatiksystem formalsprachlich zugänglich sein: In einer Kalkulationstabelle können Formeln analysiert werden, für Probleme der Programmierung stehen Quelltexte und Fehlermeldungen zur Verfügung, in Textverarbeitungen kann das Einblenden nichtdruckbarer Zeichen Hinweise auf den Problemgrund liefern.
(2) Das untersuchte Informatiksystem muss durch die Schüler manipulierbar sein, um Hypothesen durch systematisches Experimentieren ([**?**]) überprüfen zu können und die Phase des Gestaltens zu ermöglichen. Im ersten Beispiel ist also nicht das Tabellenkalkulationsprogramm das zu untersuchende Artefakt, sondern die gegebene Kalkulationstabelle, die analysiert wird und deren Inhalte zielgerichtet verändert werden können.
In der Phase des **Gestalten**s sollten die Schüler möglichst viele Freiräume haben, die identifizierten Probleme zu beheben und an der weiteren Vervollkommnung der informatischen Lösung für den gegebenen Kontext zu arbeiten. Voraussetzung für einen hohen Anteil eigenverantwortlichen Arbeitens ist, dass die zur Weiterentwicklung des Produktes nötigen Techniken bereits exemplarisch im gegebenen Artefakt verwendet worden sein sollten, um

den Schülern das selbständige Herstellen von Analogien zu ermöglichen und einen Transfer des Wissens zu ermöglichen. Im Tabellenkalkulationsbeispiel erfolgt die Verbesserung des Systems über das Implementieren weiterer Formeln. Konzepte, die in der Vorgabe keine Rolle spielten – wie im Beispiel das Verwenden von Funktionen – liegen außerhalb der Zone der nächsten Entwicklung [**?**] und sind zu vermeiden. Die Phasen des Analysieren und des Gestaltens lassen sich selten trennscharf unterscheiden. Wie im nächsten Beispiel deutlich wird, können sich diese gegenseitig durchdringen oder in zyklischer Folge wiederholen. Mit dem letzten Schritt, dem **Verankern**, abstrahieren die Schüler vom entstandenen Produkt und werden sich ihres allgemeinen Lernfortschritts bewusst. Dies kann durch das Reflektieren und Zusammenfassen, das Formalisieren, das Systematisieren und Verknüpfen der Erkenntnisse mit vorhandenem Wissen oder Überlegungen zu möglichen Verallgemeinerungen erfolgen.

4 Diskussion weiterer Unterrichtsbeispiele

Die Verzahnung des Prozessbereichs Modellieren und Implementieren der Bildungsstandards mit dem Inhaltsbereich Algorithmen spiegelt sich in den Curricula wider. Im Rahmenplan Informatik Klasse 10 in Mecklenburg-Vorpommern wird in den Hinweisen zum verpflichtend zu unterrichtenden Thema „Sprachen und Sprachkonzepte“ eine Umsetzung beschrieben, die dem informationsorientierten didaktischen Ansatz folgt und explizit den Dreischritt Benutzen – Analysieren – Gestalten umsetzt. [**?**, Br05, **?**]

4.1 Imperatives Problemlösen

4.1.1 Benutzen und Analysieren I

„Benutzen und Analysieren bedeutet, dass die Schüler z. B. das gewählte Spiel zunächst ohne und dann mit Computer spielen, sich dabei die Spielregeln zu eigen machen und den Spielablauf Schritt für Schritt protokollieren. Sie unterscheiden Ein- und Ausgaben und beobachten die schrittweise Abfolge und eventuelle Wiederholung einzelner Schritte oder Schrittfolgen.“ [**?**, S. 47] Dieser enaktive Einstieg dient auch dazu, die Grenzen des Spiels auszutesten. Neben einer verbalen Beschreibung des Spielalgorithmus sollten die Schüler bereits hier in einer Übersicht die offensichtlichen Programmfehler sowie mögliche Verbesserungen erfassen.
Kontext: Nimm-Spiel Auf dem Spieltisch liegt eine zufällige Anzahl von Hölzern. Die beiden Spieler dürfen abwechselnd ein bis drei Hölzer wegnehmen. Der Spieler, der das letzte Holz nehmen kann, hat gewonnen.

Auftrag 1 Benutzen und Analysieren des realen Spiels

- Spiele mit deinem Partner das Spiel mehrfach.
- Beschreibe den Spielverlauf mit eigenen Worten.

- Ermittle die Bedingungen unter denen ein Spieler noch einen Zug machen kann.
- Auf welche Kriterien müssen die Spieler achten, damit nicht geschummelt werden kann?

Auftrag 2 Benutzen und Analysieren des Computerspiels

- Spiele mit deinem Partner das Spiel auf dem Computer mehrfach.
- Prüfe, ob der Spielverlauf mit dem realen Spiel übereinstimmt. Überarbeite/Konkretisiere ggf. deine Ablaufbeschreibung.
- Ermittle die Informationen, die ein- und ausgegeben werden und die sich das Programm merken muss.
- Prüfe, ob sich der Computer überlisten lässt.
- Erstelle eine Übersicht über Fehler in der Computerversion und Verbesserungsvorschläge.

4.1.2 Analysieren II und Gestalten I

„Die Schüler öffnen dann die Programmierumgebung und darin den Quelltext des zuvor benutzen Programms, vergleichen den Quelltext mit dem protokollierten Spielverlauf und lernen durch dieses Rückwärtsarbeiten die zur Lösung der Aufgabe erforderlichen Elemente der Programmiersprache sowie das zugrundeliegende Programmierparadigma kennen.“ [**?**, S. 47] Je nach Rahmenbedingung können dabei visuelle Programmiersprachen wie Scratch oder textuelle Systeme zum Einsatz kommen. Für den Lerneffekt ist entscheidend, dass sich die Lehrkraft an dieser Stelle mit Erläuterungen zu algorithmischen Grundstrukturen, logischen Bedingungen oder Variablen zurückhält, sondern allenfalls kurze Impulse gibt.

Auftrag 3 Teilanalyse der Implementation und Beseitigung logischer Fehler

- Gib die Aufgabe der Variable hölzer und der Variable Antwort an.
- Beschreibe den Algorithmus in Abbildung 4, (S. 152) mit eigenen Worten.
- Nenne den Teil des Regelwerks, der durch diesen Ausschnitt implementiert werden soll.
- Begründe, warum dies nicht vollständig gelingt.
- Beschreibe mit eigenen Worten notwendige Veränderungen in diesem Ausschnitt, um die Regel zu vervollständigen.
- Implementiere diese Veränderungen.
- Prüfe, ob die durchgeführten Veränderungen das Regelwerk für beide Spieler korrekt umsetzen. Korrigiere gegebenenfalls weiter.

Abb. 4: Nimm-Spiel. Programmausschnitt in Scratch

Die Bearbeitung der oben genannten Aufgaben führt zum ersten Verständnis von logischen Bedingungen. Hier wird der Fehler, dass mehr Hölzer genommen werden können als noch vorhanden sind, durch die Erweiterung der booleschen Bedingung beseitigt. Außerdem erfolgt die Übertragung der gefundenen Lösung in den Teilalgorithmus für Spieler 2.

4.1.3 Gestalten II

Auf diese Weise lassen sich nun vorhandene logische Fehler beheben und im Anschluss die gewünschten Verbesserungsvorschläge, wie etwa einen Rundenzähler oder die Möglichkeit der Spielwiederholung, umsetzen. Die dafür notwendigen Variablen, algorithmische Grundstrukturen und logischen Bedingungen können erneut durch Analogiebetrachtungen und Wissenstransfer eingebaut werden. Die Anforderungen dafür sind jedoch höher, da die Lösungen nicht mehr an anderer Stelle im Quelltext stehen.

Auftrag 4 Ausgestaltung der Spielimplementation:

- Beschreibe eine Möglichkeit, mit Hilfe einer neuen Variable runde, die Anzahl der gespielten Runden zu zählen und am Ende auszugeben.
- Erweitere das Projekt um eine solche Option.
- Erweitere das Projekt um die Möglichkeit der Spielwiederholung.

4.1.4 Verankern und Gestalten III

„Die weiteren Probleme sollten so geartet sein, dass die Schüler zu deren Lösung bereits Bekanntes aufgreifen können und nur wenige neue Gestaltungselemente erforderlich sind.“ [**?**, S. 47] Daher bieten sich einfache Glücks- und Würfelspiele, wie etwa Zahlenraten, Pasch, Einundzwanzig oder Gerneklein an. Die Verankerung der neuen Erkenntnisse und Kompetenzen sollte durch deren gezielte Bewusstmachung etwa durch Aufträge zur Gestaltung von Lernplakaten oder Übersichten erfolgen, auf denen die Schüler die Begriffe Variable, logische Bedingung oder Schleifenstruktur und deren Umsetzung darstellen.

Auftrag 5 Verankern

- Dein Nimm-Spiel ist nun fertig. Beschreibe in maximal vier Sätzen den Weg zu Erzeugung des fehlerfreien Spiels.
- Erstelle eine Übersicht über die Verwendung einer Variablen zur Speicherung und Anzeige der Spielrunden.

4.2 Relationale Datenbanken – Das Prinzip als Makromethode

Die bisher vorgestellten Beispiele beschreiben das Prinzip Benutzen – Analysieren – Gestalten – Verankern als didaktisches Vorgehen innerhalb einer kleinen Unterrichtssequenz. Am Beispiel der Behandlung des umfangreichen Themenfelds „Relationale Datenbanken“ soll die Anwendung als Makromethode skizziert und zur Diskussion gestellt werden. Das klassische Vorgehen bei der Vermittlung der Konzepte Relationaler Datenbanken folgt dem Schema: Beispiel/Motivation – Durchlaufen der Phasen der Datenbankerstellung (ER-Modell, Relationales Modell, Datenmodell) – Implementierung – Abfragen – Datenschutz (z. B. [BL02]). Das Benutzen von Datenbanken steht hier formal am Anfang, jedoch wird die Chance der Analyse und Gestaltung nicht genutzt.
Wir schlagen das folgende Vorgehen vor. Das Benutzen eines klassisches Karteikartensystem durch die Schüler zur Untersuchung von Ausleih- und Verwaltungsaufgaben einer Bibliothek steht am Anfang der Einheit. Dieses enaktive System liegt zusätzlich auch in Form einer Tabellenkalkulation mit mehreren Tabellenblättern vor. Durch das Analysieren beider Systeme erkennen die Schüler

- die grundsätzliche Eignung von Tabellen zum Speichern großer Datenmengen,
- die Grenzen von Tabellenkalkulationssystemen als Werkzeug zum Definieren, Administrieren, Manipulieren und Abfragen von Daten,
- die Technik des Verknüpfens von Tabellen über gemeinsame Attributwerte,
- die Notwendigkeit der Betrachtung von Aspekten des Datenschutzes.

Das erste Gestalten bietet sich mit der Behandlung von Abfragen und Aspekten des Datenschutzes an, alternativ könnte aber auch über den Prozess der Normalisierung die Gestaltung von Datenbanken thematisiert werden. Da Letzteres bei komplexen Anforderungsdefinitionen für Schüler eher ungeeignet ist sollte man den Weg über alle Phasen der Datenbankerstellung gehen. Das abschließende Verankern des erworbenen Wissens erfolgt durch eine Bewusstmachung des Lernfortschritts. Dazu können Impulsfragen zu den Grenzen der Tabellenkalkulationen oder zur Funktionsweise und zum Mehrwert der Datenbankanwendung dienen. Eine weitere Möglichkeit der Festigung des erworbenen Wissens ist die Bewertung der Eignung eines Werkzeugs (Tabellenkalkulation oder Datenbanksystem) für bestimmte Probleme.

5 Fazit und Ausblick

Die didaktische Schrittfolge Benutzen – Analysieren – Gestalten – Verankern erlaubt es Schülern, bei wohldosierter Impulsgebung, selbstständig und kontextbezogen Kompetenzen zu erwerben, unmittelbar anzuwenden und zu festigen. Die bisher entwickelten Ideen – neben den hier vorgestellten auch für die Nutzung von Formatvorlagen und zur logischen Programmierung – haben sich im Unterricht bewährt und sind auf Lehrerfortbildungen gut angenommen worden. Voraussetzung für einen gelingenden Unterricht ist neben gut ausgearbeiteten und optimierten Materialien und Aufgabenstellungen auch die Bereitschaft der Schüler, selbstständig zu arbeiten. Dies kann nicht bei der ersten Erprobung der Methode vorausgesetzt werden, sondern ist das Produkt stetiger Erziehung.

Literatur

[BL02] Burkert, J.; Lächa, R.: *Datenbanken. Materialien zum Unterricht, Sekundarstufe II*. Hessisches Landesinstitut für Pädagogik, Wiesbaden, 2. Auflage, 2002.

[Br05] Breier, N.: Informatik im Fächerkanon allgemein bildender Schulen – Überlegungen zu einem informationsorientierten didaktischen Ansatz. In S. Friedrich, Hrsg., *Lecture Notes in Informatics, Unterrichtskonzepte für informatische Bildung, 11. GI-Fachtagung Informatik und Schule*, Jgg. 60, Dresden, 2005.

[GB74] Gagné, R. M.; Briggs, L. J.: *The principles of instructional design*. Holt, New York, 1. Auflage, 1974.

[He07] Hempel, T.: Sprachen und Sprachkonzepte – Erfahrungen und Umsetzungsideen für eine Unterrichtseinheit für die Sekundarstufe I nach dem informationsorientierten didaktischen Ansatz. In P. Stechert, Hrsg., *Informatische Bildung in der Wissensgesellschaft*, number 6 in Reihe „Medienwissenschaften“. universi, Siegen, 2007.

[HG14] Hellmig, L.; Gramm, A.: Lernaufgaben mit experimentellem Charakter – Forschend entdecken und forschend entwickeln. *LOG IN*, 33(176/177):96–106, 2014.

[MfB02] Wissenschaft und Kultur Mecklenburg-Vorpommern Ministerium für Bildung, Hrsg. *Rahmenplan für Informatik der Jahrgangsstufen 7 bis 10 des Gymnasiums und der Integrierten Gesamtschule*. 2002.

[MfB08] Wissenschaft und Kultur Mecklenburg-Vorpommern Ministerium für Bildung, Hrsg. *Rahmenplan für das Fach Datenverarbeitung und Informatik in den Jahrgangsstufen 11 bis 13 am Fachgymnasium*. 2008.

[SL03] Schmidkunz, H.; Lindemann, H.: *Das forschend-entwickelnde Unterrichtsverfahren – Problemlösen im naturwissenschaftlichen Unterricht.* Number 2 in „Didaktik – Naturwissenschaften“. Westarp, Hohenwarsleben, 2003.

[Vy78] Vygotsky, L. S.: *Mind in society: The development of higher psychological processes*. Harvard University Press, 1978.

Modellvorstellungen zum Aufbau des Internets

Martin Hennecke[1]

Abstract: Verschiedene Studien haben Vorstellungen zum Aufbau des Internets untersucht und im breiten Umfang Fehlvorstellungen beschrieben, die ggf. auch zu Fehlverhalten führen können. Offenbar mangelt es den Probanden dieser Studien an tragfähigen Modellvorstellungen zum Aufbau des Internets. Im Sinne eines normativen Ansatzes werden in diesem Artikel für verschiedene didaktische Reduktionsstufen geeignete Modellvorstellungen vorgestellt, mit deren Hilfe im Unterricht Aufbau und Funktion des Internets erklärt werden können. Die beschriebene Abfolge der Modellvorstellungen ist ggf. auch in Verbindung mit einem Spiralcurriculum verwendbar.

Keywords: Modellvorstellung, Fehlvorstellung, Aufbau des Internets, didaktische Reduktion

1 Einführung

Im Tagungsmotto „Informatik allgemeinbildend begreifen“ ist „begreifen“ auch wörtlich zu nehmen. Viele informatische Inhalte sind jedoch nicht physisch und können nicht angefasst werden. Hier helfen Modelle, die abstrakte Inhalte gegenständlich oder bildhaft machen. Andere Inhalte, z. B. der physische Aufbau des Internets, können aufgrund ihrer Größe nicht durch „Begreifen“ erfahren werden. Auch hier können geeignete Modelle verständnisbildend wirken und damit zukünftiges Verhalten beeinflussen.

Offenbar mangelt es aber an derartigen Modellvorstellungen vom Aufbau des Internets. Dies zeigen diverse Interview- und Fragenbogenstudien der vergangenen Jahre. So berichten z. B. [DZ10] bzw. [DWZ12] von einer Studie mit der sie Modellvorstellungen von Schülern der 7. und 8. Jahrgangsstufe erhoben haben. In 11 Interviews befragten sie 23 Schüler zum E-Mail-Versand, zum Chatten und zum Video-Streaming. Indirekt geben viele Antworten aber auch Einblicke in Modellvorstellungen zum Aufbau des Internets. [Pa05] befragte 340 griechische Schüler der 8. und 9. Jahrgangsstufe (max. 1 Jahr Informatikunterricht). Der Fragebogen wollte von den Schülern insbesondere wissen, wie sie sich den physischen Aufbau des Internets vorstellen. Hierzu sollten die Probanden ein Bild vom Internet zeichnen. Diese Methode wurde zuvor in [TG98] für eine Studie mit 51 erwachsenen Probanden einer südafrikanischen Universität verwendet.

Während die genannten Studien existierende mentale Modelle deskriptiv untersucht haben, wird in dieser Arbeit eine Abfolge möglicher vermittelbarer Modellvorstellungen vorgestellt. Es handelt sich also um einen normativen Ansatz, der für verschiedene didaktische Reduktionsstufen geeignete, ggf. in einem Spiralcurriculum unterrichtbare,

[1] Universität Würzburg, Fakultät für Mathematik und Informatik, Emil-Fischer-Str. 30, 97074 Würzburg, martin.hennecke@uni-wuerzburg.de

fachlich tragfähige Modelle zusammenfasst. Die Abfolge der Modellvorstellungen ist mit Schülern der 8. und 11. Schuljahrgangsstufe sowie mit Studierenden in der Bachelor- und Masterphase mehrfach erprobt worden.

Im Unterricht werden Modelle des Internets oft thematisch mit der Behandlung eines Kommunikationsmodells verknüpft. Das wohl einfachste Kommunikationsmodell ist das Sender-Empfänger-Modell von Shannon und Weaver. Im Sender-Empfänger-Modell werden Informationen durch eine Informationsquelle produziert. Diese Informationen werden von einem Sender codiert, d. h. durch Zeichen (Daten) dargestellt, die über den Übertragungskanal verendet werden. Dabei kann die codierte Nachricht unter Umständen durch Störquellen verändert oder mitgelesen werden. Der Empfänger muss die empfangenen Zeichen decodieren, um sie für das Ziel interpretierbar zu machen. Hierzu muss ihm die Art der Codierung bekannt sein. Durch Umkehr der Rollen kann eine Nachricht zurückgesendet werden. In vielen Kontexten wird das Sender-Empfänger-Modell durch Gleichsetzung der Informationsquelle und des Senders (bzw. des Empfängers und des Ziels) verkürzt. Nachfolgend beschriebene Modelle über den Aufbau des Internets werden daher auch unter diesem Blickwinkel betrachtet.

2 Fehlende Kommunikationsmodelle

Auch wenn Schüler heute mit dem Internet aufwachsen, heißt das nicht automatisch, dass sie sich die Existenz eines Netzwerkes ausreichend bewusst gemacht haben. So berichtet [Pa05], dass 23 % der Schüler abstrakte Bilder von Internetdiensten zeichneten. Bei weiteren 29 % der Schüler fand sich nur ein einzelner Computer ohne Anschluss an ein Netzwerk. Diese Größenordnungen werden von [TG98] bestätigt. Hier zeichneten 43 % der universitären Probanden vergleichbare Bilder. [Pa05] geht davon aus, dass sich das mentale Modell vieler dieser Schüler auf ihren konkreten Computer bzw. auf die von ihnen konkret genutzten Schnittstellen zu Internetdiensten (z. B. dem Webbrowser) beschränkt. Dazu würde passen, dass diese Schüler weitestgehend keinen Unterschied zwischen dem Internet und dem World Wide Web machen. [Pa05] fand Hinweise darauf, dass einige Schüler konkret davon ausgehen, dass die Inhalte des World Wide Webs bereits fest auf ihrem Computer gespeichert sind. Falls sie hier richtig liegt, könnten diese Schüler die Interaktion mit anderen Nutzern in Foren oder sozialen Netzwerken nicht modellkonform begreifen und folglich auch keine Gefahren bei der Preisgabe von Informationen abschätzen.

Jede Modellvorstellung vom Internet, die Lücken im Kommunikationsmodell aufweist, ist im Sinne eines auf die Lebenswirklichkeit ausgerichteten Unterrichts unzureichend. Sie sollte auf jeder didaktischen Reduktionsstufe als Fehlvorstellung eingestuft werden und im Unterricht entsprechende Reaktionen erfahren. Um den Fehlvorstellungen keinen Vorschub zu leisten, ist es sicherlich ratsam, die Begriffe Internet und World Wide Web im Sprachgebrauch so sauber wie möglich zu trennen – auch wenn einem dies die Alltagssprache („Internetseiten“) und viele Lehrwerke nicht leicht machen.

3 Naive metaphorische Modelle vom Internet

Teil einer naiven, aber verbreiteten Modellvorstellung vom Internet ist, sich der Existenz einer Datenleitung bewusst zu sein ohne eine Vorstellung über deren Aufbau zu haben. Dieses Modell beschreibt das Internet primär über seinen Nutzen als Kommunikationsmedium, d. h. um Nachrichten auszutauschen oder um Informationen abzurufen. Laut [Pa05] beschreiben 65 % der Schüler das Internet entsprechend nicht als technische Infrastruktur, sondern über die Möglichkeiten, es zu benutzen. Auch auf dieser sehr elementaren didaktischen Reduktionsstufe lassen sich geeignete Abbildungen zum Internet gestalten. Da hier vom physischen Aufbau noch vollständig abstrahiert wird, muss das Kommunikationsmodell im Vordergrund stehen. In vielen Schulbüchern finden sich Abbildungen, die zwei über ein Kabel miteinander verbundenen Rechner zeigen (vgl. Abb. 1). Anhand dieser lassen sich große Teile des Sender-Empfänger-Modells gut erklären. So finden sich unmittelbar der Sender, der Empfänger und der Übertragungskanal wieder. Störquellen, Codierungs- und Decodierungsprozesse sowie die Umkehr der Kommunikationsrichtung könnten modellkonform ergänzt werden. Der Strich in Abb. 1 lässt sich intuitiv gut als Kabel zwischen zwei Rechnern interpretieren. Weitere Rechner können modellkonform durch zusätzliche Kabel ergänzt werden. Dies ermöglicht zwar den einfachen Übergang zu einer Modellvorstellung von lokalen Netzwerken (vgl. Abb. 3), der Weg zu einer tragfähigen Vorstellung vom Internet ist bei diesem Ansatz jedoch noch sehr lang.

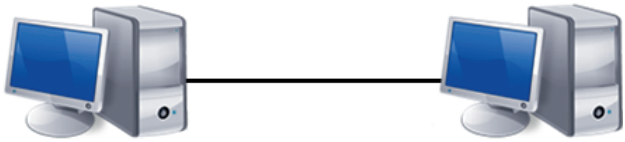


Abb. 1: Datenleitung als konkrete bzw. metaphorische Modellvorstellung

In einigen Schulbüchern wird daher versucht, den Strich zwischen den beiden Rechnern als Metapher für einen längeren Übertragungskanal zu interpretieren. Wegen der optischen Nähe zwischen Linien und Kabeln fällt dieses Verständnis eher schwer. Da ist die Gefahr groß, dass trotz einer metaphorischen Intention eine Interpretation als einfaches Kabel stattfindet. Bei Schülern sind derart reduzierte Vorstellungen selten. Laut [Pa05] zeichnen lediglich 6 % der Schüler derartige Bilder. Dabei bleibt offen, ob sie an ein konkretes Kabel oder einen metaphorischen Übertragungskanal gedacht haben.

Die metaphorische Interpretation der Linien führt zudem im weiteren Verlauf zu einer Überladung der Symbolik, so dass Schüler Linien mal konkret und mal metaphorisch interpretieren müssen. Darstellungen, die dies vermeiden wollen, zeigen das Internet oft nur als metaphorische Wolke (vgl. Abb. 2). Die Wolke ist dabei als eine Black-Box zu verstehen, die für den (noch) unbekannten physischen Aufbau des Netzes steht. Metaphorische Darstellungen in diesem Sinn werden bei [Pa05] nicht als eigene Kategorie beschrieben und sind entsprechend wohl bereits als vermittelte Modellvorstellungen

einzustufen. Umso wichtiger ist es, modellbedingten Fehlvorstellungen konsequent entgegenzuwirken. Entsprechend sollten Darstellungen mit einem einzelnen, mit der Wolke verbundenen Rechner vermieden werden. Diese Darstellung holt die Schüler zwar bei ihren Erfahrungen ab, ermöglicht aber keine Erklärung des Sender-Empfänger-Modells, da Übertragungskanal und Empfänger verschmelzen. Sinnvoller ist stattdessen, den Kommunikationspartner ebenfalls außerhalb der Wolke zu zeichnen, so dass sich Sender, Empfänger und Übertragungskanal (als metaphorische Wolke) identifizieren lassen.

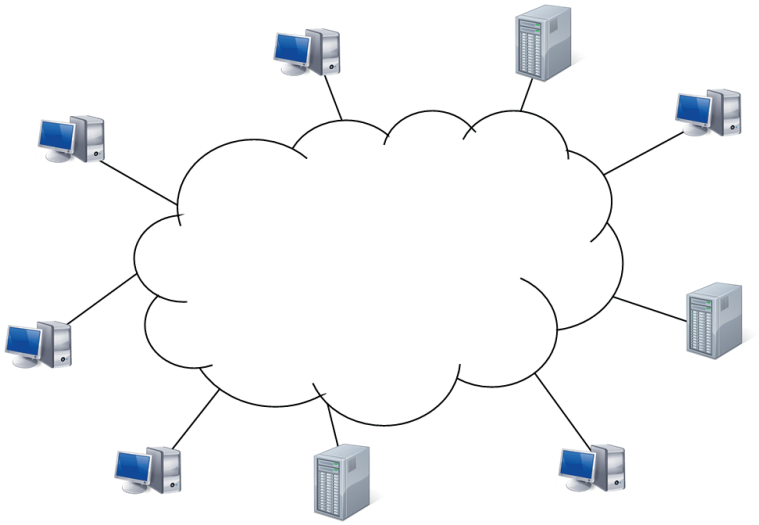


Abb. 2: Wolke als metaphorische Modellvorstellung

Von den griechischen Schülern bejahten laut [Pa05] 56 % die Frage, ob die Informationen im Internet in einem zentralen Computer gespeichert sind. [DWZ12] bestätigt die hohe Verbreitung dieser Fehlvorstellung auch bei deutschen Schülern (40 %). Entsprechende Zeichnungen finden sich jedoch nur bei 9 % [Pa05] bzw. 16 % [TG98] der Probanden. [Pa05] ergänzt, dass einige Schüler davon ausgingen, dass alle Informationen zentral überprüft und fehlerhafte Informationen entfernt würden. Ein schönes Beispiel dafür, wie Fehlvorstellungen über die Infrastruktur zu (für das tägliche Leben relevanten) Fehlvorstellungen über die darauf aufbauenden Dienste führen. Zur Vorbeugung dieser Fehlvorstellung lassen sich mehrere Server rund um die Wolke anordnen und ggf. mit bekannten Internetdiensten beschriften. Die Verwendung unterschiedlicher Symbole für Rechner von Nutzern und von Servern kann sinnvoll sein, wenn der Fehlvorstellung vorgebeugt werden soll, dass alle Informationen auf allen Rechnern liegen.

Dieses Modell ermöglicht die Erklärung des Sender-Empfänger-Modells ohne auf konkrete physische Infrastrukturen eingehen zu müssen. Damit ist es auch für jüngere Schüler gut geeignet und kann, entsprechend erklärt, wirksam einigen sehr problematischen Fehlvorstellungen vorbeugen. Im Zweifel ist eine Wolke besser, als fehlweisende Abbildungen konkreter Netzwerke, wie man sie leider in vielen Schulbüchern findet.

4 Lokale Netzwerke als Bild vom Internet

Das Innenleben der Wolke ist für Schüler weder in dem beschriebenen Modell noch in der Realität greif- oder sichtbar. Obwohl allgegenwärtig entzieht es sich weitestgehend der Erfahrungswelt der Schüler. Hingegen ist die lokale Vernetzung der Schule oder zu Hause auch physisch präsent. Diese dürfte heute größtenteils mit Twisted-Pair-Kabeln in einer sternförmigen Topologie ausgeführt sein (vgl. Abb. 3). Die (früher) übliche Diskussion der Vor- und Nachteile verschiedener Topologien (Stern, Bus, Ring) hat damit ihre Anknüpfung an die Lebenswirklichkeit fast verloren. Aus informatischer Sicht und mit Blick auf die zukünftige Bedeutung für die Schüler bleibt vielleicht der Vergleich zu Bussystemen interessant. Man denke beispielsweise an Bussystemen in der Industrie (z. B. CAN-Bus), der Hausautomation (z. B. KNX-Bus) oder auf dem Mainboard.

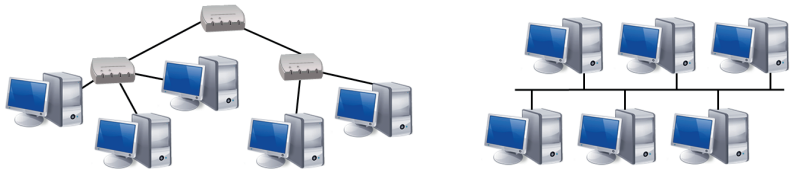


Abb. 3: Exemplarische Beispiele typischer lokaler Netzwerke (Stern, Bus)

Neben ihrem Bezug zur realen Welt ist die lokale Vernetzung auch wegen ihres enaktiven Zugangs im Unterricht bedeutsam. So lassen sich im Labornetzwerk der Aufbau und die Funktion von Netzwerken praktisch nachvollziehen, die Aufgaben der verschiedenen Komponenten erkennen und die Funktionsweise der Protokolle erproben. Dies entspräche z. B. den Anforderungen des bayerischen Lehrplans für die Realschule (vgl. [ISB08]). Wer den dafür nötigen technischen Aufwand nicht treiben kann (oder will), ist mit Simulatoren vergleichbar gut bedient. Für die Schule bietet sich insbesondere die Lernsoftware FILIUS (vgl. [Fr09]) mit ihrem Skriptum (vgl. [Ga11]) an.

So hilfreich derartige Curricula für das Verständnis lokaler Netze sind, so anfällig für Fehlvorstellungen sind sie, wenn es um die Übertragung auf größere lokale Netze oder auf das gesamte Internet geht. In beiden Fällen kommen vernetzte Infrastrukturen zum Einsatz. Ohne deren Kenntnis sind Schüler jedoch geneigt, die ihnen bekannten Topologien zu extrapolieren. So berichtet [Pa05], dass 41 % der Schüler, die überhaupt miteinander vernetzte Rechner zeichneten, lokale Vernetzungen als Bild vom Internet wählten (14 % aller Schüler). Aufgrund der hohen Verbreitung sternförmiger Topologien fördert dies also die Fehlvorstellung, dass das Internet auch phsyikalisch hierarchisch organisiert sei.

5 Internet als Netz von Routern

In Lehrwerken sind Darstellungen weit verbreitet, die das Internet als Netz von Routern zeigen (vgl. Abb. 4). Das Modell nutzt dabei aus, dass einigen Schülern der Begriff

„Router“ vom heimischen Internetanschluss bekannt ist. Im Unterschied zu Abb. 4 fehlt bei vielen derartigen Abbildungen jedoch die metaphorische Wolke oder sie wird durch eine Landkarte ersetzt. Auch die „Sendung mit der Maus“ setzt in der Sachgeschichte „Internet“ auf diese Vorstellung. So ordnet sie die Router wahlweise als wegweisende Personen in einem Gebäude oder als Punkte auf einer Deutschlandkarte an [WDR99, ca. 6. Min.]. Eine derartige nationale Einschränkung ist, von wenigen Beispielen abgesehen, fachlich unangemessen und verleitet unnötig zu Fehlvorstellungen. Als Beispiel sei die Fehlvorstellung genannt, dass eine E-Mail zwischen einem Sender und einem Empfänger in Deutschland nur innerdeutsch übertragen wird. Die methodisch schöne Brücke zur Lebenswirklichkeit ist also nur mit großer Vorsicht einsetzbar. Ferner sollte darauf geachtet werden, dass ein ausreichend großes Netz abgebildet wird, das nicht an eine der bei lokalen Netzwerken üblichen Topologien (Stern, Bus, Ring) erinnert.

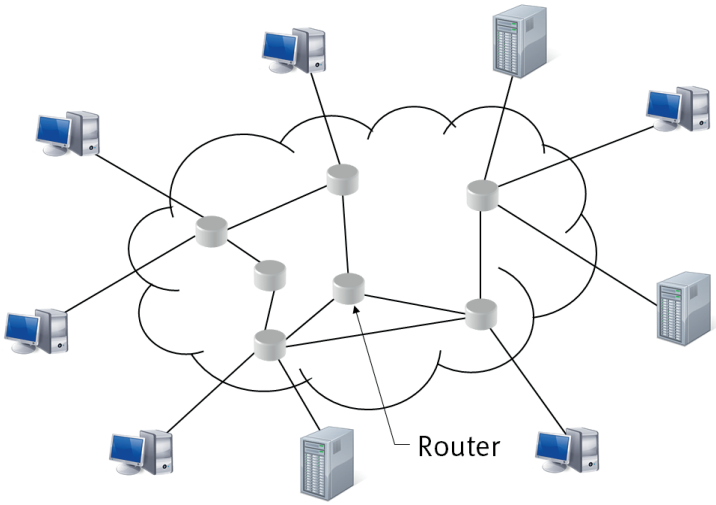


Abb. 4: Internet als Netz von Routern

Ein fachdidaktischer Vorzug der Modellvorstellung „Netz von Routern“ ist, dass sie die topologische Betrachtung der lokalen Netzwerke auf naheliegende Weise fortführt. Mit Blick auf das Sender-Empfänger-Modell sind nicht nur Sender, Empfänger und Übertragungskanal, sondern auch Störquellen sowie Gefahren durch Manipulation oder Mitlesen von Daten einfach darstellbar. Am Beispiel alternativer Wege durch das Netz lässt sich die Wegwahl (engl. Routing) anhand von Tabellen und der algorithmischen Bestimmung derselben (dynamisches Routing) auf angemessenem Niveau erklären. Damit ist das „Netz von Routern“ ein einfaches und dennoch leistungsfähiges Modell, das auch in der Unterstufe bereits ohne größere Probleme eingeführt werden kann. Fachlich betrachtet ist es jedoch nur für die Anfangstage des Internets bzw. für die Erklärung von Teilstrukturen tragfähig. Ein Netz von Routern in der dargestellten Form ist nicht gut skalierbar, da z. B. dynamisches Routing nicht effizient genug wäre und wirtschaftliche Interessen an die Grenzen ihrer administrativen Umsetzbarkeit kämen.

Im Unterricht sind die Grenzen diese Modells erreicht, wenn Schüller im Rahmen einer Routenverfolgung (tracert, pathping oder visualroute) zwischen zwei Rechnern in ihrer Stadt Routen über New York aufspüren. Es ist an dieser Stelle schwer vermittelbar, warum es in einer derartigen netzartigen Struktur keine „bessere" Verbindung innerhalb der Stadt oder wenigstens der Region geben soll. Entsprechend scheitert hier auch die Sachgeschichte der „Sendung der mit der Maus", wenn sie erklärt, „dass meine Anfrage unter Umständen einen riesigen Umweg […] machen muss. Aber das ist eben das Internet." Zunehmend werfen aber auch Computerkriminalität, geheimdienstliche Aktivitäten oder Angriffe auf Netzstrukturen bei Schülern Fragen auf, die mit den bisherigen Modellvorstellungen nicht zufriedenstellend beantwortet werden können.

6 Internet als Netz von Autonomen Systemen

Um das „Netz von Routern" zu erweitern, kann man die Verbindungen zwischen den Routern metaphorisch interpretieren oder sie als Wolken zeichnen (vgl. Abb. 5). Bei dieser Erweiterung gilt es der Fehlvorstellung vorzubeugen, dass mit den Wolken lokale Netzwerke (LANs) gemeint seien. Eine derartige Deutung wäre zwar fachlich möglich, würde aber bei der Überwindung der bisherigen Modellgrenzen nicht weiterführen. Es bietet sich daher die Nutzung des Fachbegriffs „Autonomes System" an, auch wenn dieser auf dem ersten Blick keinen Bezug zur Lebenswirklichkeit der Lernenden zu haben scheint. Der Bezug ist jedoch mit der Nennung von bekannten Beispielen, wie der Deutschen Telekom (T-Online), United Internet (1&1, Freenet), Telefónica (O2, E-Plus) und Arcor leicht herzustellen. Als Anbieter im Endkundengeschäft sind diese Autonomen Systeme vielen Schülern gut bekannt. Mit einer modellkonformen Abbildung dieser Anbieter hat die Modellvorstellung daher einen erfreulichen Nebeneffekt. Abb. 5 zeigt innerhalb der Wolken der Autonomen Systeme verschiedene topologische Grundkonzepte. Dies soll illustrieren, dass die physische Gestaltung und die verwendeten Protokolle unter (administrativer) Kontrolle des Autonomen Systems liegen. Einige große Autonome Systeme sind ihrerseits durch Autonome Systeme unterstrukturiert. Die Wölkchen innerhalb der Autonomen Systeme können daher als lokale Netzwerke oder (für leistungsstarke Schüler oder Studierende) als Autonome Systeme interpretiert werden. Im Interesse einfacher Abbildungen können die inneren Wolken alternativ leer verbleiben. Reale Autonome Systeme sind geographisch stark überlappend. Überlappende Wolken wären also zwar fachlich angemessen, aber leider schnell sehr unübersichtlich.

Mit diesem Modell lassen sich nun die von den Schülern beobachtbaren, nicht plausibel wirkenden Routen erklären (vgl. Kap. 5). Dazu ist es wichtig zu vermitteln, dass, obwohl viele Autonome Systeme einander zwar regional stark überlappen, aus administrativen, finanziellen und technischen Gründen nur an sehr wenigen Orten ein Austausch zwischen ihnen stattfindet. Derartige Austauschpunkte werden als „privat" bezeichnet. Angenommen, der einzige private Austauschpunkt zwischen der Deutschen Telekom und Telefónica befände sich in Frankfurt, dann müssten alle Daten, die zwischen zwei Rechnern dieser beiden Systeme übertragen würden, auch hier ausgetauscht werden.

Dies erklärt also z. B. den Weg der Daten von München über Frankfurt zurück nach München. Zwischen den beiden Städten, d. h. jeweils innerhalb eines der beiden Autonomen Systemen, verhält sich der Weg der Daten im Wesentlichen, wie es die Schüler mit der Modellvorstellung „Netz von Routern“ erwarten würden.

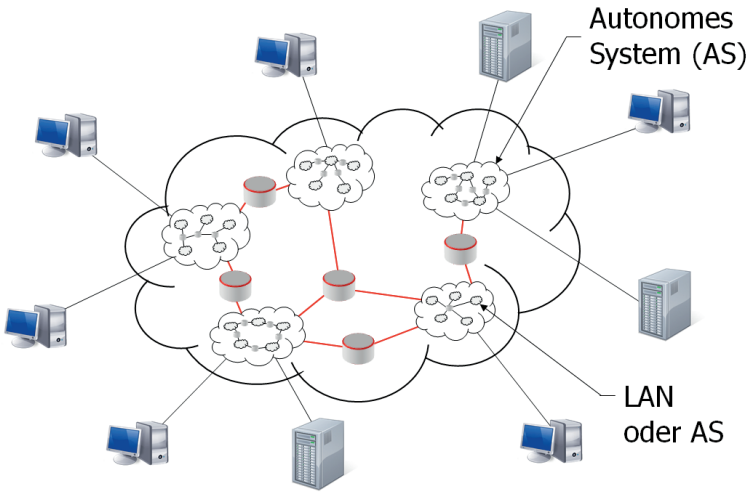


Abb. 5: Internet als Netz von Autonomen Systemen

Viele Autonome Systeme unterhalten (ggf. auch trotz regionaler Überlappung) keine gemeinsamen Austauschpunkte. Dann wird mit den Routinginformationen, die an den Austauschspunkten des Autonomen Systems des Senders zu Dritten vorliegen, ein Weg durch diese Systeme bestimmt (Transit). Beispielsweise würden in Abb. 5 die Daten aus dem Autonomen System oben links über die beiden unteren Systeme in das Autonome System oben rechts übertragen werden. Der Routingprozess kann fachdidaktisch so vereinfacht werden, dass Routing auf zwei Ebenen stattfindet: Die Router an den Austauschspunkten sind für einen Weg durch das Netz der Autonomen Systeme verantwortlich, während die Router in den Autonomen Systemen den Weg durch das jeweilige System bestimmen müssen. Dieser Kunstgriff sorgt für eine Skalierung des dynamischen Routings. Insbesondere hat damit ein kompensierbarer Ausfall innerhalb eines Autonomen Systems keine Auswirkungen auf die globalen Routingtabellen an den weltweit verstreuten Austauschpunkten. Entdecken Schüler also eine Verbindung über New York zwischen zwei deutschen Rechner, dann haben sie nicht die NSA auf frischer Tat erwischt – sondern nur eine Verbindung gefunden, deren kürzester Weg durch das Netz der Autonomenen Systemen einen entsprechenden Austauschpunkt nutzt. Wäre die NSA auf derartige (eher zufällig wirkende) Verbindungen angewiesen, wäre ihre abhörende Tätigkeit in Europa nicht sonderlich effektiv. Hingegen könnten bei einem Zugriff auf einen Austauschpunkt sehr effizient viele Verbindungen mitgelesen werden. Dieser Aspekt macht die nächste (kleinere) Erweiterung des Modells interessant.

7 Internet als Netz von Autonomen Systemen mit Public Peering

Kleine und mittlere Autonome Systeme sind in hohem Maße auf die Datenweiterleitung durch andere Autonome Systeme angewiesen. Nehmen sie dazu die Weiterleitung durch größere Autonome Systeme in Anspruch, wird ihnen dieser Dienst in der Regel in Abhängigkeit vom Datenvolumen in Rechnung gestellt. Zur Vermeidung dieser Kosten müssten sie mit möglichst vielen vergleichbaren Partnern private Austauschpunkte aufbauen und diese auf Gegenseitigkeit betreiben. Dies rechnet sich in der Regel jedoch nicht. Die Lösung sind öffentliche Austauschpunkte, an denen möglichst viele Autonome Systeme Daten (im Wesentlichen auf Gegenseitigkeit) austauschen. Derartige, in den Medien gern als Datenknoten bezeichnete Austauschpunkte, existieren weltweit nur wenige. Als Fachbegriffe eignen sich Begriff wie „Commercial Internet Exchange“ oder allgemeiner „Public Peering“. Für Schüler ist es in diesem Kontext meist sehr überraschend, dass der weltweit größte Datenknoten nicht irgendwo in den USA, sondern in Frankfurt anzufinden ist (DE-CIX). Die Spitzenlast von DE-CIX beträgt etwa 3,8 Terabit/s (Stand Januar 2015, akt. Werte unter [DC15]). Dieses Datenvolumen kann Schülern nur im Vergleich begreifbar gemacht werden. 3,8 Terabit/s entsprechen ca. 640 vollen CDs oder dem Text von Büchern aus ca. 2300 Bibliotheksregalen pro Sekunde.

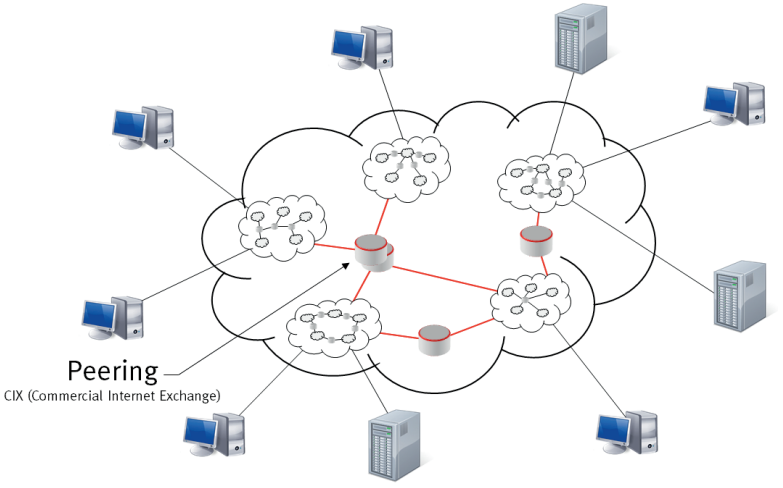


Abb. 6: Internet als Netz von Autonomen Systemen mit Public Peering

Die Erweiterung des Modells „Netz aus Autonomen Systemen“ um öffentliche Austauschpunkte ermöglicht eine fachlich sehr tragfähige Modellvorstellung. Sie ist inhaltlich nicht sonderlich anspruchsvoll, öffnet aber eine Reihe interessanter Optionen. So können vor dem Hintergrund des Datenvolumens und seines Wachstums nicht nur technische, sondern auch gesellschaftliche Fragen thematisiert werden. Viele Presseberichte zur NSA-Affäre lassen sich erst vor diesem Hintergrund fachlich verstehen und in ihrer Tragweite einordnen. Insofern ist diese letzte Erweiterung des Modells sehr lohnend.

8 Empfehlung

Aus Sicht des Autors sollten Schüler das Internet nicht nur auf der Grundlage auswendig gelernter Verhaltensregeln nutzen, sondern auch aufgrund eines informatischen Grundverständnisses zu eigenverantwortlichen und selbstbestimmten Nutzungsentscheidungen kommen können. Unabdingbar dazu ist ein Grundverständnis von Aufbau und Funktion des Internets. Das einfachste Modell, das dieses bzgl. des Aufbaus des Netzes zu liefern vermag, ist das Modell „Netz von Routern“ (Kap. 5). Es ist für die Unterstufe geeignet und kann dort problemlos vermittelt werden. Im Sinne eines Spiralcurriculums sind die beiden weiterführenden Modelle auch in späteren Informatikangeboten sinnvoll.

Literaturverzeichnis

[DWZ12] Diethelm, I.; Wilken, H.; Zumbrägel, S.: An investigation of secondary school students' conceptions on how the Internet works. In: *Proc. 12th Koli Calling conference on computing education research*, Tahko, 2012.

[DC15] DE-CIX, DE-CIX Management GmbH, www.de-cix.net, Stand: 23.01.2015.

[DZ10] Diethelm, I.; Zumbrägel, S.: Wie funktioniert eigentlich das Internet? - Empirische Untersuchung von Schülervorstellungen,". In: *Proc. Didaktik der Informatik. Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik*, Köllen Verlag, Bonn, S. 33-44, 2010.

[Ga11] Garmann, D.: Skriptum zum Unterricht – Netzwerke mit Filius, 2011. Zit. n.: http://www.lernsoftware-filius.de/Begleitmaterial, Stand: 14.01.2015.

[Fr09] Freischlad, S.: Entwicklung und Erprobung des didaktischen Systems Internetworking im Informatikunterricht. Universitätsverlag Potsdam, 2009. Zit. n.: http://dokumentix.ub.uni-siegen.de/opus/volltexte/2009/405/.

[ISB08] ISB: Informationstechnologie, Staatsinstitut für Schulqualität und Bildungsforschung München. Zitiert nach: http://www.isb.bayern.de/download/8868/lehrplan_informationstechnologie_082008.pdf, 2008, Stand 24.01.2015.

[Pa05] Papastergiou, M.: Students’ Mental Models of the Internet and Their Didactical Exploitation in Informatics Education. Education and Information Technologies, 10 (4), S. 341-360, 2005.

[TG98] Thatcher, A.; Greyling, M.: Mental models of the Internet. In: International Journal of Industrial Ergonomics, 22, S. 299-305, 1998.

[WDR99] WDR, Westdeutscher Rundfunk Köln: Die Sendung mit der Maus – Sachgeschichte zum Internet, vermutl. Erstausstrahlung 1999. Akt. Fassung unter: http://www.wdrmaus.de/sachgeschichten/sachgeschichten/internet.php5, Stand: 14.01.2015.

Projektarbeit im Informatikunterricht - Bau und Anwendung eines 3D-Druckers

Henry Herper [1], Volkmar Hinz[2] und Philipp Schüßler[3]

Abstract: Rapid Prototyping ist in der Industrie ein Fertigungsverfahren, bei dem ohne manuelle Umwege 3D-Modelle in Produkte umgesetzt werden. Nach mehr als 30 Jahren Industrieeinsatz ist dieses Verfahren mit dem 3D-Drucker auch für die Schule einsetzbar geworden. Die Schülerinnen und Schüler können alle Phasen der Modellerstellung und Modellumwandlung selbst gestalten und die einzelnen Arbeitsschritte verfolgen. Im vorgestellten Projekt wird gezeigt, wie aus einem Bausatz ein funktionsfähiger Drucker erstellt wird und wie mit diesem Drucker Produkte hergestellt werden können. Für den Technikunterricht ergibt sich die Möglichkeit, aus gedruckten Teilen komplexere Systeme zusammenzufügen und ihre Funktionsfähigkeit zu testen. Bezüglich der Anforderungen an Oberflächen und Toleranzen werden die Grenzen dieses Produktionsverfahrens deutlich.

Keywords: Prototyping, 3D-Drucker, fächerverbindendes Projekt

1 Einleitung

Informatische Inhalte dringen immer stärker in die Lebenswelt der Schülerinnen und Schüler ein. Schwerpunkt ist die Bedienung bildschirmbasierter Systeme. Informatiksysteme durchdringen auch weitere Bereiche der Lebenswelt. Für einen kontextbezogenen Informatikunterricht gibt es eine Vielzahl von Anwendungsgebieten. Etabliert haben sich im technischen Bereich vor allem die computerbasierte Steuerung von Geräten. Seit einigen Jahren nimmt die Verbreitung von 3D-Druckern ständig zu und die Preise haben ein Niveau erreicht, das diese Technik auch im Informatikunterricht einsetzbar macht. Der Aufbau und die Nutzung der Drucker berühren viele Teilgebiete der Informatik und des Technikunterrichtes. Hier bietet sich der fächerverbindende Ansatz von Informatik und Technik an. Der Aufbau eines 3D-Druckers aus einem Bausatz erfordert neben handwerklichen Fähigkeiten auch ein sehr präzises Arbeiten. Mit dem Zusammenbau lernen die Schülerinnen und Schüler die einzelnen Komponenten des Druckers in ihrem Aufbau und ihrer Funktionsweise kennen. Das Problem heutiger technischer Systeme besteht darin, dass sie in der Regel vollständig gekapselt sind. Das wird mit diesem Bausatz vermieden.

[1] Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, PF 4120, 39016 Magdeburg, henry.herper@ovgu.de
[2] Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, PF 4120, 39016 Magdeburg, volkmar.hinz@ovgu.de
[3] Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, PF 4120, 39016 Magdeburg, philipp.schuessler@st.ovgu.de

Die Nutzung des 3D-Druckers zeigt eine Vorgehensweise, die heute auch in der Industrie bei der Herstellung von Prototypen oder Ersatzteilen angewendet wird. Die Schüler können alle Schritte, von der Idee über die Zeichnung zum 3-D-Datenmodell bis zum fertigen Produkt selbst durchführen. Fehler müssen nicht vom Lehrenden korrigiert werden, sondern sind am Endprodukt deutlich sichtbar und können von den Lernenden ausgewertet werden.

2 Bau des 3D-Druckers

Im Rahmen des Workshops wird ein Projekt vorgestellt, in dem ein 3D-Drucker aus einem Bausatz zusammengebaut und anschließend genutzt wird. Für die schulische Umsetzung ist ein Projekt geeignet, welches das Fach Technik mit der Informatik verbindet. Beim Bau des Gerätes stehen zuerst einmal die technisch-mechanischen Aufgaben im Vordergrund. Exaktes Arbeiten unter Einhaltung der vorgegebenen Toleranzen ist die Grundvoraussetzung für den Bau eines funktionsfähigen Gerätes. Beim Einbau der Elektronik stehen Inhalte aus dem Umfeld Messen-Steuern-Regeln wie z.B. Aufbau, Funktionsweise und softwaremäßige Integration von Sensoren, Aktoren und Verarbeitungseinheiten im Vordergrund. Hier wird an einem technischen System, welches die Schülerinnen und Schüler in seiner Komplexität erfassen können, gezeigt, wie Mechanik und Elektronik eine Einheit bilden. Zur Steuerung des Druckers ist ein Mikrokontroller vom Typ Arduino integriert. An der Programmierung dieses Mikrokontrollers können die Schülerinnen und Schüler die Arbeitsweise eingebetteter Systeme erlernen.

Für die praktische Erprobung wurde ein Velleman 3D-Drucker Bausatz K8200 verwendet. Mit einem Anschaffungspreis von ca. €500 ist er für den schulischen Bereich erschwinglich. Der dem Workshop zu Grunde liegende Aufbau des Bausatzes wurde im Rahmen der Lehramtsausbildung Informatik/Technik erprobt und die mögliche Umsetzung für den Unterrichtseinsatz wurde abgeleitet.

Der Bausatz liegt vollständig in Einzelteilen vor, die Bauanleitung ist eine .pdf-Datei. Da verschiedene Komponenten des Druckers parallel und unabhängig voneinander gebaut werden können, wird folgende mögliche Aufteilung für den Bau vorgeschlagen.

Die Umsetzung erfolgt mit drei Arbeitsgruppen. Die erste Arbeitsgruppe baut die Spulenhalterung, den x-Tisch und den Rahmen. Die zweite Gruppe baut das linke und rechte Profil des Rahmens, den Extruderarm und die z-Achse. Als Aufgabe für die dritte Gruppe bleiben der Bau des Extruders, des Heizbetts und die Vollendung des Rahmens. Als besonders anspruchsvoll hat sich der Bau der Controllerplatine, das Verdrahten und das Einlöten der Bauteile erwiesen. Diese Aufgaben sollte eine sehr leistungsstarke Gruppe oder der Lehrende übernehmen. Als besonders wichtig hat sich auch das Qualitätsmanagement, die Kontrolle der einzelnen Bauschritte, herausgestellt. Die schwierigste Aufgabe im Projekt ist das Justieren des Gesamtsystems und das Einstellen der Parameter. Diese Einstellung ist für die Qualität der späteren Druckergebnisse

entscheidend.

Entsprechend der umzusetzenden Anforderungen wird dieses Projekt ab der Klassenstufe 9 empfohlen. Die Realisierung kann in einer Projektwoche, im Projektunterricht, im Rahmen einer Arbeitsgemeinschaft oder einer Schülerfirma erfolgen.

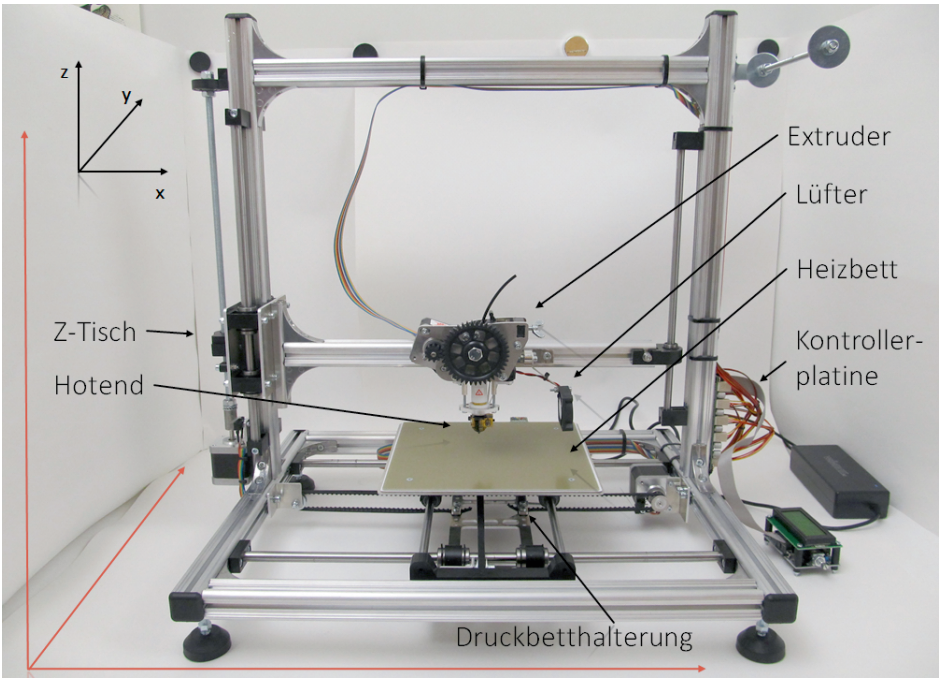


Abb. 1: Aufbau des Druckers

An den Aufbau des Druckers schließt sich die zweite Phase, die Nutzung im Technik- und Informatikunterricht an.

3 Schritte zur Erstellung des 3D-Objektes

Nachdem der Drucker aufgebaut und justiert ist, kann mit dem 3-D-Druck begonnen werden. Die folgenden Schritte zur Anwendung des Druckers werden im Workshop praktisch durchgeführt.

Als erster Schritt zur Erprobung des Druckers können Modelle aus dem Internet geladen und gedruckt werden. Dabei werden die prinzipielle Arbeitsweise und die Grenzen eines 3-D-Druckers für die Schülerinnen und Schüler sichtbar. Der Druck erfolgt, indem geschmolzener Kunststoff linien- und schichtweise miteinander verklebt wird. Daher

lassen sich mit dieser Bauart der 3D-Drucker nur solche Strukturen herstellen, die entsprechend zusammenhängend sind.

Als erster Schritt muss ein 3D-Modell erstellt werden. Im durchgeführten Projekt wurde zur Erstellung des 3D-Modells das Programm SketchUp verwendet. Die Benutzung dieses Programms ist für einfache geometrische Figuren intuitiv erlernbar und für Schülerinnen und Schüler geeignet.

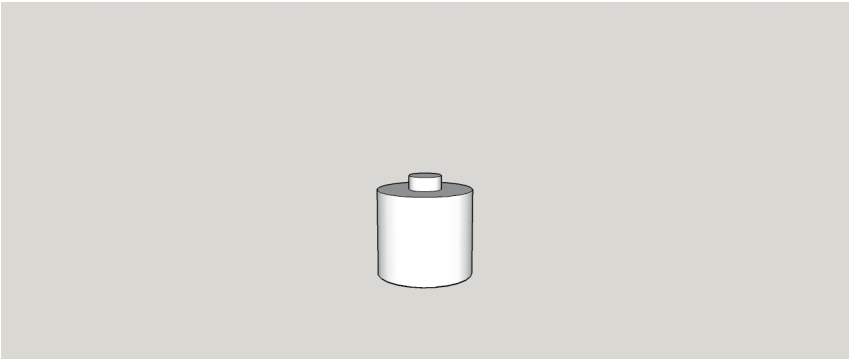


Abb. 2: Erstellung des 3D-Modells mit SketchUp

Die erstellte Datei wird im Stereolithographieformat (.stl) abgespeichert und steht nun für die weitere Verwendung zur Vorbereitung des 3D-Drucks durch die Repetier-Software zur Verfügung. Die Repetier™-Software übernimmt die Teilaufgaben zur Durchführung des Drucks. Dazu gehört die Zerlegung des Modells in Schichten und Linien, die durch die Programmkomponente Slic3r erzeugt wurde.

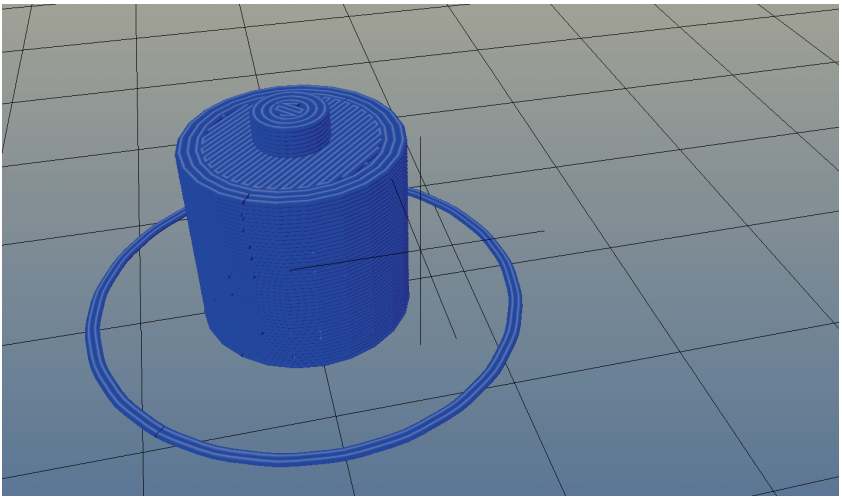


Abb. 3: Erstellung des Schichtenmodells

Die erzeugten Daten werden zur direkten Ansteuerung des Druckers genutzt. Schichtweise wird das Filament (geschmolzener Kunststoff) aufgetragen. Die Bewegungen, die das Modell realisieren, werden über Schrittmotoren vom Druckbett ausgeführt. Die Daten werden von Computer, auf dem das Repetier-Programm läuft, im G-Code an den Arduino-Steuerrechner übertragen. Der Code ist transparent und kann von den Schülerinnen und Schülern direkt analysiert werden. Das Format der zu übermittelnden Steuerinformationen ist in der Norm DIN 66025/ISO 6983 definiert. Dieses Format wird auch für die Steuerung von CNC-Maschinen eingesetzt und verbindet damit Informatikunterricht mit dem Technikunterricht. Die Schülerinnen und Schüler können die einzelnen Ebenen des Modells erkennen und sehen gleichzeitig die Daten, die zur Erstellung dieser Ebene an den Steuerrechner übertragen werden.

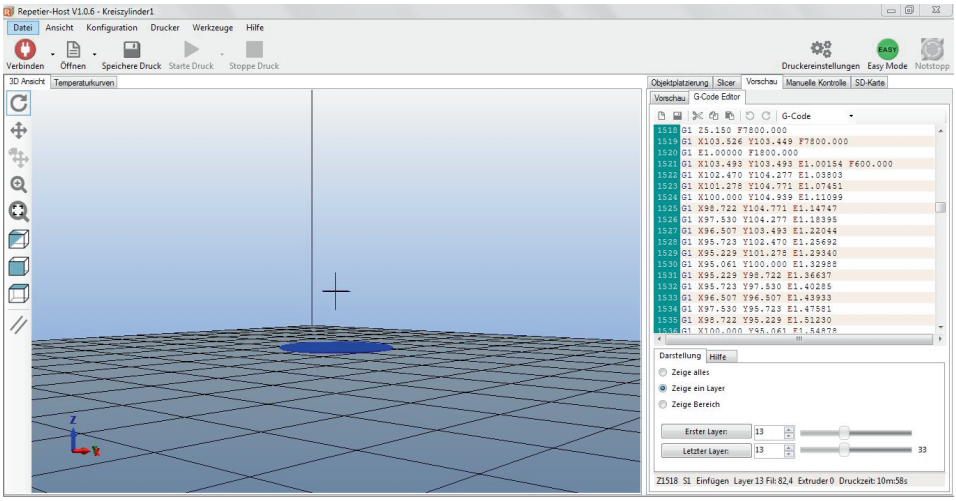


Abb. 4: Zerlegung des Modell und Generieren des G-Codes

Während der eigentlichen Druckphase können die Schülerinnen und Schüler die Abarbeitung der Daten verfolgen.

Anschließend wird das fertige Produkt beurteilt und mit der Zeichnung verglichen. Es werden die Grenzen des eingesetzten Verfahrens deutlich. Strukturen, die kleiner sind, als die Fadenstärke annähern können nicht mehr gedruckt werden. Auch die Oberflächenstruktur wird durch den Aufbau aus dünnen Fäden bestimmt, so dass glatte Oberflächen nicht herstellbar sind.

4 Fazit

Der 3D-Druck, als ein Verfahren des Rapid Prototyping, ist auf Grund der technischen Entwicklung in der Schule einsetzbar. Schüler können für den Technik- und

Informatikunterricht Bauteile selbst erstellen und alle Schritte von der Zeichnung bis zum Produkt selbst gestalten und erleben. Fehler in den einzelnen Arbeitsschritten werden am Produkt sichtbar. Die durch die technischen Parameter gegebenen Einschränkungen werden an den gedruckten Bauteilen sichtbar. Mit der im Projekt verwendetet Technik liegt die maximale Auflösung etwa bei der Erstellung eines Lego-Steins oder eines etwas größeren Zahnrades.

Schüler für Fachthemen interessieren und motivieren - Informatikunterricht im Fokus

Stefanie Jäckel[1]

Abstract: Motivation bestimmt das Gelingen von Lernprozessen entscheidend. Motivierungspotenziale und konkrete Argumente beim Motivieren informatischer Fachinhalte in der Unterrichtspraxis sind bisher ein Desiderat informatikdidaktischer Forschung. Anliegen des Workshops sind die Analyse von sowie reflektierte und kreative Auseinandersetzung mit Motivierungen im Informatikunterricht. In einer Studie ermittelte Typen motivierender Einstiege in verschiedene Inhaltsbereiche der GI-Empfehlungen zu Bildungsstandards Informatik [GI08] werden vorgestellt, an Beispielen diskutiert und Schlussfolgerungen für den aktuellen Stellenwert der Motivierung im Fach Informatik abgeleitet.

Keywords: Motivierung, Informatikunterricht, Unterrichtseinstiege, Lehrerhandeln

1 Motivieren im Informatikunterricht – ein zeitgemäßes Thema?

Produkte der Informatik und deren bereitgestellte Dienste sind allgegenwärtiger Bestandteil der Lebenswelt von Schülerinnen und Schülern. Begegnungen mit Informatiksystemen und deren Wirkungen außerhalb des Informatikunterrichts nehmen täglich zu. Benötigen heutige Schüler diesen „ersten Schubs“ im Fach Informatik überhaupt noch oder sind Motivierungen durch die Informatisierung unserer Lebenswelt nicht bereits längst überholt und dadurch unnötig geworden?

Um dieser zugespitzten Frage nachzugehen und zu ergründen, mit welchen Argumenten Informatikkräfte Fachthemen aktuell motivieren, wurde eine wissenschaftliche Erhebung durchgeführt, deren Vorgehensweise und ausgewählte Ergebnisse im Folgenden vorgestellt werden. Die Untersuchung wird außerdem als Arbeitsgrundlage im Workshop (detailierte Konzeption im Abschnitt 4) genutzt.

[1] Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, Ernst-Abbe-Platz 2, 07743 Jena, stefanie.jaeckel@uni-jena.de

2 Motivierungen im Fach Informatik – Ergebnisse einer unterrichtspraktischen Studie

2.1 Vorgehensweise und Charakterisierung der Untersuchung

Wissenschaftliche Untersuchungen zum Einfluss von Motivation auf den Lernerfolg weisen eindeutig nach, dass die Motivation des Lerners mit seiner Lernleistung und der Verwendung tiefergehender Lernstrategien korreliert (vgl. [Di12, S. 79 f.]). Nicht nur aus pädagogischer und fachdidaktischer Forschung, sondern vor allem aus der Unterrichtspraxis selbst stammt die Erkenntnis, dass effizientes Lehren und Lernen nur möglich ist, wenn Lernende hinreichend für Lernhandlungen motiviert sind (vgl. [DL11, S. 128 f.]).

Um zu ergründen, welches Motivationspotenzial in den einzelnen Themen des Informatikunterrichts steckt und wie die tatsächliche Motivierung von Fachinhalten in der Unterrichtspraxis erfolgt, wurde von der Autorin eine wissenschaftliche Studie durchgeführt. In dieser sollten typische Motivierungen im Fach Informatik offen und explorativ erschlossen werden. Hierfür wurde ein hypothesengenerierender Ansatz benutzt und als Erhebungsmethode ein offener Online-Fragebogen[2] gewählt. Motivierungen wurden über die für Lehrer alltagsnahe Aufgabe der Gestaltung motivierender Einstiege in Unterrichtsthemen[3] erfragt. Anhand charakteristischer Schülertätigkeiten wurden Inhalte und Zielkompetenzen der Unterrichtsthemen beschrieben. Dabei wurde eine inhaltliche Bandbreite im Sinne der GI-Empfehlungen zu Bildungsstandards Informatik[4] [GI08] gewährleistet. Da das Ziel der Untersuchung u.a. darin bestand, Motivierungen aus dem eigenen Unterricht der Teilnehmer analysieren zu können, durften die Befragten für jede Sekundarstufe selbst wählen, zu welchen drei der insgesamt zehn beschriebenen Themen sie sich äußerten. Im Befragungszeitraum von November 2014 bis Januar 2015 beteiligten sich 51 Informatiklehrerinnen (41,2 %) und -lehrer (58,8 %) aus insgesamt 13 Bundesländern. Die Mehrzahl der teilnehmenden Lehrpersonen hat Informatik (84,3 %), Mathematik (56,9 %) oder Physik (19,6 %) studiert und unterrichtet vorwiegend in der Sekundarstufe II (66,7 %).

In der Auswertung wurde analysiert, wie häufig einzelne Themen zur Beantwortung gewählt wurden. Eine Vielzahl der angegebenen Motivierungen (75,9 %) wurde von den Lehrpersonen bereits in ihrem eigenen Unterricht erprobt. Die Antworten der Lehrkräfte wurden mit der Software MAXQDA 10 [VE10] qualitativ untersucht. Aus dem Datenmaterial wurde zunächst durch gewichtete Codierung herausgearbeitet, aus welchen Inhaltsbereichen die angegebenen Motivierungen stammen. Die Ergebnisse dieses Analyseschrittes werden im Abschnitt 2.2 dargestellt. Außerdem wurden durch typologische

[2] Der Onlinefragebogen wurde mittels SoSci Survey [Le14] realisiert.
[3] Themen aus Informatiklehrbüchern sowie dem Thüringer Informatik-Lehrplan [TMB12].
[4] Die Inhaltsbereiche der GI-Empfehlungen wurden als inhaltliches Raster für die Sekundarstufen I und II benutzt. Dieses Vorgehen erscheint zulässig, da durch Fothe die Möglichkeit der Einordnung der EPA-Fachinhalte [KMK04] in die GI-Inhaltsbereiche belegt wurde [Fo08].

Analyse 12 Typen motivierender Einstiege gebildet (Erläuterungen dazu im Abschnitt 2.3).

2.2 Präferierte Inhaltsbereiche der Teilnehmer

Eine Analyse der erhobenen Daten zeigt, dass die angegebenen Motivierungen in ihrer Anzahl je nach Inhaltsbereich stark variieren (siehe Abb. 1). So wurden die meisten Ideen sowohl in der Sekundarstufe I als auch in der Sekundarstufe II für die Inhaltsbereiche „Information und Daten“ und „Algorithmen“ genannt. Interessanterweise ist der Verlauf der auf der Häufigkeit der Angaben beruhenden Kennlinien in beiden Sekundarstufen ähnlich. Es lässt sich vermuten, dass die Teilnehmer für beide Sekundarstufen motivierende Einstiege zu Inhaltsbereichen angegeben haben, in denen sie sich selbst sicher fühlen, gern unterrichten oder bereits häufig oder besonders erfolgreich unterrichtet haben. Andererseits könnte aus der Verteilung der Antworten auch geschlussfolgert werden, dass diese Inhaltsbereiche besonders leicht zu motivieren sind bzw. sich für die von mir angegebenen Themen unkompliziert motivierende Einstiege beschreiben ließen. Die Diskussion im Workshop soll zur Erhärtung dieser Vermutungen beitragen.

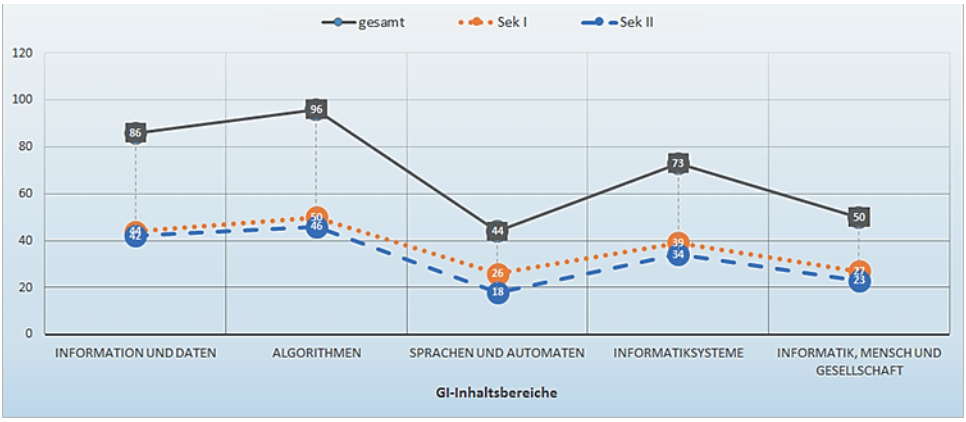


Abb. 1: Häufigkeiten der GI-Inhaltsbereiche, zu denen Motivierungen angegeben wurden

Außerdem kann mit dem erhobenen Datenmaterial gezeigt werden, dass Motivierungen nicht zwangsläufig den Inhaltsbereichen entstammen, die sie motivieren sollen. Besonders deutlich belegt diese Aussage der Inhaltsbereich „Informatik, Mensch und Gesellschaft“.

Für Themen aus diesem Inhaltsbereich wurde in beiden Sekundarstufen die zweitgeringste Anzahl an Motivierungen (insgesamt 5,7 % Differenz zur durchschnittlichen Anzahl) abgegeben. Es konnten jedoch in der Sekundarstufe I 30,9 % (Höchstwert) und in der Sekundarstufe II 25,0 % (zweithöchste Anzahl) der Motivierungen diesem Inhaltsbereich zugeordnet werden. Auffällig ist außerdem der Inhaltsbereich „Informatiksysteme“, aus welchem insgesamt die meisten Motivierungen stammen (vgl. Abb. 2).

Bei der Interpretation der Ergebnisse lässt sich schlussfolgern, dass die befragten Informatiklehrer vor allem mit Bezügen zu Informatiksystemen und zu Wechselwirkungen zwischen Informatik und Gesellschaft motivieren. Mit dieser Erkenntnis kann für die vorliegende Erhebung resümiert werden, dass die Inhaltsbereiche „Informatiksysteme“ und „Informatik, Mensch und Gesellschaft“ besonderes Potenzial zur Motivierung von Schülerinnen und Schülern besitzen.

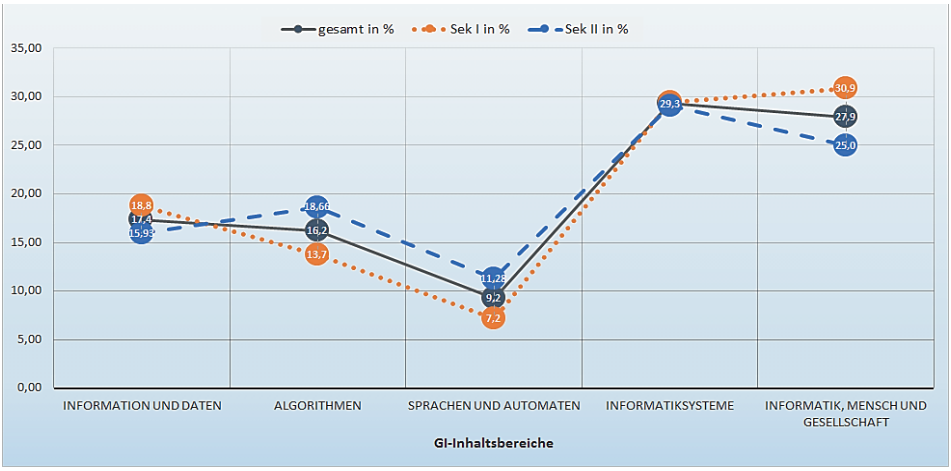


Abb. 2: Häufigkeiten der in den Antworten herangezogenen GI-Inhaltsbereiche

2.3 Typen motivierender Einstiege im Informatikunterricht

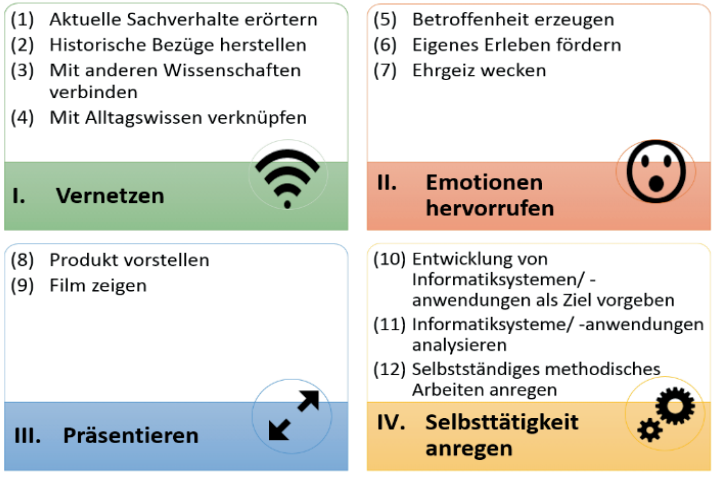


Abb. 3: 12 Typen motivierender Einstiege und deren Zuordnung zu vier Großformen

Mit dem Verfahren der typologischen Analyse nach Kuckartz [Ku09, S. 99], das es ermöglicht, aus größeren Datenmengen typische Aspekte herauszufiltern und weitergehend zu untersuchen, wurde eine Typenbildung motivierender Einstiege vorgenommen. Auch hier erfolgte die Zuordnung der Antworten durch gewichtete Codierung. Die auf diese Weise ermittelten 12 Typen von Motivierungen lassen sich wiederum in vier Großformen einordnen (siehe Abb. 3).

Die herausgearbeiteten Typen motivierender Einstiege werden im Folgenden durch ihre Eigenschaften und damit verbundenen Ziele charakterisiert. Sie weisen auch Bezüge zu allgemeinpädagogischen Untersuchungen [Sc04, S. 138 f.] und mathematikdidaktischen Empfehlungen [Po91, S. 175 ff.] auf.

I. Vernetzen

(1) Aktuelle Sachverhalte erörtern

Motivierungen dieses Typs thematisieren aktuelle Geschehnisse oder Probleme. Die Sachverhalte entstammen entweder der direkten Lebenswelt der Schüler (bspw. aus dem schulischen Umfeld) oder der Nachrichten- und Berichterstattung wie dem TV, Online- oder Printmedien. Durch das Erörtern realer Sachverhalte unter bestimmten Aspekten werden das Bewusstsein und Verständnis für Probleme und für aktuelle Mediendiskurse angebahnt.

(2) Historische Bezüge herstellen

Motivierungen dieses Typs beziehen sich auf die Geschichte der Informatik, Rechentechnik oder Mathematik. Diese Verbindungen werden über interessante Persönlichkeiten, Quellenmaterial oder Geschichtserzählungen hergestellt. Historische Problemstellungen werden nachempfunden, Vergleiche zwischen historischen und aktuellen Technologien oder Entwicklungen angestrebt und Methoden und Einstellungen im dynamischen Prozess der Wissenschaft Informatik untersucht. Historische Problemstellungen lassen die wissenschaftliche oder gesellschaftliche Bedingtheit von Lösungsansätzen, Meinungen und Einstellungen zur Informatik im Wandel der Zeit erkennen.

(3) Mit anderen Wissenschaften verbinden

Motivierungen dieses Typs thematisieren Querbezüge zu anderen Wissenschaften direkt oder stellen Bezüge zu anderen Schulfächern und deren Arbeitsmethoden bewusst her. Hierdurch wird die Relevanz der Thematik für weitere Bereiche in Wissenschaft und Unterricht deutlich. Weiterhin kann durch eine fachfremde Herangehensweise ein Perspektivwechsel und anderer Blick auf den Inhalt erfolgen. Durch das Vernetzen von Informatikthemen mit anderen Wissenschaften können Interessen der Schüler auf breiterem Gebiet angesprochen werden.

(4) Mit Alltagswissen verknüpfen

Motivierungen dieses Typs nutzen Bezüge zu eigenen Tätigkeiten und Situationen im Alltag und zu Daten oder Eigenschaften der Lebenspraxis. Hierbei wird das Vorwissen über vertraute, aber häufig nicht triviale Probleme der Alltagswelt (wie z.B. der eigenen Sprache) aufgegriffen oder überraschend eingesetzt. Die Arbeit an

authentischen Beispielen aus dem Alltag verdeutlicht die Nützlichkeit der Thematik für diesen und trägt dazu bei, die eigene Lebenswelt beser zu verstehen.

II. Emotionen hervorrufen

(5) Betroffenheit erzeugen

Motivierungen dieses Typs lassen durch einen Sachverhalt oder eine Problemstellung aus passiven Beteiligten direkt betroffene Akteure werden. Die durch eigene Vorerfahrungen bestehenden Erwartungen der Schüler werden erfüllt oder absichtlich enttäuscht. Durch die eigene Betroffenheit wird die Bedeutung des Sachverhalts gesteigert und es entsteht das Bedürfnis, sich mit dem Problem bzw. Sachverhalt auseinanderzusetzen.

(6) Eigenes Erleben fördern

Durch Motivierungen dieses Typs werden Schüler beim eigenen körperlichen Tun in Rollenspielen, szenischen Darstellungsformen oder Experimenten zum Akteur. Die Lerner schlüpfen in verschiedene Rollen, versetzen sich in andere Sichtweisen und spielen diese nach. In diesen real erlebbaren Situationen werden Vorstellungskraft, Fantasie, Kreativität und Einfühlungsvermögen aktiviert.

(7) Ehrgeiz wecken

Motivierungen dieses Typs erfolgen in Quiz- oder Rätselform, durch kniffelige Aufgaben oder im Wettbewerbscharakter. Alter und Vorerfahrung der Schüler bestimmen Art und Schwierigkeit dieser Lern- und Denkspiele. Durch den kognitiven Charakter der Arbeitsaufträge können neues Wissen und neue Kompetenzen erworben und das Spielbedürfnis der Schüler befriedigt werden. Ein initiierter Wettbewerb unter den Lernern, konkrete Zeitvorgaben oder die Aussicht auf Anerkennung und Belohnung können zusätzlich den Ehrgeiz der Beteiligten wecken.

III. Präsentieren

(8) Produkt vorstellen

Motivierungen dieses Typs stellen ein digitales oder physisches Produkt vor. Die Produkte werden vom Lehrer oder Schüler selbst mitgebracht und präsentiert. Digitale Produkte können beispielsweise Programme, Animationen oder Online-Spiele sein. Mit ihnen werden Eigenschaften und Funktionalitäten anschaulich präsentiert. Reale Gegenstände wie Brettspiele, Modelle, technische Geräte und Dinge des täglichen Gebrauchs können Schüler von allen Seiten betrachten, erfühlen und erforschen.

(9) Film zeigen

Motivierungen dieses Typs werden mit Hilfe von Filmen, Filmsequenzen, Online-Videoclips oder didaktisch aufbereiteten Unterrichtsfilmen erzeugt. Das Medium Film ist für Schüler eine aus dem Freizeitbereich vertraute Art der Informationsdarstellung. Authentische Bilder und visualisierte Handlungen ermöglichen es, Inhalte zunächst gedanklich und emotional wirken zu lassen. Filmszenen können durch ihre hohe Intensität und Ausdrucksstärke beim Lerner besonderes Interesse wecken. Unterrichtsfilme werden didaktisch speziell auf das Zielpublikum abgestimmt und für dieses produziert.

IV. Selbsttätigkeit anregen

(10) Entwicklung von Informatiksystemen/ -anwendungen als Ziel vorgeben

Motivierungen dieses Typs geben das Ziel vor, Informatiksysteme, einzelne Bestandteile oder Informatikanwendungen selbst zu entwickeln. Hierfür werden Hard- und/oder Softwaresysteme um einzelne Komponenten oder ganze Funktionalitäten erweitert oder noch nicht fertige Anwendungen vervollständigt. Es werden Anwendungsprogramme erstellt, die man in der Realität benötigt oder reale Informatiksysteme (in Teilen) didaktisch nachgebildet, wodurch beim Schüler ein tieferes Verständnis erzielt werden kann. Mittels eigener aktiver Tätigkeit entsteht ein Produkt, welches in der Realität nützlich ist und Anwendung finden kann.

(11) Informatiksysteme/ -anwendungen analysieren

Motivierungen dieses Typs analysieren reale Informatiksysteme bzw. deren Komponenten, wodurch erste Erklärungen für Funktionsweisen realer und häufig komplexer Systeme deutlich werden. Außerdem können technische Geräte miteinander verglichen werden. Modelle von Informatiksystemen werden zur Analysetätigkeit genutzt oder durch sie erstellt.

(12) Selbstständiges methodisches Arbeiten anregen

Motivierungen dieses Typs regen zum selbständigen Erschließen von Inhalten durch eigene Arbeitstätigkeit an. Methoden und Arbeitsschritte werden hierbei entweder vorgegeben oder dem Lerner selbst überlassen. Die Schüler erschließen sich durch diese Herangehensweise Inhalte selbst, entdecken Zusammenhänge und entwickeln eigene Strategien zum Lösen von Problemen.

Die charakterisierten Typen werden im folgenden Abschnitt anhand von Beispielen spezifiziert. Dies erfolgt exemplarisch durch je fünf Antworten für die Typen: „Mit Alltagswissen verknüpfen“, „Eigenes Erleben fördern“ und „Produkt vorstellen“.

	***Inhaltsbereich** mit Thema*		**Beispielaussage**
a)	***Information und Daten (S II)*** *Objektorientierte Datenmodellierung*	⇒	Ein „Museum der Objekte“ wird erstellt, indem elektrische Gegenstände (Lampen, Bohrmaschinen, Wecker usw.) auf Tischen verteilt und mit Steckbriefen versehen werden.
b)	***Algorithmen (S I)*** *Handlungsvorschriften für das Arbeiten mit Informatiksystemen*	⇒	Die Schülerinnen und Schüler beschreiben Handlungsvorschriften aus dem Alltag, wie Zähneputzen, Brötchen schmieren oder spezielle Mahlzeiten zubereiten.
c)	***Sprachen und Automaten (S II)*** *Zustandsorientierte Modellierung endlicher Automaten*	⇒	Die Funktionalität geeigneter Alltagsgegenstände (z.B. Stirnlampe, Stoppuhr) wird analysiert, verbal erläutert und in Diagrammen und Tabellen formalisiert.
d)	***Informatiksysteme (S II)*** *Charakterisierung von Computernetzen*	⇒	Gemeinsam wird untersucht, warum eine Internetseite auf verschiedenen Endgeräten lesbar ist.
e)	***Informatik, Mensch und Gesellschaft (S I)*** *Automatisierung*	⇒	Den Schülern wird ein Bild des Container-Terminals im Hamburger Hafen unter der Überschrift „Menschenfreie Zone“ gezeigt.

Beispiel 1: Motivierung (4) „Mit Alltagswissen verknüpfen“

	Inhaltsbereich *mit Thema*		**Beispielaussage**
a)	***Information und Daten (S II)*** *Binäre Codierung als Prinzip der Informationsverarbeitung*	⇒	Die Schüler erleben Dualzahlen selbst, indem ihnen nur zwei Zustände zum Zählen mit den Fingern ermöglicht werden.
b)	***Algorithmen (S I)*** *Handlungsvorschriften für das Arbeiten mit Informatiksystemen*	⇒	Die Schüler steuern ihre Mitschüler „als Roboter“ auf dem Schulhof.
c)	***Sprachen und Automaten (S II)*** *Formale Sprachen und Grammatiken*	⇒	Es werden den Schülern einfache Regeln für das Bilden von Wörtern vorgegeben. Die Schüler sollen sich mit erlaubten Wörtern unterhalten.
d)	***Informatiksysteme (S II)*** *Charakterisierung von Computernetzen*	⇒	Spiel auf dem Schulhof mit Datenleitungen (Fäden), unterschiedlichen Nachrichten (Läufern) und sprechenden Geräten (Schülern).
e)	***Informatik, Mensch und Gesellschaft (S II)*** *Schutz personenbezogener Daten*	⇒	Die Schüler spielen in Szenen den Besuch eines Aids-Patienten in einer Arztpraxis nach.

Beispiel 2: Motivierung (6) „Eigenes Erleben fördern“

	Inhaltsbereich *mit Thema*		**Beispielaussage**
a)	***Information und Daten (S II)*** *Objektorientierte Datenmodellierung*	⇒	Verschiedene Spielzeugautos werden mit in den Unterricht gebracht, um an ihnen Begriffe der Objektorientierung abzuleiten.
b)	***Algorithmen (S II)*** *Iterative und rekursive Sortier- und Suchverfahren*	⇒	Die Schüler arbeiten mit einer Software, die die Korrektheit einer nicht zu großen sortierten Zahlenfolge anzeigt. Die Zahlen bleiben aber verborgen, sodass die Lerner in die Situation des Computers versetzt werden.
c)	***Sprachen und Automaten (S I)*** *Analyse von Automaten*	⇒	Ein mitgebrachter Kaugummi-Automat wird den Schülern präsentiert.
d)	***Informatiksysteme (S II)*** *Arbeitsweise des PC auf Grundlage des von-Neumann-Rechnermodells*	⇒	Anhand des Modellprogramms ALI werden die grundlegenden Arbeitsschritte der von-Neumann-Rechner-Architektur simuliert.
e)	***Informatik, Mensch und Gesellschaft (S I)*** *Urheberrecht*	⇒	Die Schüler bringen eine kopierte CD oder DVD mit, an welcher Software-Kopien aus dem Netz thematisiert werden.

Beispiel 3: Motivierung (8) „Produkt vorstellen“

3 Motivieren im Informatikunterricht – vielfältig & vielbedeutend!

Durch die zunehmende Informatisierung unserer Lebenswelt begegnen Schüler Wirkungen der Informatik vor allem außerhalb des Informatikunterrichts.

Um mit diesen immer vielschichtiger werdenden Zusammenhängen im Alltag bewusst und verantwortungsbewusst umgehen zu lernen, werden informatische Kompetenzen benötigt, welche nur an anspruchsvollen Fachinhalten ausgebildet werden können. Informatikunterricht hat diesen Beitrag zur Allgemeinbildung zu leisten, worauf Bethge und Fothe mit der entwickelten Konzeption der Grunderfahrungen im Informatikunterricht [BF13] hinweisen. So lautet die Grunderfahrung 1: *„Informatikunterricht ist dadurch allgemeinbildend, dass er […] ermöglicht, Informatiksysteme und ihre Wirkungen in unterschiedlichen Lebensbereichen zu entdecken, zu verstehen und zu bewerten“.* [BF13, S. 116 f.]
Diese Grunderfahrung verdeutlicht, wie wichtig es aktuell ist, Jugendlichen beim Entdecken von Informatiksystemen zu unterstützen und sie zu motivieren, diese Systeme auch verstehen zu wollen. Nur so können Entwicklungen der Informatik von mündigen Bürgern bewertet und sinnstiftend genutzt werden.
Die Ergebnisse der durchgeführten Studie zeigen, dass Lehrerinnen und Lehrer im Informatikunterricht auf vielfältige und kreative Art und Weise motivieren und „der erste Schubs beim Schaukeln“ im Informatikunterricht keineswegs vernachlässigt wird. Besonders häufig wurde von den Lehrkräften das Potenzial der Inhaltsbereiche „Informatiksysteme“ und „Informatik, Mensch und Gesellschaft“ genutzt und mit deren Hilfe motiviert. Diese Motivierungen stehen im engen Zusammenhang mit der Grunderfahrung 1.

4 Ziele und Vorgehen im Workshop

Nach der Abstimmung zu Zielen und Verlauf des Workshops werden einführend allgemeinpädagogische und fachdidaktische Grundlagen zu Lernmotivation und Interesse sowie Ergebnisse der durchgeführten Studie präsentiert. Auf dieser Theoriebasis werden inhaltlich oder methodisch verschiedene motivierende Einstiege zu unterrichtsrelevanten Themenbereichen in Arbeitsgruppen entwickelt und eine Einordnung in das vorgestellte Kategoriensystem angestrebt. Die Arbeitsergebnisse werden im Plenum analysiert, besonders motivierende Beispiele favorisiert und in Beziehung zu Tendenzen und Ergebnissen aus der Studie gesetzt. Sollte es die Zeit zulassen, können eigene Entscheidungsgrundlagen bei der Auswahl und Planung eines Unterrichtseinstiegs sowie Kennzeichen motivierender Einstiege reflektiert werden. Abschließend wird eine Diskussion zu Zielen und Einsatzmöglichkeiten der entwickelten Unterrichtseinstiege initiiert, deren Ergebnisse auch in weitere Forschungsaktivitäten einfließen können. Dank der vorgestellten Materialien und der Präsentation der Arbeitsergebnisse wird den Teilnehmern ein „breiter Fundus” an verschiedenen Ideen zur Motivierung informatischer Fachinhalte bereitgestellt. Die gemeinsame Arbeit im Workshop lässt Intentionen und Perspektiven anderer Fachkollegen deutlich werden, mit welchen eigene unterrichtspraktische Konzepte weiterentwickelt werden können. Insgesamt soll der Workshop sowohl zur fachdidaktischen Analyse konkreter Unterrichtshandlungen beitragen als auch Informatiklehrkräften durch Ergebnisse und Materialien der Fachdidaktik-Forschung bei der Vorbereitung und Effektivierung ihres Unterrichts unterstützen.

Literaturverzeichnis

[BF13] Bethge, B.; Fothe, M.: Grunderfahrungen des Informatikunterrichts - ein Beitrag zur Frage der Allgemeinbildung von Informatik. In (Breier, N.; Stechert, P.; Wilke, T. Hrsg.): INFOS 2013. 15. GI-Fachtagung „Informatik und Schule“. Praxisband, Kiel, 2013; S. 113–121.

[Di12] Ding, K.: Wie motiviere ich im Unterricht? Ein praxisorientiertes Handbuch zum Motivationsprinzip. Brigg, Augsburg, 2012.

[DL11] Dresel, M.; Lämmle, L.: Motivation. Maßnahmen zur Herstellung und Förderung der Lern- und Leistungsmotivation von Schülerinnen und Schülern. In (Götz, T. Hrsg.): Emotion, Motivation und selbstreguliertes Lernen. Schöningh, Paderborn, München, Wien, Zürich, 2011; S. 128–137.

[Fo08] Fothe, M.: Bildungsstandards Informatik für die Sekundarstufe II - Vorüberlegungen zur Entwicklung. In (Torsten Brinda; Michael Fothe; Peter Hubwieser Hrsg.): Didaktik der Informatik - Aktuelle Forschungsergebnisse. 5. Workshop der GI-Fachgruppe "Didaktik der Informatik" 24.-25.09.2008 an der Universität Erlangen-Nürnberg. GI, 2008; S. 107–117.

[GI08] GI (Gesellschaft für Informatik) e. V. Hrsg.: Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. LOG IN Verlag GmbH, Berlin, 2008.

[KMK04] Kultusministerkonferenz: Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik. Beschluss vom 1.12.1989 i. d. F. vom 5.2.2004, Luchterhand, 2004.

[Ku09] Kuckartz, U.: Einführung in die computergestützte Analyse qualitativer Daten. VS Verlag für Sozialwissenschaften, 2009.

[Le14] Leiner, D. J.: SoSci Survey. SoSci Survey GmbH, München, 2014.

[Po91] Posamentier, A. S.: Motivation im Mathematikunterricht: Eine vernachlässigte Kunst. In Schriftenreihe zur Didaktik der Mathematik der Österreichischen Mathematischen Gesellschaft (ÖMG), 1991; S. 174–188.

[Sc04] Schiefele, U.: Förderung von Interessen. In (Lauth, G. W.; Grünke, M.; Brunstein, J. C. Hrsg.): Interventionen bei Lernstörungen. Förderung, Training und Therapie in der Praxis. Hogrefe, Göttingen, 2004; S. 134–144.

[VE10] VERBI GmbH: MAXQDA, Berlin, 2010.

Wahlverhalten zum Schulfach Informatik in der SI - eine Studie im Regierungsbezirk Münster

Irina Janzen[1], Marco Thomas[2] und Angélica Yomayuza[3]

Abstract: Seit mehr als 40 Jahren etabliert sich das Fach Informatik in Nordrhein-Westfalen, doch die strukturellen Rahmenbedingungen machen Informatik für viele Schülerinnen und Schüler wenig attraktiv. Die Erwartungen und Erfahrungen im Hinblick auf Informatikunterricht schrecken scheinbar vor einer Wahl von Informatik ab. Den schon länger vermuteten Zusammenhängen möchte die Studie nachgehen, denn empirische Daten zu Anforderungen für Informatikunterricht von Schülerinnen und Schülern existieren kaum. Im Herbst/Winter 2013 und 2014 wurde eine Umfrage unter ca. 2400 Schülerinnen und Schülern der Sekundarstufe I in Informatik- und Nicht-Informatik-Kursen durchgeführt.

Keywords: Informatikunterricht, Informatikdidaktik, Schule, Schülerverhalten, Datenerhebung

1 Informatische Bildung in der Sek. I NRW

Im größten Bundesland Deutschlands, Nordrhein-Westfalen (NRW), hat das Schulfach Informatik als „Hauptfach“ einer Informatischen Bildung eine lange Tradition ([Kn14]). Mit einem Pilotprojekt in den 1970er Jahren in der gymnasialen Oberstufe (GOSt) an einer Gesamtschule in Gelsenkirchen (Regierungsbezirk Münster) startete in Deutschland die curriculare Etablierung der Informatik; zunächst in den GOSt und anschließend im Wahlpflichtbereich der Sekundarstufe I. In einigen Bundesländern konnte Informatik mittlerweile als Pflichtfach im Schulkanon integriert werden. In NRW ist Informatik ein Wahlpflichtfach geblieben, so dass viele Schülerinnen und Schüler (SuS) keine grundständigen informatischen Kompetenzen in der Schule erwerben.

Verblüffend sind jedes Jahr die Ergebnisse der amtlichen Schulstatistiken in NRW, die ein umfassendes Angebot zur Informatik an den nordrhein-westfälischen Schulen suggerieren. Im Schuljahr 2011/12 ([NR11]) soll das Fach Informatik an 78% der Hauptschulen, 95% der Realschulen, 82% der Gesamtschulen (inkl. GOST) und 94% der Gymnasien (inkl. GOST) und 41% der Berufskollegs angeboten worden sein. Desweiteren ergibt sich aus der offiziellen Statistik, dass im Schuljahr 2011/12 über die Klassen 5 bis 10 an Hauptschulen ca. 21%, aller SuS am Unterrichtsfach Informatik

[1] Westfälische Wilhelms-Universität Münster , Didaktik der Informatik, Fliednerstrasse 21, 48149 Münster, irina.janzen@uni-muenster.de
[2] Westfälische Wilhelms-Universität Münster , Didaktik der Informatik, Fliednerstrasse 21, 48149 Münster, marco.thomas@uni-muenster.de
[3] Westfälische Wilhelms-Universität Münster , Didaktik der Informatik, Fliednerstrasse 21, 48149 Münster, angelica.yomayuza@uni-muenster.de

teilgenommen haben, an Realschulen ca. 22%, an Gesamtschulen und an Gymnasien je 7% (s. Tab. 1).

	Schulen		Teilnehmer		darunter in Jgst.							
	absolut	in%	absolut	in %	5	6	7	8	9	10	Sonstige	GOSt
Hauptschule	472	78	38686	22	2104	3240	8130	7984	8972	5724	2532	
Realschule	534	95	72622	24	8433	7706	16079	14194	12637	9757	3816	
Gesamtschule	191	82	25641	54	2637	1452	2869	3161	4070	3129	4209	4114
Gymnasium	588	94	87675	34	8000	2812	5485	14211	13567	10	4616	38974

Tab. 1: Informatikangebot und Teilnehmerzahlen 2011/12 nach [NR11]

Diese offiziellen Daten widersprechen nicht nur unseren Erfahrungen[4], sondern verblüffen regelrecht: beispielsweise wird Informatik als Schulfach frühestens ab Klasse 7 in NRW angeboten bzw. für diese Jgst. existieren entsprechende Lehrpläne für ein Wahlpflichtfach. Schulinterne Lehrpläne – an manchen Schulen bereits ab Klasse 5 – und die neuen Kernlehrpläne für das Wahlpflichtfach Informatik tragen sicherlich zu einer – auch statistischen – Stärkung der Informatik an den Schulen bei; die Probleme eines „nur" Wahlpflichtfaches lassen sich jedoch nicht kaschieren.

Im Wahlpflichtbereich steht Informatik in NRW stets in starker Konkurrenz zu fremdsprachlichen Angeboten und Schulfächern, die die SuS bereits im Pflichtbereich kennengelernt haben und dann vertiefen können. Erfahrungen zu Informatik haben SuS vorab i.d.R. nur außerschulisch erwerben können, was zu einer verstärkten Heterogenität im Wahlpflichtfach Informatik führt. Insbesondere in der Oberstufe ist das im Vergleich zu anderen Fächern erhöhte Abwahlverhalten von SuS vermutlich durch eine starke Heterogenität der Kurse, Fehlvorstellungen zum Fach Informatik und ungünstige Rahmenbedingungen bedingt[5].

Eine Analyse der Gründe von SuS zu der Anwahl, der Nicht-Anwahl und der Abwahl von Informatikkursen ist ein Schwerpunkt unseres Projekts KISS (Kriterien zum Informatikunterricht von Schülerinnen und Schülern). Mit KISS soll unter anderem die Situation des Wahlpflichtfachs Informatik (zunächst regional begrenzt auf den Regierungsbezirk Münster) präziser und treffender erfasst werden, um die Schulinformatik besser fördern zu können. Die in diesem Artikel beschriebene Studie bestätigt (teilweise schon lange) postulierte Erkenntnisse aus Praxis- und Erlebnisberichten und zeigt neue Zusammenhänge auf.

[4] Es gelang uns bisher nicht zu klären, inwieweit das statistische Erhebungsverfahren und andere Faktoren die Daten für NRW beeinflussen, aber wir werden in Kürze eine Umfrage unter Schulleitern durchführen, da deren Meldungen an die Schulaufsicht offenbar die Grundlage für diese Daten bilden.

[5] Im Schuljahr 2011/12 lag der Rückgang der SuS-Zahlen in Informatik beim Übergang von der 11. in die 12. Jahrgangsstufe am G9-Gymnasium bei gut 45%, während im Fach Mathematik ein Rückgang von ca. 30% und in den Naturwissenschaften von ca. 40% erfolgte. Auch vergrößert sich das Verhältnis von männlichen zu weiblichen Teilnehmern in Informatik in höheren Jahrgangsstufen.

2 Ziele der Studie

Informatische Methoden zur Digitalisierung sind in unserer Welt allgegenwärtig, auch wenn die Auswirkungen der Informatik oft nicht direkt sichtbar sind. Diese Allgegenwart und die Universalität von Informatiksystemen erfordern Informatische Bildung. Doch wie können sich SuS für ein Fach Informatik interessieren, und gegebenenfalls einen entsprechenden Beruf wählen, wenn sie zuvor keine Möglichkeit zur Auseinandersetzung mit dem Fach hatten? Diese Frage spielt gerade bei der Wahl eines Faches für den Wahlpflichtbereich eine Rolle – aus Sicht der SuS, aber auch aus der Perspektive der Schulen bzw. der Lehrenden[6]. Welche Faktoren beeinflussen die Wahl der SuS? Was erwarten SuS von einem Fach Informatik? Welche Unterschiede lassen sich zwischen Schülerinnen und Schülern hinsichtlich der Fächerwahl feststellen?

Für die Jahrgangsstufe 11 (damals das erste Jahr der GOSt) haben unter anderem Magenheim/Schulte ([MS05]) und Engbring ([En14]) das Wahlverhalten von SuS untersucht. Magenheim und Schulte führten eine NRW-übergreifende Studie mit 570 SuS durch. Die Autoren kritisierten fehlende aktuelle empirische Daten und untersuchten aus Interviews gewonnene Hypothesen zum Informatikanfangsunterricht. Engbring hat 2011/12 an drei Schulen in NRW die SuS in der Einführungsphase der GOSt gefragt, welche Fächer sie aktuell belegen, welche sie gewählt hätten und ob sie Informatik weiterbelegen werden. Sicherlich gibt es einzelne weitere Studien zum Wahlverhalten (in NRW), die wir im Rahmen von KISS zusammentragen möchten, wobei detaillierte Informationen zum Untersuchungsdesign (Fragebögen etc.) und die ausgewerteten Daten zugänglich gemacht werden sollten, um eine fundierte Diskussion zur Situation des Wahlpflichtfachs Informatik zu ermöglichen

Unsere Studien konzentrieren sich derzeit auf die Sekundarstufe I, hier auf die Klassenstufen, in denen die SuS erstmalig Informatik belegen können (z. B. am Gymnasium die 8ten Klassen). Lang gehegte Vermutungen sollen bestätigt werden und neue Zusammenhänge möglicherweise aufgedeckt werden, wobei die Fragestellungen durchaus vergleichbar zu denen ähnlicher Studien bzw. zu ähnlichen Problemen des Faches Informatik in der Oberstufe sind.

3 Untersuchungsmethodik

Die Erhebung der Daten erfolgte quantitativ mit einem standardisierten Fragebogen, u.a. unter Verwendung 5-stufiger Skalen (zzgl. zweier offener Fragen), wobei zu berücksichtigen war, dass die Bearbeitungszeit für SuS 15 Minuten nicht überschreiten

[6] Allen teilnehmenden Schulen wurde eine schulspezifische Auswertung der Daten angeboten, was durchaus auf Interesse stieß.

sollte, um den Ausfall von Unterrichtszeit gering zu halten[7]. Der Fragebogen wurde sowohl online als auch als vierseitige Papierversion angeboten[8]. Auf diese Weise konnte eine höhere Rücklaufquote erwartet werden.

Die Umfrage erfolgte jeweils in den Monaten Oktober bis Januar in den Schuljahren 2013/14 und 2014/15. Die digitalen oder papiergebundenen Fragebögen sollten unter der Aufsicht der Fachlehrerin oder des Fachlehrers von allen SuS bearbeitet werden, die im Jahr zuvor erstmals das Fach Informatik gewählt haben oder hätten wählen können.

Von 75 Schülern aus zwei Schulklassen bekamen wir Texte, in denen sie Gründe für die Wahl bzw. Nicht-Wahl von Informatik niedergeschrieben hatten. Diese Information diente als eine Grundlage zur Formulierung und Auswahl von Fragen. Die Auswahl der Items orientiert sich insbesondere an gängigen Fragen, vor allem aus dem Instrument zur Computerbildung INCOBI ([RNG01])[9], jedoch musste die Anzahl der Variablen stark reduziert werden, damit die vorgesehene Bearbeitungszeit eingehalten werden konnte. Formulierungen wurden an das Sprachniveau von SuS angepasst und Items im Hinblick auf das Ziel der Studie fokussiert oder ergänzt. Sowohl der Fragebogen für das Jahr 2013/14 als auch der überarbeitete Fragebogen für 2014/15 wurden von Experten vorab begutachtet. Trotzdem kam es in der ersten Umfrage 2013/14 möglicherweise zu Missverständnissen (vgl. [TY14]), so dass wir uns zu einer Überarbeitung des Fragebogens für 2014/15 entschlossen haben[10]. In Freitexten konnten SuS ihre Angaben ergänzen.

4 Datenerhebung

Gegenstand der Untersuchung sind SuS in der Sekundarstufe I in NRW, die an Realschulen, Gesamtschulen und Gymnasien, im vorausgegangenen Schuljahr das Fach Informatik für das aktuelle Schuljahr neu anwählen konnten (dies sind z. B. am Gymnasium alle Klassen 8), also auch SuS, die Informatik nicht gewählt haben. Wir haben die Population auf den Regierungsbezirk Münster eingeschränkt, sind jedoch der Ansicht, dass die Ergebnisse auf andere Bezirke in NRW übertragbar sind.

Im Schuljahr 2013/14 wurden insgesamt 1366 SuS (w: 47%) an 25 Schulen befragt, im Schuljahr 2014/15 haben sich 997 SuS (w: 45%) an 21 Schulen beteiligt, davon 733 online und 264 per Papierbogen. Von den befragten SuS haben 40% Informatik als

[7] Trotzdem meldeten einige Schulen zurück, dass Zeitmangel oder organisatorische Probleme die Durchführung der Umfrage verhindert haben. Es gab auch Rückmeldungen von Schulen, dass die zuständigen Fachlehrerinnen und Fachlehrer die Information über die Umfrage nicht erhalten haben.

[8] Ca. 1/3 der Schulen nutzten die papiergebundenen Fragebögen.

[9] INCOBI enthält Skalen zur Erfassung von Aspekten einer Computer Literacy, von deklarativem (theoretischem) und prozeduralem (praktischem) Computerwissen, einer Selbsteinschätzung der Vertrautheit mit Computeranwendungen (VECA) und zur Sicherheit imUmgang mit Computeranwendungen (SUCA). Der Einsatz des Erhebungsinstruments ist vor allem für Studenten der Geistes- und Sozialwissenschaften vorgesehen.

[10] Zu dem zweiten Fragebogen haben wir von SuS sogar lobende Worte erhalten.

Erstwunsch bei der Wahl angegeben und 44% der befragten SuS haben das Fach Informatik belegt.

Über die Bezirksregierung Münster wurden die Schulen angeschrieben und zur Teilnahme an der Umfrage eingeladen. Unter den Schulen, die tatsächlich teilgenommen haben, sind nur Gymnasien und Realschulen[11]. Insofern ist die erzielte Stichprobe statistisch „nur" für eine Population von SuS an Realschulen und Gymnasien aussagekräftig.

5 Ergebnisse und Auswertung

Es ist zunächst auffällig, dass keine Informatikkurse aus den Klassen 5 und 6 an unseren Umfragen teilgenommen haben. Wir interpretieren dies so, dass an den Schulen – im Gegensatz zur Schulstatistik des Ministeriums (vgl. Tab. 1) – kein Fach Informatik angeboten wird. Denkbar ist aber auch, dass in diesen Klassenstufen Informatik (schulintern) verpflichtend angeboten wurde und damit die Kurse für die Umfrage als nicht relevant angesehen wurden[12].

Die Unterschiede in den Fragen(formulierungen), die in den beiden Erhebungen auftreten, können an dieser Stelle nicht umfassend berücksichtigt werden. Wir stellen daher im Folgenden die Ergebnisse bezogen auf die Erhebung im Schuljahr 2014/15 dar und werden nur bei relevanten auffälligen Abweichungen zur vorausgegangenen Erhebung auf diese eingehen. Es ist stets zu berücksichtigen, dass die Daten „nur" auf den Angaben der SuS beruhen und (bisher) keine weitergehenden Studien durchgeführt wurden.

Unsere Untersuchungen ergeben für alle befragten SuS, also auch die, die Informatik nicht gewählt haben:

- SuS sehen ihre Wahl eher nicht von anderen Personen beeinflusst. Den Einfluss von Fachlehrern sehen 82% der SuS als eher gering[13] an, ähnliches gilt für Geschwister (81%), Eltern (64%) und andere Schüler (66%).

- Die meisten SuS sind der Ansicht, dass sie wussten, was in den Fächern unterrichtet wird (69%). Bei den erwarteten Tätigkeiten im Informatikunterricht dominieren einerseits das Arbeiten mit Office-Produkten wie Textverarbeitung, Tabellenkalkulation und Präsentationen und andererseits das Programmieren von

[11] Für den Regierungsbezirk Münster haben wir 2014 insgesamt 90 Gymnasien und 98 Realschulen festgestellt und angeschrieben, mehr als 10% dieser Schulen haben sich beteiligt; in 2014: 9 Realschulen mit 348 (35%) Schülern, 12 Gymnasien mit 645 (65%) Schülern.

[12] In diesem Fall müssten rechnerisch ca. 1/10 aller Gymnasien und 1/6 aller Realschulen in NRW ein entsprechendes Angebot haben.

[13] Um die Daten aussagekräftiger analysieren zu können, wurden häufig die Skalenenden zusammengefasst. D.h., da die Skala von 1 bis 5 geht, bilden 1 und 2 eine (untere) Kategorie; 3 eine mittlere Kategorie sowie 4 und 5 eine obere.

Software. Die Tätigkeiten "Diskussion der Gefahren von Computernutzung", "Recherchieren im Internet" und "Strukturieren und Darstellen von komplexen Zusammenhängen" verbinden die SuS eher nicht mit dem Informatikunterricht. Bezüglich der erwarteten Tätigkeiten sind keine auffälligen geschlechterspezifischen, schulspezifischen sowie auf dem Wahlverhalten oder der Belegung von Informatik beruhende Unterschiede ersichtlich. Es gibt jedoch schwache Korrelationen zwischen der Schulform und einigen erwarteten Tätigkeiten (z. B. Programmieren von Robotern[14]).

- Die hohe Verbreitung von Informatiksystemen unter den SuS und der vorhandene Zugang zum Internet erlauben organisatorisch die schulische Einbindung dieser Informatiksysteme, z. B. in Formulierungen von Hausaufgaben. Denkbar ist insbesondere eine zunehmende Verwendung von schülereigenen Smartphones oder Tablets im Informatikunterricht[15]. Die SuS nutzen Informatiksysteme intensiv: PCs[16] (39% der SuS 1-5 Stunden am Tag), das Internet (60%, 1-5 Stunden am Tag) und Smartphones/Tablets (96% der SuS besitzen ein Smartphone, ca. 45% ein Tablet). 99% aller SuS haben einen Zugang zum Internet.

- Die Vertrautheit mit Standard-Anwendungen wird von SuS als recht hoch eingeschätzt. SuS in Informatikkursen fühlen sich mit allen Anwendungen etwas vertrauter als SuS, die Informatik nicht belegt haben. Sehr vertraut fühlen sich alle SuS mit „WWW/Web-Browser“ (51%), „Computerspielen“ (46%) und „E-Mail-Anwendungen“ (37%). Dies geht einher mit den am häufigsten für das Internet genannten Tätigkeiten: „Webseiten lesen und Videos anschauen (83% aller SuS), gezielt nach Informationen suchen (57%) oder Kommunizieren (56%). Auch mit Textverarbeitungs- und Präsentationssoftware fühlen sich viele SuS vertraut, vermutlich weil sie diese auch in Fächern zuvor kennengelernt haben. Die Vertrautheit mit Tabellenkalkulation und Bild-/Videobearbeitung wird nicht so hoch eingeschätzt. Die Vertrautheit mit „Programmieren“, „Tabellenkalkulation“ und „Computerspielen“ korrelliert schwach bis mittel mit der Anwahl von Informatikkursen[17].

- SuS haben wenige Vorkenntnisse zum Programmieren. Die Vertrautheit mit Programmiersprachen wird von SuS in Nicht-Informatikkursen (78%) als gering eingeschätzt, aber auch in Informatikkursen geben nur 22% der SuS an, dass sie mit

[14] Trifft zu Realschule: 29%; Gymnasium: 53%. Chi-Quadrat: 56,068***, Cramer-V: 0,240***. Signifikanzniveaus werden wie folgt notiert: *: p<0,05, **: p<0,01, ***: p<0,001.

[15] Für die wenigen SuS, die keine Geräte besitzen, dürften sich an den Schulen Modelle zur Ausleihe von Geräten organisieren lassen. Ob „Handyverbote“ an Schulen den Einsatz dieser Informatiksysteme behindern dürfen, ist zu diskutieren.

[16] Im Fragebogen wurde deutlich zwischen PCs und Smartphones/Tablets unterschieden.

[17] Vertrautheit mit:
- Programmiersprachen: Cramer-V: 0,3401***; Lambda (Wahlverhalten abhängig): 0,221***
- Computerspielen: Cramer-V: 0,253***
- Tabellenkalkulation, z.B. Excel: Cramer-V: 0,247***

Programmiersprachen vertraut sind, wobei sich Jungen vertrauter mit Programmiersprachen halten als Mädchen[18]. Viele SuS geben allerdings an, dass sie nicht wissen, was Programmieren (für sie) bedeutet (47%). Von den SuS, die Informatik nicht als Fach haben, geben 64% „ich weiß nicht“ an, bei den SuS in Informatikkursen sind es immerhin noch 25%.

- Die Bedeutung von Informatik wird von Informatikwählern höher eingeschätzt als von Nicht-Wählern. Die SuS, die Informatik wählten, unterhalten sich öfter über informatische Themen, als die Nicht-Wähler. Außerdem ist für ca. 68% der Informatikwähler wichtig, dass Informatik in der Schule unterrichtet wird und dass im Informatikunterricht programmiert wird (nur ca. 32% bei den Nichtwählern). Der Anteil der SuS, die nicht Informatik gewählt haben und sich gegen eine Benotung des Informatikkurses aussprechen, beträgt 44% und ist somit größer als bei den Informatikwählern (30%).

- Ein Desinteresse an Computern führt in der Regel dazu, dass Informatik nicht gewählt wird. Von den 56% aller SuS, die sich für Computer interessieren, hat mehr als die Hälfte (57%) Informatik gewählt. Allerdings haben die SuS, die sich nicht für Computer interessieren, zu 79% Informatik nicht gewählt[19].

- Die Alternativen zum Fach Informatik sind für viele SuS im Wahlpflichtbereich interessanter. Nur 41% geben an, dass dies für sie eher nicht zutrifft.

- Informatik wird eher nicht als Fach für Computerfreaks gesehen (69%). Von den SuS, die Informatik gewählt haben, negieren 78% die Aussage "Informatik ist nur etwas für Computerfreaks". Bei den Nichtwählern sind es aber auch fast 60%.

- Etwa die Hälfte der befragten SuS geben an, dass sie in einem Informatikkurs gerne „mit berufspraktischen Beispielen“ und „viel in Kleingruppen“ arbeiten würden sowie „kleinere Softwareprojekte“ realisieren möchten. Allerdings lehnen auch rund 25 % der SuS die letzten Aussagen eher ab. Deutlicher fällt der Wunsch bei „viel am Computer zu arbeiten“ (79%) und „Software benutzen“ (71%) aus. Es gibt keine geschlechtsspezifischen Unterschiede, lediglich die Realisation von Softwareprojekten wird stärker von den Jungen (61%) als von den Mädchen (39%) präferiert.

- Programmieren und Mathematik beeinflussen bei allen SuS die Wahl von Informatik. Programmieren wird von 37% der SuS als schwierig angesehen. Informatikwähler wollen eher programmieren als nicht Informatikwähler. Von denjenigen, die gerne programmieren wollen, wählten 85% Informatik an (als Erst-, Zweit- oder Drittwahl)[20]. 64% der SuS sprechen sich gegen das Lösen von

[18] Cramer-V: 0,225***

[19] 93% haben Informatik nicht als Erstwunsch angegeben. Lambda: 0,200***, d.h. wenn bekannt ist, ob sich SuS für Computer interessieren, verbessert sich die Vorhersage über die (Nicht-)Wahl von Informatik um 20%!

[20] Cramer-V: 0,489***, Lambda (Wahlverhalten abhängig): 0,416***

mathematischen Aufgaben aus. Allgemein besteht eine Abneigung gegenüber mathematischen Aufgaben im Informatikunterricht: Von denjenigen, die Informatik nicht gewählt haben, wollen 71% keine mathematischen Aufgaben lösen, und auch bei den Informatikwählern sind es noch 58%.

Zahlreiche geschlechtsspezifische Zusammenhänge lassen sich aufzeigen:

- Mädchen verbringen weniger Zeit am PC als Jungen, aber etwa gleich viel Zeit im Internet. Während 53% der Jungen angeben, zwischen 1-5 Stunden pro Woche am PC zu verbringen, sind es unter den Mädchen nur 22%[21]. Im Internet verbringen 60% der Jungen sowie der Mädchen zwischen 1 und 5 Stunden wöchentlich ihrer Zeit[22], unterscheiden sich aber hinsichtlich der Nutzung des Internets: Während beispielsweise „Webseite lesen und Video schauen" sowohl von über 80% der Jungen als auch der Mädchen benannt wird, lassen sich auffällige Unterschiede hinsichtlich der Tätigkeiten „Programmieren", „gezielt nach Informationen suchen" und „Shoppen" feststellen. Ein größer Unterschied besteht bei den Geschlechtern erwartungsgemäß zu „Online-Spielen" (Jungen 64%; Mädchen 18%).

- Das Fach Informatik wird stärker von Jungen als von Mädchen angewählt. Lediglich 31% der Mädchen haben Informatik überhaupt auf ihrem Wahlzettel angegeben, gegenüber 74% der befragten Jungen[23]. Immerhin haben 22% aller befragten Mädchen Informatik als Erstwunsch angegeben (Jungen 55%)[24].

- Hinsichtlich der Faktoren/Überlegungen zur Wahl- bzw. Nicht-Wahl von Informatik zeigen sich zwischen Mädchen und Jungen die folgenden auffälligen Unterschiede (hinsichtlich der Zustimmung):

    - Ich interessiere mich für Computer (m: 79%; w: 29%)
    - Informatik ist wichtig für meine Zukunft (m: 43%; w: 20%)
    - Ich möchte programmieren (m: 45%; w: 17%)
    - Andere Wahlfächer sind für mich interessanter (m: 31%; w: 66%)

6 Fazit und Ausblick

Je länger wir uns mit den Daten beschäftigen, desto mehr potentiell interessante

[21] Cramer-V 0,374***, Lambda (Zeit am PC abhängig): 0,173**
[22] 22% der Jungen und 21% der Mädchen geben an, dass sie mehr als 5 Stunden ihrer wöchentlichen Zeit im Internet verbringen.
[23] 60% der Jungen und 25% der Mädchen haben Informatikunterricht bekommen. Weitere häufig gewählte Fächer: Fremdsprachen: w: 30%, m: 12%; Naturwissenschaften: w: 29%, m: 22%; Andere: w: 14%, m: 21%.
[24] Cramer-V 0,427***, Lambda (Wahlverhalten abhängig): 0,288***

Korrelationen werden uns bewusst. Die Auswertung wird daher sicherlich noch fortschreiten und zu weiteren neuen Erkenntnissen und Hypothesen führen. Es wird zu diskutieren sein, welche Ergebnisse zu welchen Konsequenzen für den Informatikunterricht und seine „Bewerbung“ führen könnten.

Sicherlich sind einige unserer Ergebnisse nicht überraschend, sind vergleichbar mit Erkenntnissen zur GOSt und lassen sich teilweise auch in anderen Studien (z.B. zum Medienkonsum) wiederfinden. Allerdings ergeben sich aus letzteren Studien keine direkten Rückschlüsse zum Wahlverhalten bezüglich des Fachs Informatik.

Vorerfahrungen und Vorkenntnisse bestimmen die Wahl bzw. Nicht-Wahl von Informatikkursen bereits in der Sekundarstufe I mit. Solange es in NRW kein Pflichtfach Informatik vor dem Wahlpflichtbereich gibt, könnten die Ergebnisse einen (verpflichtenden) Schnupperkurs zur Informatik motivieren, wie dies bereits zur Förderung von Mädchen angeregt wurde. Effektiver sind vermutlich informatiknahe Pflichtkurse, wie sie schulintern teilweise in NRW bereits angeboten werden; hier fehlen jedoch entsprechende Studien.

Wie bereits angedeutet, planen wir derzeit keine Ausdehnung dieser Studie auf andere Regierungsbezirke in NRW. Interessanter erscheinen uns vergleichbare Studien in anderen Bundesländern und vertiefende Studien zu den sich ergebenden Hypothesen. Mit einer umfassenden Veröffentlichung des Designs und aller ausgewerteten Daten möchten wir jedoch zu einer Kooperation im Rahmen des Projekts KISS einladen.

Literaturverzeichnis

[Bu87] Bund-Länder-Kommission für Bildungsplanung und Forschungsförderung (1987): Gesamtkonzept für die informationstechnische Bildung. Bonn (Materialien zur Bildungsplanung. Heft 16). http://www.blk-bonn.de/papers/heft16.pdf, 17.04.2014.

[En14] Engbring, D.: Zum Verhältnis von Informatik und Naturwissenschaften. Ein Vorschlag zur MINT-Förderung. In: Thomas, M.; Weigend, M. (Hrsg.): Informatik und Natur. 6. Münsteraner Workshop zur Schulinformatik - 9. Mai 2014. Books on Demand GmbH, Norderstedt. S. 9-18.

[Kn14] Knobelsdorf M.; Magenheim J.; Brinda T.; Engbring D.; Humbert L.; Pasternak A.; Schroeder U.; Thomas M.; Vahrenhold J.: ‘Computer Science Education in North-Rhine Westphalia, Germany – A Case Study.’ ACM Transactions on Computing Education 2014.

[MS05] Magenheim, J.; Schulte, C. (2005): Erwartungen und Wahlverhalten von Schülerinnen und Schülern gegenüber dem Schulfach Informatik. In: Friedrich, S. (Hg.): Unterrichtskonzepte für informatische Bildung. INFOS 2005, 11. GI-Fachtagung Informatik und Schule, 28. - 30. September 2005 an der TU Dresden. Fachtagung Informatik und Schule; Gesellschaft für Informatik; GI-Fachtagung Informatik und Schule; Infos 2005. Bonn: Ges. für Informatik (GI-Edition Proceedings, 60), S. 111–122.

[NR11] NRW: Das Schulwesen in NRW aus quantitativer Sicht. Amtliche Schuldaten zum Schuljahr 2011/12. http://www.schulministerium.nrw.de/docs/bp/Ministerium/Service/Schulstatistik/Amtliche-Schuldaten/StatUebers375-Quantita2011.pdf, 17.03.2014.

[RNG01] Richter, T.; Naumann, J.; Groeben, N.: Das Inventar zur Computerbildung (INCOBI): Ein Instrument zur Erfassung von Computer Literacy und computerbezogenen Einstellungen bei Studierenden der Geistes- und Sozialwissenschaften. Psychologie in Erziehung und Unterricht, 48, 2001; Seite 1-13.

[TY14] Thomas, M.; Yomayuza, A.: Wahlverhalten zum Schulfach Informatik in der SI – Erste Ergebnisse einer Studie in NRW. In: Thomas, M., Weigend, M. (Hrsg.): Informatik und Natur. 6. Münsteraner Workshop zur Schulinformatik - 9. Mai 2014. Books on Demand GmbH, Norderstedt. S. 19-25.

Wir bedanken uns bei den Schulen, Lehrkräften, Experten und der Bezirksregierung für die Unterstützung dieser Umfrage im Rahmen des Projekts KISS.

Jens Gallenbacher (Hrsg.): INFOS 2015 – Informatik allgemeinbildend begreifen
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 191

Agile Softwareentwicklung im Informatikunterricht – Ein Best-Practice-Beispiel am Spiel „Pengu“

Petra Kastl, Silva März und Ralf Romeike[1]

Abstract: Im Beitrag wird der Einsatz agiler Methoden im Informatikunterricht an einem konkreten Beispiel illustriert. Am Spiel „Pengu“, welches mit der Programmierumgebung Greenfoot implementiert werden soll, werden agile Praktiken wie User Stories, Tasks, Zeit- und Prioritätenabschätzung, Iterationen und ein mögliches Project Board exemplarisch dargestellt. Das Beispiel kann als Orientierung genutzt werden, um agile Methoden zu verstehen und in eigenen Projekten umzusetzen.

Keywords: Projektunterricht, Agile Methoden der Softwareentwicklung

1 Einleitung

Eine wesentliche Herausforderung bei der Implementierung innovativer Unterrichtsmethoden stellt das Finden geeigneter Beispiele dar, an denen sowohl die Lehrkraft, als auch die Schülerinnen und Schüler Kernideen und Umsetzungsmöglichkeiten konkret nachvollziehen und erfassen können. Eine solche Innovation stellen im Moment agile Methoden der Softwareentwicklung dar, die Potential besitzen, Probleme zu vermeiden, die Projektunterricht, der herkömmlichen, linearen Prozessmodellen folgt, regelmäßig offenbart. Unter agilen Methoden ist eine Sammlung von Artefakten und Praktiken zu verstehen, die den Projektablauf strukturieren und dabei den Beteiligten konkrete Handlungsoptionen und Kommunikationsanlässe zur Hand geben. Die Praktiken müssen nicht starr verwendet werden, sondern können passend zu dem jeweiligen Projekt neu zusammengestellt werden. Etablierte Frameworks agiler Methoden sind zum Beispiel Scrum, eXtreme Programming oder auch Kanban.

Im Folgenden werden für Unterrichtsprojekte sinnvolle agile Praktiken kurz charakterisiert und am Beispiel des Spiels „Pengu“ exemplarisch umgesetzt. Der Ablauf orientiert sich dabei am agilen Projektmodell für den Informatikunterricht AMoPCE ([RG12], vgl. Abb. 1). Hierbei werden zu Beginn alle bisher bekannten und gewünschten Anforderungen an das System bzw. die Software definiert und in diesem Projekt als sogenannte User Stories festgehalten. Diese als „Geschichten“ beschriebenen Funktionen werden noch weiter hinsichtlich ihrer Umsetzung spezifiziert, Prioritäten und Aufwand abgeschätzt und auf einem Project Board festgehalten. Der weitere Prozess gliedert sich in mehrere Iterationen, wobei jede Iteration ein „Mini-Projekt“ darstellt, in welchem die

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, Martensstr. 3, 91058 Erlangen
petra.kastl@fau.de, silva.maerz@fau.de, ralf.romeike@fau.de

Phasen des Planens, Entwerfens, Implementierens und Testens jeweils abschließend durchlaufen werden. Innerhalb der Iterationen wird zu Beginn jeder Unterrichtseinheit ein Standup-Meeting durchgeführt, die Projektorganisation wird unterstürzt durch ein Project-Board.

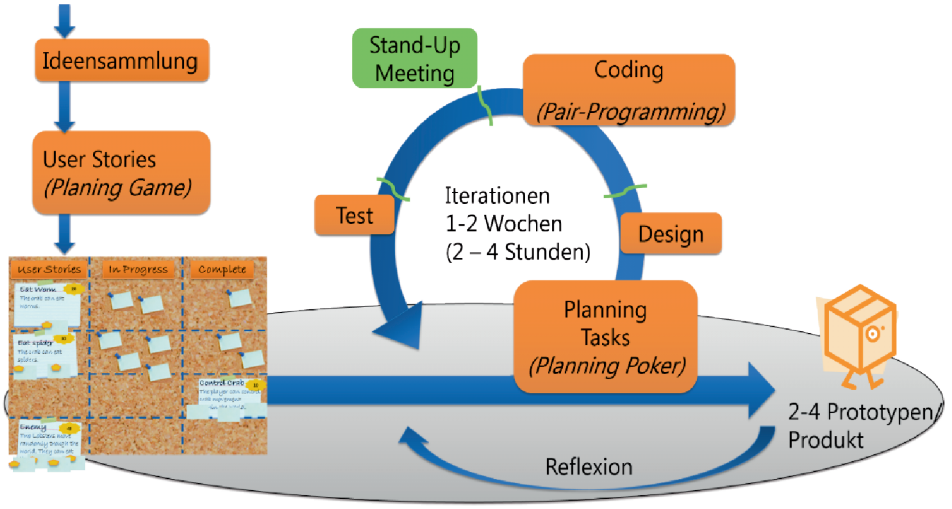


Abb. 1: Agiles Projektmodell für den Informatikunterricht AMoPCE

2 Agile Methoden am Beispiel von „Pengu“

Das Spiel „Pengu“ ist ein Jump ’n’ Run-Spiel, bei welchem es die Aufgabe des Spielers ist, eine Spielfigur auf einer bewegten Wolke über einen Abgrund zu steuern (siehe Abb. 2). Umgesetzt wird dieses Szenario in diesem Beispiel mit Greenfoot. Greenfoot stellt auf der Programmiersprache Java basierende Miniwelten zur Verfügung, welche insbesondere für zweidimensionale Spiele und Simulationen geeignet sind. Greenfoot ermöglicht Programmieranfängern, die objektorientierte Programmierung auf interaktive Weise kennenlernen.

Im Folgenden soll nun die Spielidee konkret mit Hilfe agiler Methoden umgesetzt werden. Hierzu wird das Szenario zuerst in sogenannte User Storys und Tasks aufgeteilt. Anschließend wird das weitere Projektvorgehen beschrieben.

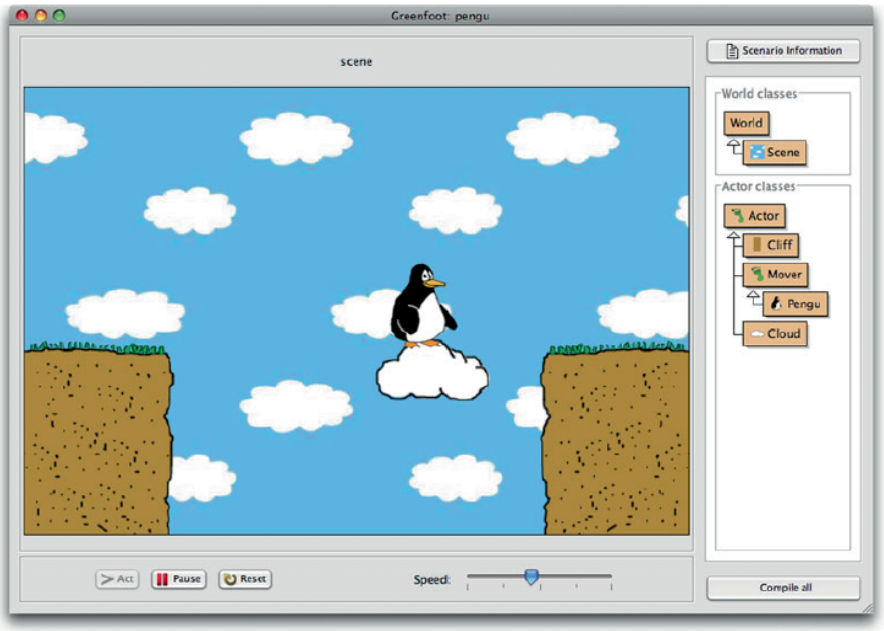


Abb. 2: Spiel „Pengu“ in Greenfoot

2.1 User Stories und Tasks

User Stories beschreiben in kurzer Form Funktionalitäten und Fähigkeiten der zu entwickelnden Software, die dem Nutzer zur Verfügung stehen sollen. Jede User Story beschränkt sich dazu auf eine konkrete Aktivität und wird aus Sicht des Kunden beschrieben. Entsprechend sollten sie so kurz sein, dass sie auf eine Karteikarte passen und auch ohne Programmierkenntnisse verständlich sind. Auf den Karteikarten kann später auch der prognostizierte Aufwand sowie die Priorität vermerkt werden.

Die Erstellung der User Stories erfolgt unter Einbeziehung des Auftraggebers und erfordert domänenspezifisches Wissen aus dem Kontext der zu erstellenden Software. So ist es in unserem Beispiel erforderlich, dass das Team eine Vorstellung vom Spielfluss und Möglichkeiten solcher Jump-and-Run-Spiele besitzt, um die Anforderungen klar aufzustellen und auch eigene Ideen mit einzubringen.

Eine User Story besteht aus einem Titel, einer Beschreibung, einer zeitlichen Abschätzung und einer Priorität. Sie ist nicht länger als drei Sätze, verwendet keine technischen Begriffe und spezifiziert auch keine Werkzeuge. Auch im professionellen Bereich werden für User Stories häufig Karteikarten verwendet, da diese ein passendes Format haben und gut zu handhaben sind.

Alle User Stories zusammen spezifizieren das finale Produkt. Deshalb bilden diese auch die Grundlage für die Präzisierung des Projektziels und die Verständigung mit dem Kunden und der Projektgruppe. Gemeinsam mit dem Kunden werden anschließend die einzelnen Prioritäten festgelegt, aus welchen hervorgeht, welche User Stories zuerst umgesetzt werden sollen und welche ggf. auch nicht umgesetzt werden (vgl. Abschnitt 2.3).

In unserem „Pengu"-Beispiel helfen die User Stories, das umfangreiche Spiel in mehrere Teilbereiche zu gliedern und aufzuteilen. Weiterhin wird durch User Stories leichter deutlich, ob etwas vergessen wurde. Als Grundsatz sollte für jede Rolle (hier z. B. Pinguin bzw. Wolke) und jede einzelne Aufgabe (hier z. B. Pinguin kann sich bewegen) eine eigene User Story geschrieben werden (vgl. [Wo11]).

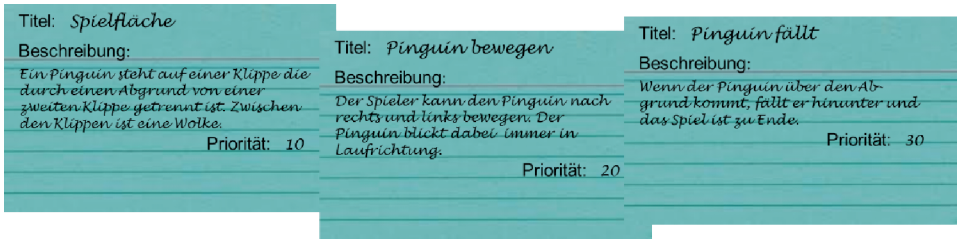


Abb. 3: Die ersten drei User Stories von „Pengu"

Die ersten drei User Stories dienen in unserem Beispiel der Grundfunktionalität des Spiels: „Spielfläche", „Pinguin bewegen" und „Pinguin fällt". Weitere User Stories beinhalten mögliche Ausbaufunktionen. Abb. 3 zeigt die entsprechenden Beschreibungen der User Stories, sowie die Prioritäten. Da alle drei User Stories für das Spiel grundlegend sind, wurden sehr hohe Prioritäten festgelegt (vgl. Abschnitt 2.2), aus denen auch implizierte Abhängigkeiten ersichtlich sind (ehe das Szenario steht, kann keine Aktivität implementiert werden). Nach der Festlegung der User Stories kann mithilfe des Planungsspiels (Planning Poker) der zeitliche Aufwand der Umsetzung der User Stories abgeschätzt werden. Möglichkeiten zur Umsetzung werden im nachfolgenden Abschnitt beschrieben.

Nach der ersten Zeitabschätzung und Priorisierung werden die User Stories der ersten Iteration in Tasks überführt. Ein Task ist eine grobe Beschreibung eines Arbeitspakets, das von einem Entwickler oder Entwicklerpaar (beim Pair Programming) umgesetzt werden kann. Entsprechend kann eine User Story als eine Sammlung von Tasks angesehen werden. Während User Stories die Ziele des „Pengu"-Spiels aus Sicht des Kunden beschreiben, müssen die Schüler nun ihre Perspektive ändern und die Ziele aus Sicht des Entwicklers betrachten. Beim Aufteilen der User Stories zu Tasks sind bereits verschiedene Designentscheidungen zu treffen. Ein prototypisches Beispiel, wie eine umfangreichere User Story in Tasks überführt werden kann, zeigt Abb. 4. Die Überführung der ersten drei User Stories von „Pengu" in Tasks ist in Abb. 5 dargestellt. Hierbei wird deutlich, dass die Tasks bereits konkrete Umsetzungsangaben beinhalten, woran die

Entwicklersicht deutlich wird. Tasks dürfen auch konkrete, technische Begriffe oder kurze Quellcodehinweise enthalten. Notiert werden die Tasks in der Regel auf Klebezetteln (Post-Its). Jeder Task erhält eine eigene zeitliche Abschätzung, die wieder über ein Planungsspiel erarbeitet werden kann. Bei größeren Projekten bietet es sich außerdem an, das Namenskürzel der Schüler zu notieren, die diesen Task bearbeiten. Lehrer und Schüler können hierdurch genau sehen, welcher Schüler welchen Tasks übernommen hat. Weiterhin wird es für die Schüler leichter, am Ende eine Soll-Ist-Gegenüberstellung vorzunehmen.

Abb. 4: Aufteilen von User Stories in Tasks

2.2 Prioritäten- und Zeitabschätzung

Die erste Prioritätenschätzung wird in der Regel gemeinsam mit dem Kunden vorgenommen, wofür sich im Unterricht der Lehrer anbietet. Hierbei ist zu berücksichtigen, was dem Kunden besonders wichtig ist und was gegebenenfalls nur ein „nice to have“ wäre. Hier kommt der flexible Charakter des agilen Modells zum Tragen: Entsprechend dem agilen Manifest (vgl. [Be01]) ist es wichtiger, auf Veränderungen reagieren zu können, als nur strikt einem Plan zu folgen. Da – wie jeder Lehrer im Unterricht regelmäßig erlebt – organisatorische, technische oder gesundheitliche Probleme den geplanten Unterrichtsverlauf stören können, wird durch die Festlegung und Umsetzung höher priorisierter Aufgaben ermöglicht, dass selbst bei reduzierter Projektlaufzeit ein lauffähiger und getesteter Prototyp vorhanden ist – und selbst die Prioritäten im Laufe des Projekts noch angepasst werden können. Die Prioritäten werden in Vielfachen von 10 angegeben,

wobei 10 die höchste Priorität hat, bei unserem Spiel ist es die Spielfläche, und nach oben hin die Priorität abnimmt. Verschiedene User Stories können auch die gleiche Priorität besitzen.

Die zeitliche Einschätzung von Tasks wird von den Schülern eigenständig ohne Kunden vorgenommen, es ist aber wichtig, dass alle Teammitglieder in den Schätzprozess mit eingebunden sind. Als Grundlage sollten sinnvolle Zeiteinheiten gewählt werden. Im professionellen Bereich werden Tage als Basis genommen, in Unterrichtsprojekten haben sich Einheiten von 15 Minuten als sinnvoll und gut handhabbar erwiesen. Eine gute Möglichkeit, Vorgaben durch nur wenige erfahrene Schüler zu vermeiden und alle Schüler in die Kommunikationsprozesse einzubinden, stellt das Planungsspiel (Planning Poker, vgl. [PM08]) dar. Hierbei legen sich alle Teammitglieder unter Verwendung eines speziellen Kartenspiels verdeckt fest, welcher Aufwand ihrer Meinung nach für einen bestimmten Task erforderlich ist. Nach Aufdecken der Einschätzungen müssen die extremen Schätzungen begründet werden und es wird versucht, in der Diskussion einen Konsens zu finden. Alternativ wird ein Durchschnittswert der Schätzungen herangezogen. Der Aufwand einer User Story ergibt sich aus den addierten Schätzwerten der einzelnen Tasks. Die Diskussion zur Einschätzung ist gerade für unerfahrene Schüler sehr wichtig, da die Schüler hierbei auch mögliche Strategien austauschen und voneinander lernen können. Gleichzeitig ist dies eine gute Kontrolle, ob alle das gleiche Verständnis von dem haben, was sie mit den Tasks geplant haben. Wenn die Schätzungen auseinander gehen, werden ggf. Unklarheiten in der Diskussion aufgedeckt.

2.3 Iterationen und Prototypen

In Schulsoftwareprojekten stellt sich das Planen und Durchführen langer Projektphasen regelmäßig als Hauptschwierigkeit heraus. Insbesondere ist es schwierig, die Motivation der Schüler aufrecht zu erhalten, wenn über lange Strecken der Erfolg nicht sichtbar wird und das Produkt erst in späten Implementierungsphasen oder zum Ende des Projekts ausprobiert werden kann. Durch die iterative Entwicklung haben die Schüler nun die Möglichkeit, Prototypen des „Pengu“-Spiels in verschiedenen Entwicklungsphasen auszuprobieren und dabei zu überprüfen, ob die Teilziele erreicht wurden.

Im „Pengu“-Beispiel wird die erste Iteration (vgl. Abb. 5) bestimmt durch die Herstellung der Grundfunktionalitäten des Spiels. Der Vorteil der iterativen Vorgehensweise wird unmittelbar deutlich: Bereits früh im Projekt ist das Spiel „spielbar“, der erste Erfolg kann bereits getestet und gegenüber Mitschülern und dem Lehrer demonstriert werden. Sogar das Hinzufügen weiterer Spielideen (Features) ist vor einer Iteration noch möglich, was bei linearen Vorgehensmodellen nahezu ausgeschlossen wäre. Zum Abschluss jeder Iteration ist es möglich, zu reflektieren, wie gut der Prozess bis dahin gelaufen ist und ob sich das Team auf dem richtigen Weg befindet.

In den weiteren Iterationen kommen nun nach und nach, der zuvor bestimmten Priorität entsprechend, weitere Funktionalitäten hinzu, wie sie auf den User Stories festgehalten wurden, z. B. das Bewegen der Wolke, Springen, Überqueren des Abgrunds in Iteration

2, Punktezähler, Sternenhimmel und Sternesammeln in Iteration 3 sowie das Berücksichtigen verschiedener Leben, Gewinnen und Spielende in Iteration 4. Erweiterungen des Spiels, wie bspw. die Eiszapfen, dargestellt in Abb. 4, können auch später noch hinzugefügt werden.

So stellt sich jede Iteration als Mini-Projekt dar, in welchem jede Phase des Softwareentwicklungsprozesses (Anforderungen, Entwurf, Implementieren und Testen) einmal durchlaufen wird und aufgrund der geringeren Komplexität einfacher zu handhaben ist. Hierdurch erhalten die Schüler die Gelegenheit, den gesamten Prozess mehrfach in einem Projekt zu durchlaufen, daran zu lernen, bisherige Erfahrungen zu reflektieren und verschiedene Aufgaben im Team zu übernehmen.

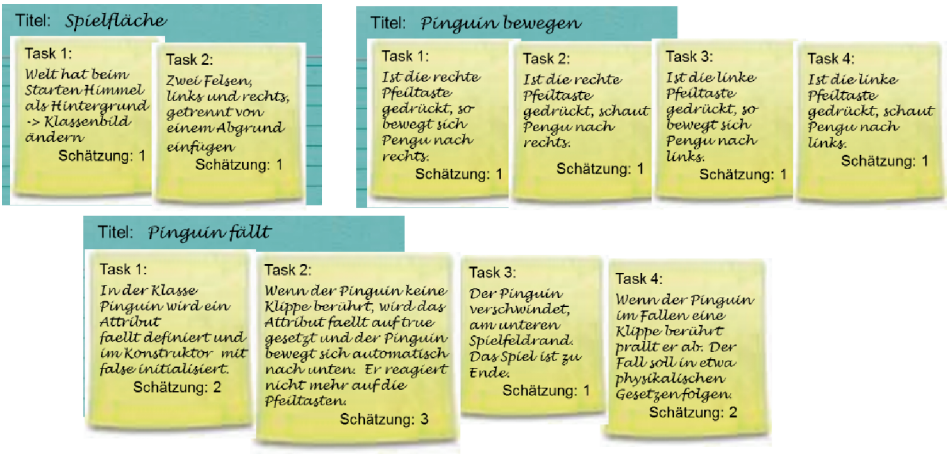


Abb. 5: Tasks für die erste Iteration

2.4 Project Board und Standup-Meetings

Das Project Board ist der zentrale Informations- und Organisationsort des Projekts. Es visualisiert die Ziele und den Status der aktuellen Iteration und unterstützt zielgerichtete Diskussionen anhand der angebrachten User Stories und Tasks. Hier finden sich im Wesentlichen drei Bereiche: Der Vorrat der User Stories mit Tasks, Tasks in Bearbeitung, sowie erledigte Tasks und User Stories. Zu Beginn einer Iteration befinden sich alle Karten auf der linken Seite. Sobald ein Schüler oder Schülerpaar mit einer Aufgabe beginnt, wird ein Post-it mit dem entsprechenden Task aus der Vorratsspalte genommen, mit Namenskürzel versehen und in die Spalte „in Bearbeitung“ gehängt. Sobald der Task bearbeitet wurde, wird er auf „erledigt“ weitergehängt. Ein Beispiel für ein Project Board in der ersten Iteration des „Pengu“-Spiels zeigt Abb. 6.

Am Project Board finden auch die regelmäßigen Standup-Meetings statt, in welchen zu Beginn einer Unterrichtsstunde organisatorische Aspekte der Projektdurchführung be-

sprochen werden. Sie stellen eine geschickte Möglichkeit dar, den Stand des Projekts, das Geleistete der letzten Unterrichtsstunde und Probleme, aber auch die Ziele des Tages zu besprechen. In der Regel sind diese Meetings so kurz, dass es sich gar nicht lohnt, sich erst hinzusetzen. Durch das Stehen bemüht sich auch jeder, sich kurz zu fassen und auf das Wesentliche zu beschränken. Reihum beantwortet hier jeder Schüler die Fragen: Was habe ich seit dem letzten Mal getan? Was werde ich heute in Angriff nehmen? Welche Probleme hatte ich oder sehe ich auf mich zukommen?

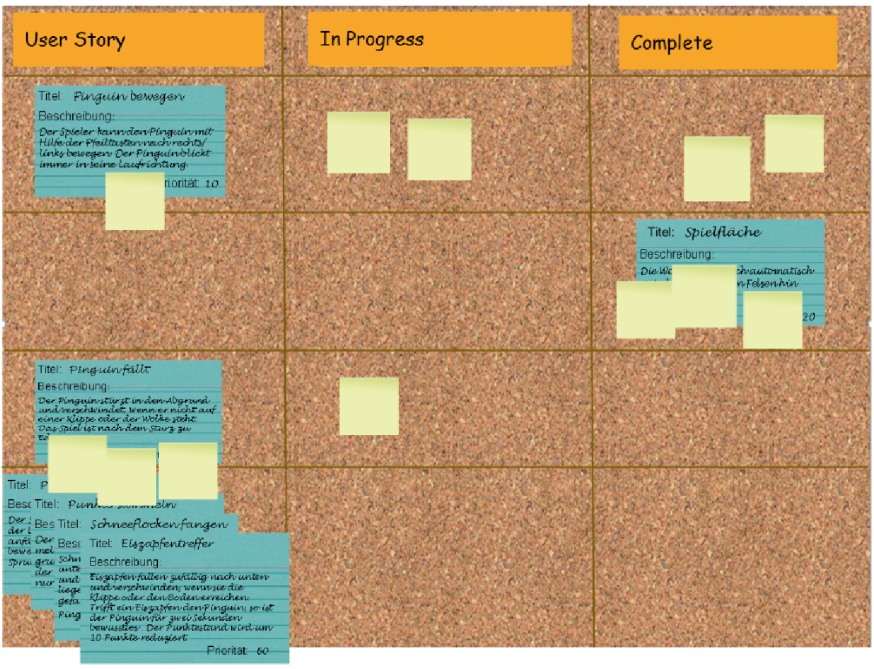


Abb. 6: Das Project-Board zum „Pengu“-Spiel in der ersten Iteration

3 Praxiserfahrungen und Diskussion

Das beschriebene Beispiel wurde erstmals in einem Workshop zu agilen Methoden für den Informatikunterricht von Lehrern ausprobiert und anschließend an verschiedenen Schulen in der Unterrichtspraxis eingesetzt (vgl. verschiedene Beiträge in diesem Band). Dabei hat sich gezeigt, dass das Spiel als Unterrichtsgegenstand sehr motivierend war und sich die Methoden als sehr flexibel erweisen und für Projekte unterschiedlichster Ausprägungen geeignet sind. So können sowohl die Lerngruppen, Vorkenntnisse als auch zeitlichen Gegebenheiten ganz unterschiedlich ausgestaltet werden, weshalb in diesem Beispiel auf die Konkretisierung der Rahmenbedingungen verzichtet wurde.

Schwierigkeiten ergaben sich regelmäßig aus der Notwendigkeit, die agilen Praktiken zuerst selbst erfassen zu müssen. Wie bspw. User Stories formuliert sein sollen und wie man diese in Tasks überführt, stellte sich mangels Erfahrung im ersten Workshop als eine Herausforderung dar. Mit diesem Beispiel ist nun eine Vorlage gegeben, die zur Orientierung für das Erlernen agiler Praktiken und die Durchführung agiler Softwareentwicklungsprojekte im Informatikunterricht dienen kann.

Literaturverzeichnis

[Be01] Beck, K.; Beedle, M.; Bennekum, A.v. et al.: Manifesto for Agile Software Development. http://www.agilealliance.org/the-alliance/, abgerufen am 10.04.2015.

[PM08] Pilone, D.; Miles, R.: Softwareentwicklung von Kopf bis Fuß. Beijing: O'Reilly, 2008.

[RG12] Romeike, R.; Göttel, T.: Agile Projects in High School Computing Education – Emphasizing a Learners ' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE'12): ACM, Hamburg, 2012.

[Wo11] Wolf, H.: Agile Softwareentwicklung (2., aktualisierte und erweiterte Aufl). Heidelberg: Dpunkt.Verlag, 2011.

Anhang: User Stories des „Pengu“-Spiels

Titel: Spielfläche

Beschreibung:

Ein Pinguin steht auf einer Klippe die durch einen Abgrund von einer zweiten Klippe getrennt ist. Zwischen den Klippen ist eine Wolke.

Priorität: 10

Titel: Pinguin fällt

Beschreibung:

Wenn der Pinguin über den Abgrund kommt fällt er hinunter und das Spiel ist zu Ende.

Priorität: 20

Titel: Pinguin überquert Abgrund

Beschreibung:

Der Pinguin kann auf der Wolke stehen und bewegt sich mit ihr mit. Der Spieler kann ihn auf der Wolke wie auf den Klippen nach rechts und links bewegen.

Priorität: 40

Titel: Punktezähler

Beschreibung:

Das Spiel hat einen Punktezähler, den die Welt verwaltet.

Priorität: 55

Titel: Sternenhimmel

Beschreibung:

Es erscheinen Sterne am Himmel, stehen dort unterschiedlich lang und verschwinden dann wieder.

Priorität: 60

Titel: Pinguin bewegen

Beschreibung:

Der Spieler kann den Pinguin nach rechts und links bewegen. Der Pinguin blickt dabei immer in Laufrichtung.

Priorität: 20

Titel: Wolkenbewegung

Beschreibung:

Die Wolke bewegt sich automatisch zwischen den Klippen hin und her.

Priorität: 30

Titel: Pinguin springt

Beschreibung:

Der Spieler kann den Pinguin springen lassen. Wenn der Pinguin fällt reagiert er nicht. Während des Sprungs behält der Pinguin eine anfängliche horizontale Bewegung und seine Blickrichtung bei.

Priorität: 50

Titel: Pengu gewinnt

Beschreibung:

Auf der rechten Klippe steht eine Tür. Solange der Spieler mehr als 15 Punkte hat ist sie offen, sonst ist sie geschlossen. Erreicht Pengu die Tür, wenn sie offen ist ertönt eine Fanfare und der Spieler hat gewonnen.

Entwicklung eines agilen Frameworks für Projektunterricht mit Design-Based Research

Petra Kastl und Ralf Romeike[1]

Abstract: Fachdidaktische Innovationen stellen Forscher und Praktiker grundsätzlich vor die Herausforderung, Theorie mit Praxis in Einklang zu bringen. Besonders in der Informatik bergen kontinuierliche fachliche Weiterentwicklungen bedeutendes Potential für didaktische und methodische Neuerungen. Der Beitrag skizziert am Beispiel eines agilen Modells für Projekte einen Forschungsprozess, der die Implementierung und Weiterentwicklung des Modells unter Einbeziehung unterrichtspraktischer Expertise begleitet. Zusätzlich werden Erfahrungen aus der ersten Iteration des dem Prozess zugrunde liegenden Design-Based Research-Ansatzes beschrieben.

Keywords: agile Methoden, agile Praktiken, Design-Based Research, Projektunterricht

1 Motivation

In der professionellen Softwareentwicklung hat sich in den letzten Jahren herausgestellt, dass sequentielle Vorgehensmodelle wie das Wasserfallmodell oft zu mangelhafter Qualität, größeren Terminverschiebungen und wenig Kundenzufriedenheit führen – für Entwickler eine frustrierende Situation. Ähnliche Probleme treten auch in Schulprojekten auf, die sich am Wasserfallmodell orientieren: Projekte sind schwer planbar, werden aus Zeitmangel oft nicht mit einem nutzbaren Produkt beendet und lange Modellierungs- und Testphasen sind kaum zu motivieren [HNR07, We05]. In der IT Branche setzt man zur Lösung mehr und mehr auf agile Methoden, also verstärkt auf Kommunikation und Kooperation, Mitarbeitermotivation sowie eine kontinuierliche Produkterstellung in kurzen, iterativen Entwicklungsphasen, die durch eine langfristige Produktvision gelenkt werden. Agile Methoden beruhen entsprechend des agilen Manifests [Be01] auf Werten und Praktiken, die auch für den Schuleinsatz angebracht erscheinen. Auf der WiPSCE 2012 stellten Romeike und Göttel ein auf agilen Methoden basierendes Modell für Projekte im Informatikunterricht (AMoPCE) vor [RG12]. Das Modell stellt für die Praxis einen leicht einsetzbaren Satz agiler Praktiken bereit, der es erlaubt im Unterricht agile, kommunikationsfördernde Prozesse einzusetzen, die schnell nutzbare Teilergebnisse liefern und die das Entwicklerteam ins Zentrum stellen.

Ein Problem bei der theoretischen Entwicklung eines so umfangreichen Modells ist, dass praxisrelevante Aspekte und auftretende Schwierigkeiten nur eingeschränkt vorhergesehen werden können. Um Schwierigkeiten zu vermeiden, wie sie bei der Umsetzung von

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didakitk der Informatik, Martensstr. 3, 91058 Erlangen
petra.kastl@fau.de, ralf.romeike@fau.de

am Wasserfallmodell orientierten, linearen Modellen im Schulunterricht auftreten, soll das theoretisch entwickelte agile Modell in verschiedenen schulpraktischen Kontexten erprobt und mit Hilfe eines Design-Based Research-Ansatzes in einem zyklischen Prozess verfeinert und weiterentwickelt werden. Ziel ist es, die berufsspezifischen Kompetenzen erfahrener Lehrkräfte in den Prozess einzubinden und ihre Beobachtungen bei der Umsetzung, ihre Ideen, Interpretationen und individuellen Anpassungen in jedem Zyklus zu bündeln und zu reflektieren, um die Weiterentwicklung mittels formativer Evaluation empirisch abzusichern. Schrittweise und forschungsgeleitet soll AMoPCE so zu einem praxiserprobten, offenen und flexiblen „Methodenkoffer" weiterentwickelt werden, mit dessen Hilfe Projekte mit einer individuell angepassten, sinnhaften agilen Vorgehensweise erfolgreich durchgeführt werden können. Das vorgestellte Vorgehen und die zugrundeliegenden Überlegungen können als Basis dienen, auch andere Innovationen forschungsgeleitet in der Praxis des Informatikunterrichts zu verankern.

2 Projekte im Informatikunterricht

Projektunterricht beschreibt unabhängig vom Schulfach eine ganzheitliche, offene Lernform, die mit hohem Anteil an Mitbestimmung der Teilnehmer in Phasen abläuft [FS82]. Beschrieben wird er meist über eine Liste von Arbeitsschritten und/oder Merkmalen, wie beispielsweise bei Gudjons oder Frey. Ausgehend von komplexen Aufgaben oder offenen Fragestellungen entwickeln und konkretisieren die Teilnehmer entsprechend ihren Neigungen und Interessen Ideen, bei deren Umsetzung sie selbstorganisiert und zielgerichtet arbeiten und die Verantwortung für das Projekt gemeinsam übernehmen [Gu14]. Sie benötigen dazu vielfältige fachliche, methodische und soziale Fähigkeiten und Fertigkeiten, deren Erwerb, Anwendung und Weiterentwicklung Ziele von Projektunterricht sind [Fr83]. Eine bekannte Schwierigkeit ist, dass Schülerinnen und Schüler (SuS) mit der Komplexität und den Anforderungen eines Projekts oft überfordert sind. Aufgabe der Lehrkraft ist es dann, das Projekt in kleinere Teilprojekte zu gliedern [SS05].

Bei Softwareentwicklungsprojekten im Informatikunterricht liegt eine besondere Situation vor: Da professionelle Softwareentwicklung in Projekten stattfindet, gibt es hierfür seit den 1970iger Jahren wissenschaftlich verankerte Vorgehensmodelle. Will man den Lernenden einen Einblick in die „echte Welt" vermitteln, muss man diese Modelle berücksichtigen. Gleichzeitig stehen sie zum Teil im Widerspruch zu wesentlichen Kriterien und Zielen von Projektunterricht. Daher findet man – historisch bedingt – in der Informatik überwiegend adaptierte Modelle, die versuchen, die Kernideen und Ziele von Projektunterricht bestmöglich mit dem Wasserfallmodell zu verweben [Fr83, SS11]. Unterhält man sich mit Lehrkräften über Projekte und studiert man didaktische Literatur, legt das die Vermutung nahe, dass solche linearen Modelle aufgrund schulischer Anforderungen und Organisationsstrukturen in der Umsetzung häufig zu vielerlei Problemen führen [HNR07, Hu05, MH10, SS11, We05]. SuS sind regelmäßig mit der Komplexität überfordert und können Inhalte und Ressourcen kaum selbständig organisieren bzw. planen. Eine Folge ist, dass sie die Verantwortung für das Projekt nicht gemeinsam

übernehmen können, sodass oft geringe Eigeninitiative und Motivation bei Teilen des Teams beobachtet wird[2]. Gleichzeitig muss die Lehrkraft in hohem Maße motivieren, unterstützen, planen und leiten[3,4]. Aber auch für Lehrkräfte ist die Planung innerhalb der Organisationsstrukturen von Schule schwierig. Unfertige Produkte und schlechte Produktqualität sind die Folge[5,6]. Diese können dann auch nur sehr eingeschränkt von SuS reflektiert und bewertet werden. Eine weitere Schwierigkeit für Lehrer besteht darin, dass das Wasserfallmodell keine konkreten Techniken zur Verfügung stellt, die Kommunikation und Interaktion unterstützen, um soziales Lernen zu ermöglichen.

Theoretische Überlegungen zu agilen Methoden lassen vermuten, dass sie den Spagat zwischen den Zielen professioneller Softwareentwicklung (effiziente Erstellung nützlicher Software von hoher Qualität zur Zufriedenheit des Kunden) und den oben dargestellten Zielen von Schulprojekten müheloser schaffen als lineare Modelle, weist man ihnen doch Eigenschaften wie verstärkte Betonung von Kommunikation und Kooperation sowie individueller Fähigkeiten und Bedürfnisse zu, ebenso die Stärkung der Eigenverantwortung und das Setzen auf motivierte Projektbeteiligte, die von sich aus verantwortungsvoll und couragiert handeln [Be01, Ru04]. Von einem agilen Modell für Schulprojekte könnten demnach Lernende und Lehrende gleichermaßen profitieren zumal eine an das individuelle Projekt angepasste „schlanke“ Vorgehensweise sinnhaft, zeitgemäß und realitätsnah ist. Vereinzelte Untersuchungen in der Vergangenheit bestärken diese Annahmen: Weigend [We05] beschreibt positive erste Erfahrungen bei der Anwendung einzelner agiler Elemente. Meerbaum-Salant und Hazzan [MH10] haben eine Studie durchgeführt, in der sie agile Werte nutzen, um Lehrer bei der Betreuung von Softwareprojekten zu unterstützen. Göttel [Gö12] verwendet einige agile Praktiken in verschiedenen Projekten mit der Intention, soziale Interaktionen in der modernen Softwareentwicklung hervorzuheben und so Lernende ein attraktives Bild der Informatik zu vermitteln.

3 Das agile Framework AMoPCE

Agile Methoden stellen konkrete Vorgehensweisen für die Projektorganisation und Projektarbeit zur Verfügung, die flexibel und nach den individuellen Bedürfnissen eines Projekts ausgewählt werden. Wesentliche innovative Neuerungen gegenüber dem linearen Modell werden im Folgenden kurz dargestellt[7].

---

[2] „Einer hat dann zu Hause den größten Teil programmiert.“ (Interview mit einem Lehrer)

[3] „In jedem Kurs das Gleiche: Die Leute wollen einfach nicht zuhören, wenn es darum geht, zuerst die Struktur einer Website auf Papier zu überlegen.“ [HNR07]

[4] „Obwohl ich nach den Doppelstunden regelmäßig nass geschwitzt war, haben die Schülerinnen und Schüler vor allem die langen Wartezeiten bemängelt, bis Unterstützung kam.“ (Interview mit einem Lehrer)

[5] „Zum Testen sind wir nicht mehr gekommen.“ (Interview mit einem Lehrer)

[6] „Ein paar Fehler sind noch drin, so dass es noch nicht läuft. Aber zum Fertigstellen hatten wir keine Zeit mehr.“ (Interview mit einem Lehrer)

[7] Eine detaillierte Beschreibung insbesondere auch der Anpassung professioneller Elemente an den Schulkontext durch didaktischen Transposition findet man bei Romeike und Göttel [RG12].

Iteratives Vorgehen und Prototypen: Ausgehend von einer Produktvision werden die gewünschten Funktionalitäten in User Stories aus Nutzersicht beschrieben und priorisiert. In den anschließenden Iterationen werden jeweils einige User Stories mit hoher Priorität ausgewählt und die zugehörigen Tasks formuliert, die aus Entwicklersicht bei der Umsetzung der jeweiligen Story zu erledigen sind. In einem (linearen) Mini-Projekt mit Planungs-, Entwurfs- Implementierungs- und Testphase werden dann die Tasks realisiert. Am Ende eines jeden Mini-Projekts steht ein lauffähiger und getesteter Prototyp, der ausprobiert, vorgestellt und bewertet werden kann (vgl. Abb. 1).

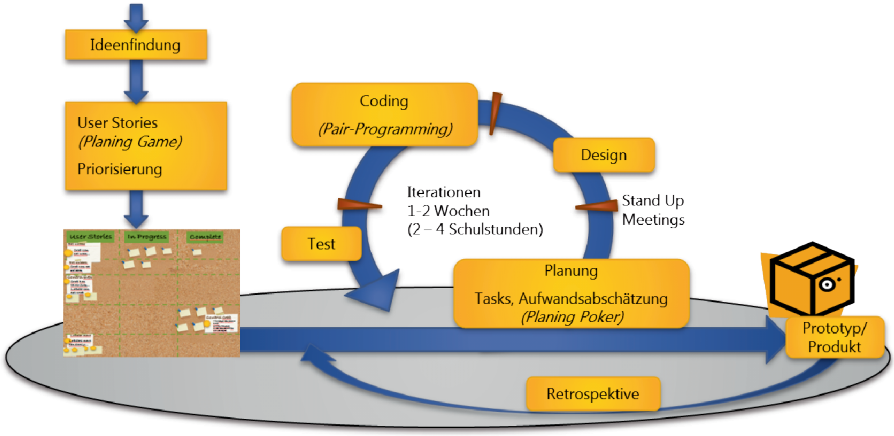


Abb. 1: Ein agiles Framework für Schulprojekte

Solche Mini-Projekte sind wesentlich leichter zu überblicken und sollen den Schülerinnen und Schülern das autonome Planen und Arbeiten erleichtern. Durch die iterative Vorgehensweise können die Projektteams auch mit geringer fachlicher und methodischer Kompetenz beginnen, und Erkenntnisse, die sie durch Reflexion und Peerfeedback gewinnen, in der nächsten Iteration anwenden. Der Abschluss jeder Phase mit einem lauffähigen Prototyp führt zu sichtbaren Ergebnissen, die die Motivation erhalten und erleichtert gleichzeitig die Gesamtplanung durch die Lehrkraft.

Anforderungen als variable Projektgröße: Während in professionellen agilen Projekten in der Regel Zeit, Personal und Produktqualität fixe Größen darstellen, sind Anforderungen flexibel, folgen einer Produktvision und werden durch intensive Zusammenarbeit mit dem Kunden nach und nach konkretisiert. Der Vorgang der Softwareentwicklung wandelt sich: Aus einem stark prozessfokussierten Vorgang wird ein dynamischer und kreativer [FST13], der die Projektbeteiligten zu gestalterischem und verantwortungsvollem Handeln ermutigt. Reichen die Ressourcen nicht aus, werden keine Kompromisse hinsichtlich Qualität (Testen), Zeit (Endtermin) oder mehr Personal eingegangen, sondern niedrig-priorisierte Anforderungen nicht umgesetzt. Diese innovative Herangehensweise erleichtert die Planungsarbeit in Schulprojekten enorm, weil sie Planungsfehler unerfahrener Schüler verzeiht. Darüber hinaus kann die Lehrkraft die Rolle des Kun-

den übernehmen und damit steuernd eingreifen.

Kommunikationsfördernde Praktiken: Einige Praktiken wie Pair-Programming und Stand-Up Meetings, in denen sich die Teammitglieder gegenseitig kurz über Erledigtes, Probleme und Geplantes informieren, fördern die Kommunikation und geben der Lehrkraft einen Einblick in den individuellen Lernfortschritt.

Praktiken, die den Planungs- und Organisationsprozess unterstützen: Durch Praktiken wie das Planning Poker erhalten SuS Handlungsanweisungen, die sie in der schwierigen Phase der Aufwandsabschätzung unterstützen sollen. Regelmäßige Stand-Up Meetings, feste Iterationslängen und Retrospektiven am Ende jeder Iteration strukturieren den Prozess. Das Project Board ist zentraler Informations- und Planungsbereich, an dem der Projektfortschritt visualisiert und begreifbar gemacht wird und auftretende Probleme und Fragen notiert werden können. Es soll Lernenden und Lehrenden einen Überblick über den Stand des Projekts bieten, die Planung der nächsten Schritte erleichtern und helfen, Entscheidungen zu treffen. Im Zusammenspiel sollen diese Praktiken Schülerinnen und Schülern ein selbständiges Arbeiten und Planen ermöglichen und in Konsequenz eine starke Änderung der Lehreraufgaben bewirken.

Die theoretischen Überlegungen zeigen, dass ein agiles Framework Projektarbeit für Lernende und Lehrende im positiven Sinne verändern kann. Neben der begründeten Relevanz und der Konsistenz gilt es in den nächsten Schritten die Praxistauglichkeit sicher zu stellen, um dann – möglichst umfassend und gestützt durch theoretische und empirische Argumente – die Bedingungen zu beschreiben, die zum Gelingen führen. Ein dazu geeignetes Forschungsformat erlaubt es, Lern- und Arbeitsprozesse während des Projektverlaufs bezüglich hinderlicher und fördernder Faktoren zu untersuchen und liefert idealerweise neben dem Erkenntnisgewinn für die Theorie auch ein für die Praxis nützliches Artefakt, das zukünftig eine Verbesserung der Unterrichtspraxis erleichtert.

4 Forschungsgeleitete Adaption und Weiterentwicklung

Als Problem der Unterrichtsentwicklung hat sich gezeigt, dass allein durch theoretische Überlegungen erlangte Erkenntnisse und Modelle oft nur unzureichend zu nachhaltiger Innovation in der Unterrichtspraxis führen [Re05]. Die Gründe hierfür sind vielfältig. So können beispielsweise verschiedenste Probleme bei der Implementierung eines theoretisch erarbeiteten Modells in realen Kontexten auftreten, wie beim Verwenden des adaptierten Wasserfallmodells. Diese können inhärent dem Entwurf innewohnen oder abhängig vom Kontext auftreten. Manchmal fehlt rein theoretisch begründeten Ansätzen auch einfach die Praxisrelevanz. Gleichzeitig ist zeitgemäßer Informatikunterricht auf kontinuierliche didaktische Aufarbeitung der Themen und auf nachhaltige Innovation in der Unterrichtspraxis angewiesen, da Informatik eine dynamische Wissenschaft ist. Beispiele hierfür finden sich in der fortwährenden Entwicklung neuer Software- und Hardwarewerkzeuge zur Verbesserung der Unterrichtspraxis oder der Weiterentwicklung zentraler Unterrichtsgegenstände, wie sie z. B. derzeit beim Thema Datenmanagement, -schutz

und -sicherheit zu beobachten ist. Die agilen Methoden schließlich haben den berufstypischen Prozess der Softwareentwicklung ebenso wie die wissenschaftliche Sicht auf ihn enorm verändert und vermutlich haben sie auch das Potential, Unterrichtsprojekte gewinnbringender zu gestalten. Einer engen Theorie-Praxis-Verschränkung scheint deshalb in der Informatik eine substantielle Bedeutung inne zu wohnen, damit in der Theorie begründete und erarbeitete Entwürfe forschungsgeleitet so weiterentwickelt werden können, dass sie zu nachhaltiger Innovation und konkreter Unterrichtsweiterentwicklung führen.

Ein vielversprechendes Forschungsformat, das hilft generalisiertes Wissen über das Entwerfen und das nachhaltige Implementieren innovativer Unterrichtsmodelle zu entwickeln, ist Design-Based Research (DBR) [De03]. Reed und Guzdial [RG12] halten DBR für die Informatikdidaktik für besonders geeignet, da hiermit Polarisierungen und Diskussionen des „einzigen richtigen Vorgehens“ vermieden werden können und stattdessen Interventionen hinsichtlich der Ursachen erfolgreicher Lernprozesse beschrieben werden. Charakteristische Merkmale des DBR sind die Motivation, Unterrichtspraxis verbessern zu wollen, das Ziel, reale Probleme zu lösen und dabei Erkenntnisse für Theorie und Praxis zu gewinnen, sowie der Stellenwert des Designs als zentraler Teil des Forschungsprozesses. DBR ermöglicht es, verschiedenste Aspekte einer Intervention in unterschiedlichen Kontexten zu untersuchen, um Designprinzipien zu generieren, die durch formative Evaluation der Beobachtungen und Erfahrungen empirisch gestützt sind [CJB04]. Die Arbeit verläuft typischerweise zyklisch und zusammen mit Praktikern, wobei der Entwurf der Intervention flexibel ist – sowohl innerhalb eines Zyklus als auch über den Forschungsprozess hinweg. Deshalb ist insbesondere bei der Analyse und der Reflexion der Beobachtungen und beim Re-Design ein intensiver Austausch zwischen Forschern und Lehrerinnen und Lehrern wichtig.

Aus den oben beschriebenen Gründen sind wir der Meinung, dass die Theorie-Praxis-Verschränkung und eine Zusammenarbeit von Forschern und Lehrkräften auf Augenhöhe für unsere Arbeit besonders wichtig sind. Das Forschungsvorhaben fußt deshalb auf der berufsspezifischen Kompetenz von Lehrkräften, fachliches Wissen, allgemeindidaktisches, fachdidaktisches sowie pädagogisches Wissen in einem Wissensbereich zu integrieren [St10]. Erfahrene Lehrkräfte passen das agile Modell individuell an ihren Kontext an, setzten ergänzend eigene Ideen um, beobachten und sammeln Erfahrungen.

Der DBR-Prozess zu den agilen Methoden gliedert sich in Phasen des Inputs und der Vernetzung (Workshops) sowie Phasen der individuellen Anpassung, Weiterentwicklung und Erprobung durch die Lehrkräfte (vgl. Abb. 2). Wichtigste Funktion der Workshops ist die Bündelung der Beobachtungen und Erfahrungen aus den Implementierungen, um in gemeinsamer Analyse und Reflexion die Erkenntnisse herauszuarbeiten, auf deren Basis die Theorie angepasst und Lösungsansätze und Anregungen für die weitere Verbesserung der Praxis entwickelt werden. Eine Datenerhebung findet in jedem Zyklus statt und umfasst Leitfadeninterviews, exemplarische Projektdokumentationen und -ergebnisse sowie die Ergebnisse des Workshops. Ziel ist es, die Interventionen als Fallstudien bezüglich möglichst vieler beeinflussender Faktoren zu rekonstruieren und die

theoretischen Überlegungen, die Ergebnisse aus den Workshops und die Beobachtungen in den Interventionen auf Konsistenz zu prüfen.

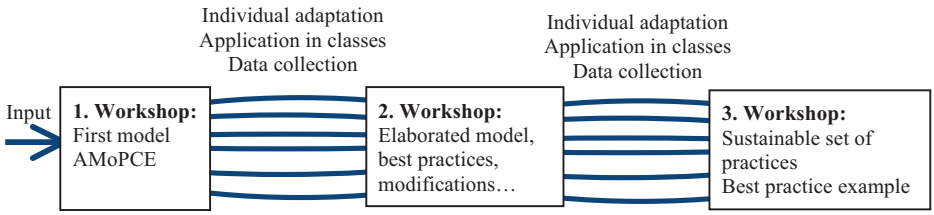


Abb. 2: Forschungsprozess zur Weiterentwicklung von AMoPCE

5 Durchführung des ersten Zyklus und erste Ergebnisse

Im Folgenden beschreiben wir wesentliche Teile der konkreten Umsetzung des ersten Zyklus, insbesondere mit Blick auf die Ausgestaltung der Zusammenarbeit mit den Lehrkräften, eine Einordnung in den Gesamtkontext sowie erste Erkenntnisse.

Die Idee einer Zusammenarbeit von Theoretikern und Praktikern auf Augenhöhe und die aktive Einbeziehung des berufsspezifischen Wissens der Lehrkräfte von Beginn an waren entscheidend für die Gestaltung des ersten Workshops, der zugleich die Funktion einer Fortbildung haben musste, damit die Lehrkräfte nach der Teilnahme in der Lage waren, das agile Modell zu implementieren. Interessierte Lehrkräfte sollten dazu ermutigt werden, das agile Modell in einer an ihren Kontext angepassten Form in einem Unterrichtsprojekt anzuwenden und dabei auch eigene Ideen und Interpretationen einzubauen. Darüber hinaus sollten sie motiviert werden, ihr berufsspezifisches Wissen in einen theorie-praxis-verschränkten Prozess der Weiterentwicklung einzubringen. Am ersten, zweitägigen, Workshop, nahmen elf Lehrerinnen und Lehrer sowie drei Forscher teil. Zur Motivierung, Verdeutlichung der Praxisrelevanz und thematischen Einstimmung beschrieb ein Softwareentwickler zunächst an einem konkreten Projekt, wie in seinem Team professionelle agile Softwareentwicklung umgesetzt wird und begründete deren Sinnhaftigkeit an konkreten, eingängigen Beispielen. Im Anschluss wurden typische Probleme in Schulsoftwareprojekten besprochen, AMoPCE vorgestellt und Möglichkeiten praktischer Umsetzungen der einzelnen Vorgehensweisen des Modells ausführlich und offen diskutiert. In dieser Phase haben die Lehrkräfte aktiv mögliche Schwierigkeiten und Probleme herausgearbeitet, die bei der Implementierung von AMoPCE im Klassenraum auftreten können und erste Ideen zu deren Lösung entwickelt. Nach einer anschließenden praktischen Erprobung wurden zu allen Vorgehensweisen des agilen Modells die diskutierten Punkte fixiert sowie weitere Ideen und Lösungsansätze dazu erarbeitet. Im weiteren Verlauf des ersten Zyklus haben sechs Lehrerinnen und Lehrer aus drei Bundesländern die agilen Methoden in sechs Mittelstufenklassen und drei Oberstufenkursen mit insgesamt ca. 170 SuS umgesetzt. Als Werkzeuge wurden Scratch, Greenfoot, BlueJ und eine Processing-IDE eingesetzt, die Projektlänge variierte von wenigen

Wochen bis hin zu acht Monaten und ebenso unterschiedlich waren die Vorkenntnisse der SuS, die vom Programmieranfänger bis zum fortgeschrittenen Programmierer reichten. Vorrangiges Ziel der ersten Implementierungsphase war es, die Stellen im Prozess zu identifizieren, die den SuS bzw. den Lehrkräften noch generell oder kontextbedingt Probleme bereiten. Ein weiteres Ziel war es herauszufinden, inwiefern sich die theoretischen Überlegungen in der Praxis an Stellen zeigen, an denen die Umsetzung bereits ohne Schwierigkeiten verlief und welche Kontextfaktoren dabei eine Rolle spielen. Auch hier galt es, die Perspektiven der Lernenden und die der Lehrenden zu berücksichtigen. Die Interpretationen, Ideen und Anpassungen der Lehrer stellen einen wesentlichen Schritt hin zu einem an Best Practices orientierten Leitfaden dar, der Lehrer bei Kontextanpassungen unterstützen soll. Diese Aspekte wurden in Leitfadeninterviews erfasst.

Generell zeigte sich in den Interviews, dass die Erfahrungen und Beobachtungen unabhängig vom Kontext motivierend gut mit den theoretischen Überlegungen übereinstimmten. Klare Schwierigkeiten zeigten sich lediglich bei SuS mit wenig Implementierungserfahrung die Java als Programmiersprache verwendeten. Diesen SuS fiel das Formulieren von Tasks schwer, also der Perspektivwechsel vom Nutzer zum Entwickler. Abgesehen von diesem Hemmnis berichten die Lehrer übereinstimmend, dass die Teams mit Hilfe der agilen Praktiken ihre Projektarbeit von Beginn an selbst organisiert haben. So berichtet ein Lehrer zwar von anfänglichen Schwierigkeiten einer Gruppe, die „innerhalb von den ersten 45 Minuten des Unterrichts fünf Stand-Up Meetings brauchte“ aber auch, dass sie rasch lernten „zielgerichtet [zu] planen“. Ein anderer formuliert den Vorteil für SuS so: „Sie haben halt eine Struktur, in der sie ihre Fehler, ihre Probleme lösen können. Ich fand […], dass sie auch mehr Selbständigkeit erlebt haben.“ Weil sie ihnen einen regelmäßigen, guten Einblick in die fachlichen und sozialen Fähigkeiten der SuS boten, schätzten die Lehrer die Stand-Up Meetings auch für sich als sehr wertvoll ein. So berichtet ein Lehrer „ich lerne auch was über die, ja, wie denken die, wie ticken die“ und weiter führt er aus „du siehst Entwicklungen, wo ich der Meinung bin, die hätte ich sonst eben nicht gesehen“ und fügt hinzufügt „ich hab viel mehr Zeit auch, die Schüler zu beobachten“, womit er die geänderte Lehrerrolle anspricht. „Ich fand es total interessant,“ führt ein anderer Lehrer aus, „…also du hängst nicht mitten drin, du musst nicht moderieren, sondern du kannst einfach zuhören. Klar, man kriegt sehr viel mehr mit, darüber wie sie arbeiten, wie sie denken, wie dieser Prozess abläuft.“ Im Zusammenspiel mit dem iterativen Vorgehen konnten sie die Lernfortschritte der SuS erkennen: „Also die Kommunikation und die Kooperation ist von Mal zu Mal intensiver geworden, also von Iteration zu Iteration.“ Sagt ein Lehrer und fügt erklärend an, dass sich immer mehr SuS an fachlich immer tiefergehenden Gesprächen beteiligten: „Wenn du so zuhörst, was da an Problemerörterungen nach dem gegenseitigen Testen lief, ist [das] beeindruckend.“ Auch der organisatorische Aspekt wurde durchweg als hilfreich empfunden. Dazu führt ein Lehrer aus: „[Alle] zusammen gekommen sind wir immer am Ende der Stunde […], da gab es immer diese Vorstellung der Prototypen“. Testen wurde nach Aussagen der Lehrerinnen und Lehrer zu einem normalen Bestandteil der Projektarbeit, der uneingeschränkt positiv von den SuS aufgenommen wurde und keiner weiteren Motivierung bedurfte: „Dann waren die gegenseitigen Tests immer erst – rein aus Motivation: Ich will mal sehen, was die gemacht haben“ berichtet ein Lehrer und kontrastiert die

aktuelle Situation etwas später zu Erfahrungen mit dem Wasserfallmodell; „weil wenn sie überhaupt am Testen sind, ist ja das schon mal was." Ein Lehrer berichtet, dass eine Gruppe von sich aus typische, wiederkehrende Fehler identifizierte „und diese Fälle haben sie sich […] auch aufgeschrieben, auch dokumentiert, welche Fehler es sind. Und die bei jeder Iteration wieder getestet. Also dann schon so ein bisschen Regressionstest". Die Prototypen gaben Gelegenheit zu (Peer-)Feedback, so schwärmt ein Lehrer, dass „dieser Kurs von mir permanent gelobt wurde, was für Schüler ich da habe […] weil die haben sich wirklich auch Mühe gegeben, […] tolle Sachen gebracht".

6 Diskussion

Das Gesamtbild aller Interviews, von dem wir oben einen Ausschnitt skizziert haben, zeigt, dass sich die theoretischen Überlegungen in der Unterrichtspraxis unabhängig vom Kontext widerspiegelten. Die Anpassungen an das spezifische Projekt waren durchweg gelungen und werden in Best Practices festgehalten. Interessant war auch, dass die SuS selbst erkannten, welche Praktiken für sie sinnvoll und hilfreich waren. So hat ein Lehrer beispielsweise das Planning Poker erst während des laufenden Projekts eingeführt, als die SuS schon ein Gefühl für sinnvolle Größen von User Stories und Tasks hatten. Diese SuS haben in der Aufwandsabschätzung keinen planerischen Mehrwert gesehen und die Praktik nach zwei Iterationen aufgegeben – gleiches wurde uns auch von professionellen Softwareentwicklern berichtet. In einem anderen Projekt wurde das Abschätzen am Anfang eingeführt und von den SuS als hilfreich empfunden, wobei sie die Erfahrung aus der ersten Iteration nutzten, um im Folgenden vergleichend abzuschätzen.

Interessant ist auch der Aspekt der Nachhaltigkeit. Ein Lehrer wurde von einer Kollegin gefragt, was er mit seinen SuS im Vorjahr gemacht habe. „Die fangen direkt an zu arbeiten, wenn ich eine Aufgabe stelle. Meine heben erst mal alle den Finger." sagte sie. Möglicherweise hat sich die Selbstwirksamkeit und/ oder die Herangehensweise der SuS an komplexere Aufgaben durch das Projekt verändert.

Wir waren überrascht, mit welcher Leidenschaft und mit welchem Engagement die Lehrerinnen und Lehrer sich beteiligen. Dies manifestierte sich beispielsweise in der mutigen und kreativen Umsetzung, die viel Vorbereitungszeit beanspruchte und unter anderem die Idee der „Student Story" hervorbrachte, ein vom Lehrer zu den User Stories der Schüler hinzugefügter Lernauftrag. Auch die inspirierende Atmosphäre und die engagierte Zusammenarbeit in den Workshops zeigt, wie sehr sich die Lehrerinnen und Lehrer mit dem Thema identifizieren. Möglicherweise ist dieses Format auch geeignet, um Lehrerfortbildungen über Zusammenarbeit auf Augenhöhe nachhaltiger zu gestalten, als durch rein inputorientierte Formate. Im Weiteren ist vorgesehen, diesen Aspekt mit zu betrachten, um festzustellen, ob aus der Anlage und Durchführung des Projekts auch Aussagen getroffen werden können, wie theorie-praxis-verschränkte Lehrerfortbildungen effektiv und nachhaltig durchgeführt werden können. Durch den zyklischen Verlauf und das flexible Design ermöglicht es DBR, diese interessanten, unerwartet aufgetreten Aspekte in einem nächsten Zyklus mit zu untersuchen.

Literaturverzeichnis

[Be01] Beck, Kent; Beedle, Mike; Bennekum, Arie van; et al.: Manifesto for Agile Software Development. http://www.agilealliance.org/the-alliance/, abgerufen am 10.04.2015.

[CJB04] Collins, Allan; Joseph, Diana; Bielaczyc, Katerine: Design Research: Theoretical and Methodological Issues. Journal of the Learning Sciences 13 (2004), Nr. 1, S. 15-42.

[De03] Design-based Research Collective: Design-Based Research: An Emerging Paradigm for Educational Inquiry. In: Educational Researcher vol. 32 (2003), Nr. 1, S. 5–8.

[Fr83] Frey, Karl: Die sieben Komponenten der Projektmethode - mit Beispielen aus dem Schulfach Informatik. In: Log in vol. 3 (1983), Nr. 2, S. 16 – 20.

[FS82] Frey, Karl; Schäfer, Ulrich: Die Projektmethode. Beltz, Basel, 1982.

[FST13] Fuch, Alexander; Stolze, Carl; Thomas, Oliver: Von der klassischen zur agilen Softwareentwicklung. In: Praxis der Wirtschaftsinformatik vol. 290 (2013), S. 17–26.

[Gö12] Göttel, Timo: Agiler Informatikunterricht : Soziale Aspekte der professionellen Softwareentwicklung im Schulunterricht erfolgreich erfahrbar machen. University of Hamburg, 2012.

[Gu14] Gudjons, Herbert: Handlungsorientiert lehren und lernen. Verlag Julius Klinkhardt, Bad Heilbrunn, 2014.

[HNR07] Hartmann, Werner; Näf, Michael; Reichert, Raimond: Informatikunterricht planen und durchführen. Springer, 2007.

[Hu05] Humbert, Ludger: Didaktik der Informatik. Teubner, 2005.

[MH10] Meerbaum-Salant, Orni; Hazzan, Orit: An Agile Constructionist Mentoring Methodology for Software Projects in the High School. In: ACM Transactions on Computing Education vol. 9 (2010), Nr. 4, S. 1–29.

[Re05] Reinmann, Gabi: Innovation ohne Forschung? Ein Prädoyer für den Design-Based Research-Ansatz in der Lehr-Lernforschung. In: Unterrichtswissenschaft vol. 33 (2005), Nr. 1, S. 52–69.

[RG12] Romeike, Ralf; Göttel, Timo: Agile Projects in High School Computing Education – Emphasizing a Learners' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE'12). ACM, Hamburg, 2012.

[Ru04] Rumpe, Bernhard: Agile Modellierung mit UML. Codegenerierung, Testfälle, Refactoring. Springer, 2004.

[SS11] Schubert, Sigrid; Schwill, Andreas: Didaktik der Informatik. Springer, 2011.

[SS05] Schneider, Daniel K; Synteta, Paraskevi: Conception and implementation of rich pedagogical scenarios through collaborative portal sites, 2005. http://tecfa.unige.ch/proj/seed/catalog/docs/schneider-icool-final.pdf, abgerufen am 10.04.2015.

[St10] Strunz-Maireder, Edith: Pedagogical Content Knowledge. In: wissenplus vol. 28 (2010), Nr. 5, S. 41–44.

[We05] Weigend, Michael: Extreme Programming im Klassenraum. In: S. Friedrich (ed.): INFOS 2005. Dresden: GI- Edition - Lecture Notes in Informatikcs [LNI], 2005.

Jens Gallenbacher (Hrsg.): INFOS 2015 – Informatik allgemeinbildend begreifen
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 211

Agiler Informatikunterricht als Anfangsunterricht

Lennard Kerber[1], Petra Kastl[2] und Ralf Romeike[2]

Abstract: Agile Methoden unterstützen die Organisation und Durchführung von Softwareentwicklungsprojekten und finden inzwischen breite Anwendung im professionellen Bereich. Auch für die Schule sind sie vielversprechend. In diesem Bericht wird beschrieben, wie mit Hilfe adaptierter agiler Methoden Anfangsunterricht in der Programmierung methodisch neu in Form eines „geskripteten Projekts" gestaltet werden kann. In einer ausgearbeiteten Unterrichtssequenz lernen die SchülerInnen selbstreguliert mit dafür erstelltem Material und wenden neu erworbene Wissensbausteine jeweils bei der schrittweisen Weiterentwicklung eines Geschicklichkeitsspiels an. Agile Methoden unterstützen die SchülerInnen dabei unmittelbar und sinnvoll in ihrem Lern- und Arbeitsprozess. Gleichzeitig erwerben die SchülerInnen durch das Einbinden agiler Methoden in die Unterrichtssequenz indirekt bereits eine Methodenkompetenz, die sie in späteren Projekten benötigen. Es werden insbesondere die Anpassungen der agilen Vorgehensweise an den Kontext eines „geskripteten Projekts" und eigene Ergänzungen sowie Beobachtungen und Erfahrungen beschrieben.

Keywords: Agile Methoden, agile Praktiken, Anfangsunterricht, Unterrichtsprojekt

1 Einleitung

Die Verwendung agiler Methoden in professionellen Projekten nimmt seit 2010 erheblich zu [Ko14], denn agile Methoden schaffen Raum für das gestalterische Moment in der Softwareentwicklung, das für manche Projekte sehr wichtig ist [FST13], und wirken sich sehr positiv auf die Motivation der Mitarbeiter und die Kooperation im Team aus [Ko14]. Auch für Schulprojekte sind agile Methoden vielversprechend [RG12]. Im Rahmen des Projekts Agile Methoden im Informatikunterricht (AMI) entwickeln Forscher der FAU Erlangen-Nürnberg zusammen mit engagierten LehrerInnen ein agiles Modell für die Praxis im Informatikunterricht weiter [KR15]. Im Zentrum der Untersuchung stehen Projekte in denen SchülerInnen selbstorganisiert arbeiten und vielfältige fachliche, methodische und soziale Fähigkeiten und Fertigkeiten anwenden und weiterentwickeln – also eigentlich kein Anfängerunterricht. Umso spannender ist das Vorhaben, Anfangsunterricht unter Verwendung der textbasierten Programmiersprache Processing und unter Einbindung sinnvoll gewählter agiler Praktiken methodisch neu zu gestalten, um eine Alternative zu schaffen zu einer langen Lernzeit am Anfang und einer kurzen Projektzeit am Ende, die unter enormem Zeitdruck steht.

Unterricht für Programmieranfänger in einer textbasierten Programmiersprache zu struk-

[1] Otto-Nagel-Gymnasium, Schulstr. 11, 12683 Berlin, l.kerber@otto-nagel-gymnasium.de
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, petra.kastl@fau.de, ralf.romeike@fau.de

turieren ist eine Gratwanderung. Stark geführtes, kleinschrittiges Vorgehen mit kurzen praktischen Übungen erzieht zu Unselbständigkeit, schult einseitig die Programmierfertigkeiten und demotiviert leistungsstarke SchülerInnen. Längere Übungsphasen am Rechner dagegen sind schnell ineffektiv und demotivieren Leistungsschwächere, denn sie gehen für viele SchülerInnen mit langem Warten auf Unterstützung einher. Eine Modellierung vorweg ist generell kaum motivierbar, weil ihr Sinn bei kleinen Aufgaben nicht erkennbar ist. Um diese Probleme ein Stück weit aufzulösen, werden beispielsweise Lernaufgaben vorgeschlagen [HNR07] oder das schrittweise Modellieren, Entwickeln und Ausgestalten eines kleinen Spiels über einige Wochen oder Monate hinweg [St09], wobei sich meist Phasen des lehrerzentrierten Unterrichts mit Phasen der Schüleraktivität am Computer abwechseln. Für LehrerInnen bleibt es einer der anstrengendsten Unterrichte.

Dieser Artikel berichtet vom Versuch, Anfangsunterricht methodisch anders zu gestalten. Die Idee war, die SchülerInnen bereits zu einem frühen Zeitpunkt in einem „geskripteten Projekt“ selbstreguliert lernen zu lassen und gezielt einige agilen Praktiken zu nutzen, die die SchülerInnen innerhalb des vorgezeichneten Projekts sinnvoll unterstützen, indem sie beispielsweise einen klaren Organisationsrahmen oder Interaktivität fördernde Handlungsanweisungen vorgeben. Auf der dabei indirekt erworbenen Methodenkompetenz kann im Anschluss in agilen (Softwareentwicklungs-) Projekten aufgebaut werden. Darüber hinaus adressiert das Vorgehen eine Herausforderung der Projektarbeit: Begriffe, Vorgehensweisen und Kommunikationsformen in einem Softwareprojekt müssen in der Regel selbst Gegenstand des Unterrichts werden, ehe sie von den SchülerInnen aktiv angewandt werden können. Das vorliegende Beispiel vermeidet theorielastiges „Lernen auf Vorrat“, indem Terminologie und Aufbau eines Softwareprojekts implizit im Unterricht mit „eingeschliffen“ werden.

2 Rahmenbedingungen

An dem „geskripteten Projekt“ arbeiteten 20 SchülerInnen eines Berliner Wahlpflichtkurses über 12 Wochen jeweils eine Doppelstunde pro Woche. Die Gruppe setzte sich aus verschiedenen 9. Klassen zusammen und es bildeten sich für das Projekt fünf Teams zu je vier SchülerInnen. Für die SchülerInnen war es ihr erster Informatikunterricht. Gemäß internem Curriculum besprachen wir zunächst Algorithmen des Alltags in umgangssprachlicher Formulierung und erarbeiteten dann Kontrollstrukturen mit Robot Karol. Mit Processing[3] setzten sich die SchülerInnen zum ersten Mal im Rahmen des geskripteten Projekts auseinander.

3 „Processing ist eine [...] stark typisierte Programmiersprache [... die] für die Einsatzbereiche Grafik, Simulation und Animation spezialisiert [ist]. Processing hat den Charakter einer stark vereinfachten Version der Programmiersprache Java [...] und richtet sich vorwiegend an Gestalter, Künstler und Programmieranfänger.“

(de.wikipedia.org/wiki/Processing)

Der Computerraum bot mit 14 außen stehenden Computern, von denen 10 genutzt wurden, und einem großen Tisch in der Mitte für Besprechungen im Plenum eine gute Aufteilung sowie genug Raum für Diskussionen in den Teams und ausreichend Abstand zu den anderen Teams.

3 Gestaltung des Lehr-Lernarrangements

Das über die gesamte Einheit des „geskripteten Projekts“ tragende Thema war die Entwicklung eines Geschicklichkeitsspiels, in dem eine Scheibe mit der Maus durch ein 2D-Labyrinth gesteuert wird (vgl. Abb. 1). Wenn die Scheibe zu dicht an die Wände des Labyrinths kommt, wird sie kleiner, die Größe der Scheibe im Ziel bestimmt die erreichte Punktzahl.

Abb. 1: Demoversion der Lehrkraft (links) und unterschiedlich aufwändige Schülerversionen

Nach einer einführenden Doppelstunde, in der ich eine eigene Implementierung des Spiels vorgestellt und unsere agile Vorgehensweise erklärt hatte, waren vier Iterationen, also Zeitfenster fester Länge, geplant. Von diesen vier Iterationen konnten letztendlich aus schulischen Gründen nur drei durchgeführt werden. Jede Iteration war drei Doppelstunden lang und am Ende stellten die SchülerInnen jeweils ihre weiterentwickelten, lauffähigen Prototypen und ein dazu passendes Benutzerhandbuch vor. Jede der drei Iterationen umfasste auf Arbeitsblättern vorgegebene „Student Stories“ und „User Stories“ mit ihren Tasks. Die Student Stories waren vorgegebene Lernaufträge in Form von User Stories (vgl. Abb. 2), die in dieser Iteration erfüllt werden mussten, die User Stories enthielten die in dem gegebenen Zeitfenster zu implementierenden Funktionalitäten des Spiels aus Benutzersicht zusammen mit ihren Tasks, bei deren Umsetzung die neu erlernten Konzepte angewandt wurden.

Inhaltlich war das Lehr-Lernarrangement so gestaltet, dass die SchülerInnen am Ende der ersten Iteration Rechtecke in selbst gewählten Farben zeichnen und damit ihr Spielfeld individuell gestalteten konnten. Sie haben sich dazu mit der *Leinwand* des Processing-Systems und dem Thema *RGB-Farbraum* vertraut gemacht. In der zweiten Iteration wurden die Konzepte *Funktion* (ohne Parameter und ohne Rückgabewert) sowie *Variable* erarbeitet und zur Weiterentwicklung des Spiels verwendet, beispielsweise beim Zeichnen der Rechtecke bzw. der Spielfigur. In dieser Iteration wurde Refactoring, also

die Umstrukturierung von Code ohne Änderung der Funktionalität, genutzt, um bestehenden Code mit Hilfe der neu erlernten Konzepte besser zu gestalten. Am Ende der dritten Iteration schließlich implementierten die SchülerInnen auch *Funktionen mit Rückgabewert* und *Funktionen mit Parametern.* Damit konnte der Spielplan mit einer bewegten Spielfigur, die dem Mauszeiger folgt, programmiert werden. Die Hindernisse waren für die nächste Iteration vorbereitet, in der die Spielfigur bei Berührung verkleinert wird.

Eine Modellierung wurde nicht verlangt, sie wird bei uns erst im zweiten Lernjahr eingeführt, wenn die Beispiele etwas umfangreicher sind. Aber die SchülerInnen mussten Schnittstellen, Variablen und Funktionen nach einem vorgegebenen Schema im Quellcode dokumentieren.

4 Einbindung unterstützender agiler Methoden

Jede Doppelstunde begann mit einem **Stand-Up Meeting**, in dem sich die SchülerInnen jeder Gruppe kurz und knapp klar machen sollten, was sie in der letzten Doppelstunde geschafft haben und was sie für diese Stunde planen. Das heißt, es gab durch mich zwar immer eine Begrüßung, aber die Wiederholung, die Besprechung des Fortschritts und die Sicherung fand individuell in den Gruppen statt und orientierte sich am jeweiligen Stand der einzelnen Gruppe im Lernprozess und im Projekt. Indem die Stand-Up Meetings kurze individuelle Rekapitulationen am Stundenanfang automatisch anstießen, unterstützten sie die Organisation des selbstregulierten Lernens ideal. Hilfreich bei der Rekapitulation waren die Arbeitsblätter der vorangegangenen Stunden und das gruppeneigene **Project Board**, das den Stand der Spieleentwicklung visualisierte. Nach dem Stand-Up Meeting arbeiteten die SchülerInnen jeweils an der Student- bzw. User Story weiter, bei der sie aktuell standen.

Student Stories sind ein von mir für das geskriptete Projekt ergänzend eingeführtes Artefakt. Dabei handelt es sich um Lernaufträge, die von allen SchülerInnen bearbeitet werden mussten. Zu einem Lernauftrag gab es meist eine Lesekarte, auf der neue Theorie stand, zum Teil auch Material, das vom Lehrertisch abgeholt werden musste oder eine kleine Besprechung des Themas mit mir. Bei der Student Story zum Variablenkonzept beispielsweise (vgl. Abb. 2) kamen die Schülergruppen, die so weit waren, zu mir an den Lehrertisch, zu einer kurzen Sitzung. Zur Veranschaulichung des Speicherkonzepts verwendete ich Streichholzschachteln, die den SchülerInnen als Zusatzmaterial auch für die weitere Arbeit an dem Thema zur Verfügung standen. Wenn es zur Theorie praktische Übungen am Computer gab, arbeiteten die SchülerInnen zu zweit als Programming-Pairs zusammen. Beim **Pair Programming**, das den Wissenstransfer unterstützt und planloses Hacken verhindert, ist ein Schüler Driver, er bedient die Tastatur und erklärt, was er sich bei der Programmierung denkt. Der andere Schüler übernimmt die Rolle des Navigators und überlegt sich, ob es eine bessere oder elegantere Lösung gibt. Die Rollen sollten bei uns regelmäßig alle 5 Minuten gewechselt werden.

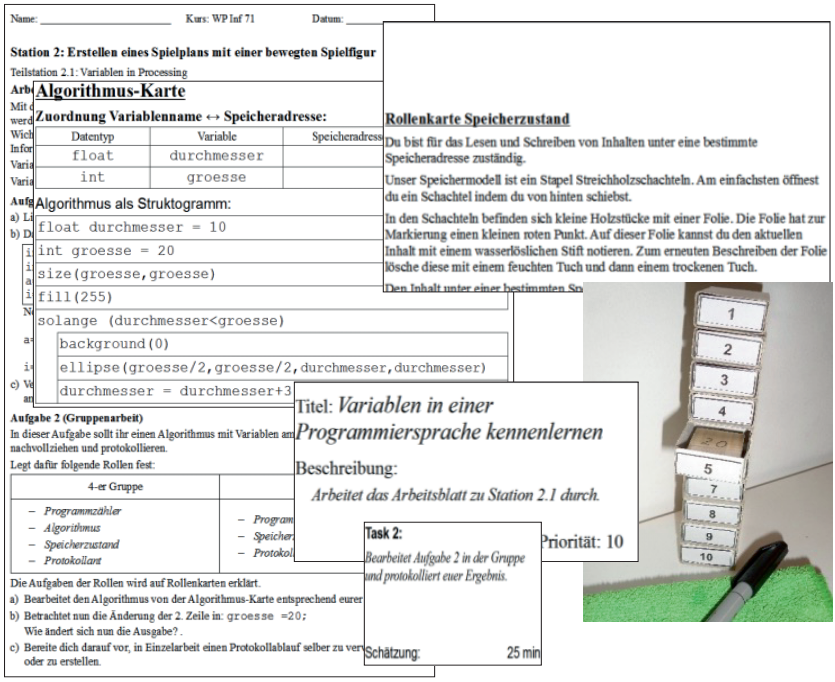


Abb. 2: Student Story zum Thema Variablen

Die **User Stories** dienten der Weiterentwicklung des Geschicklichkeitsspiels unter Verwendung des neu Gelernten. Bei ihrer Implementierung benutzten die SchülerInnen ebenfalls Pair Programming, wobei die beiden Paare eines Vierer-Teams dann teilweise auch an unterschiedlichen User Stories arbeiteten und ihren Code vor dem Test zusammenführten. Zur Koordination und Planung der Spieleentwicklung stand jeder Gruppe ein eigenes **Project Board** zur Verfügung. An dieses waren kleine Zettel mit allen User Stories und den zugehörigen **Tasks** gepinnt, die in den Arbeitsblättern der Iteration enthalten waren. Im Verlauf der Projektarbeit wurden die Tasks nacheinander entsprechend ihrem aktuellen Status in die Spalten „In Progress“ bzw. „Done“ des Project Boards verschoben. Wenn alle Tasks einer User Story erledigt waren und damit die entsprechende Funktionalität im Spiel implementiert war, wurde auch die User Story in die Spalte „Done“ verschoben. Einige User Stories bzw. Tasks stießen **Refactoring**-Tätigkeiten an, die keine neue Funktionalität zum Spiel hinzubrachten sondern dazu dienten, den besehenden Code mit Hilfe neu erlernter Konzepte besser zu strukturieren. Refactoring-Aufgaben wurden ebenfalls von Programming-Pairs durchgeführt und ihr Status am Project Board visualisiert.

Wie weiter oben beschrieben, fanden all diese Tätigkeiten innerhalb von **Iterationen** statt und die Reihenfolge der Bearbeitung war in den Arbeitsblättern vorgegeben. Zur zeitlichen Planung einer Iteration konnten die SchülerInnen das **Planning Poker** nutzen, d. h. für jede anstehende Aufgabe machte jedes Gruppenmitglied zunächst verdeckt eine Zeitschätzung, nach dem Aufdecken mussten stark nach oben oder unten abweichende Schätzungen begründet werden. Am Ende musste sich die Gruppe einigen, beispielsweise auf einen Mittelwert oder auch auf einen Extremwert, bei einsichtiger Begründung.

Am Ende jeder Iteration wurden die **Prototypen,** also die lauffähigen Zwischenstände des Spiels im Plenum besprochen. Jedes Mal stellte dazu ein anderes Gruppenmitglied den jeweiligen Prototypen und das Benutzerhandbuch vor und erläutert sie. In diesen Runden gaben die Mitschüler und ich regelmäßig Feedback, was eine hervorragende Motivation für die weitere Arbeit war. Da wir zur Besprechung immer alle im Plenum versammelt waren und da zu diesem Zeitpunkt stets alle den gleichen Stand hatten, nutzte ich die Runden auch zur Nachgestaltung der Fachsprache oder zur Sicherung zentraler Inhalte. Am Ende jeder Plenumsrunde wurde der Ausblick besprochen. So richteten die SchülerInnen regelmäßig den Blick auf das gesamte Projekt, das Erreichte, das unmittelbar Anstehende und das Ziel, ehe sie in der nächsten Iteration wieder an kleinen Teilschritten arbeiteten.

Die feste, überschaubare Dauer einer Iteration und die sich daran anschließende Prototypenbesprechung gaben dem Lehr-Lernarrangement eine verbindliche zeitliche Taktung und bildeten für die SchülerInnen einen hilfreichen Rahmen zur Orientierung.

5 Erfahrungen und Bewertung

Das agile Vorgehensmodell konnte für eine längere Phase selbstorientierten Lernens bei ProgrammieranfängerInnen gewinnbringend angewandt werden. Die genutzten agilen Praktiken unterstützten die Organisation und Planung, soziales Lernen, die Motivation und das Beurteilen der (Teil-)Produkte optimal. Der selbstregulierte Erwerb fachlicher Kompetenzen im Bereich der Programmierung wurde ergänzt durch die Weiterentwicklung wichtiger sozialer Kompetenzen und ganz nebenbei haben die SchülerInnen eine typische Arbeits- und Vorgehensweise beim Entwickeln von IT-Systemen erlernt und intuitiv reflektiert. Idealerweise folgt dann im selben oder im darauf folgenden Lernjahr ein „richtiges" Unterrichtsprojekt. (Bei mir war es für das Folgejahr geplant, kam aber leider wegen meines Schulwechsels nicht zu Stande.)

Interessant war, dass die SchülerInnen fast alle Praktiken mit Begeisterung aufgenommen haben, ihren Sinn und Nutzen intuitiv reflektierten und sie im späteren Verlauf von sich aus passend bzw. angepasst einsetzten oder wegließen.

Die Project Boards beispielsweise wurden anfangs fortwährend aktualisiert. Vermutlich wegen der Skriptung, der kleinen Gruppengröße, dem geringen Anteil arbeitsteiliger Aufgaben und/oder dem Standort des Boards direkt neben dem Rechner erlebten die

SchülerInnen es mit der Zeit jedoch als unwichtig, dass das Project Board zu jedem Zeitpunkt den aktuellen Stand zeigte. So haben sie später meist erst am Ende einer Doppelstunde die Zettel umgehängt – um zu sehen, was sie geschafft haben und zur Orientierung in der nächsten Stunde. Auch für mich war der Einblick in den Arbeitsstand am Stundenende ausreichend und hilfreich. Die Stand-Up Meetings behielten die SchülerInnen bei, wobei mehr und mehr im Sitzen durchgeführt wurden. Da die Besprechungen trotzdem kurz und zielgerichtet blieben, habe ich es dabei belassen. Refactoring wurde von den Schülerinnen als „Praktik der Profis" akzeptiert und es motivierte den Umbau des Codes, der mit dem schrittweisen Einführen der Lerninhalte verbundenen war. Pair Programming war die einzige Praktik, bei der ich steuernd eingreifen musste, damit der Wechsel der Rollen eingehalten wurde, denn die Beobachtung zeigte, dass sonst meist die Stärkeren programmieren und die Schwächeren zusehen, weil sie sich in der entsprechenden Rolle wohlfühlen. Für mich war es allerdings nicht leicht, konsequent für einen Wechsel zu sorgen. Gleichwohl war es mir wichtig, weil mir diese aus der Praxis stammende Praktik, und insbesondere die Definition der Rollen auch sinnvoll und gewinnbringend erscheint. Das Planning Poker wurde ein, zwei Mal ausprobiert und dann weggelassen, weil es keinen Mehrwert brachte, denn Probleme bei der selbständigen Zeiteinteilung innerhalb einer Iteration gab es nicht. Die Aufgaben einer Iteration waren ja von mir fest vorgegeben und so geplant, dass auch die Schwächeren sie in der gegebenen Zeit bearbeiten konnten. Die Idee bei der Einführung des Planning Poker war auch mehr, dass die SchülerInnen es kennenlernen, um dann selbst zu entscheiden, ob sie es verwenden wollen.

Besonders bereichernd war für mich allerdings das Prototyping, wobei hier verschiedene Aspekte zu meiner sehr positiven Erfahrung beitragen. Beim Vorgehen nach dem Wasserfallmodell liegt erst am Ende ein Produkt vor. Was, wenn eine Gruppe kein lauffähiges Produkt abgibt? Eine schwierige Situation für die Benotung, aber auch unbefriedigend für die Schülergruppe. Durch das Prototyping löste sich diese Situation auf, weil ich früh und regelmäßig Einblick in Zwischenstände hatte und steuernd eingreifen konnte. Darüber hinaus kann ein lauffähiges Zwischenprodukt immer auch benotet werden. In diesem Fall war angekündigt, dass jede Prototypenvorstellung benotet wird. Dazu gab es jeweils klar kommunizierte Anforderungen und Kriterien an den Prototypen und das Benutzerhandbuch und daraus resultierend dann jeweils Teilnoten, die die SchülerInnen einsahen, weil sie transparent und nachvollziehbar zustande kamen. Insbesondere kann aber zu jedem lauffähigen Prototypen Rückmeldung gegeben werden. Ich empfand es als sehr positiv, dass ich so oft die Möglichkeit hatte, zu loben. Auch die SchülerInnen schätzten die Gelegenheit, sich gegenseitig Feedback zu geben. Beides steigerte die Motivation der SchülerInnen sehr und der Ansporn, das Produkt weiter zu entwickeln und erneut zu präsentieren, war bei allen groß. Die SchülerInnen gaben sich wirklich Mühe und es wurden immer tolle Prototypen vorgestellt. Ich konnte beobachten, dass schnelle Gruppen in der verbleibenden Zeit einer Iteration entsprechend der von mir gegebenen Bewertungskriterien mit viel Engagement ihr Spiel ausschmückten und ihr Benutzerhandbuch sehr ansprechend gestalteten. Aber auch schwächere Gruppen kamen mit Fleiß und Einsatz immer zu einem lauffähigen Produkt, für das sie Anerkennung

erfuhren. Insbesondere hat sich hier der Vorteil der agilen Softwareentwicklung bestätigt: Obwohl aus schulischen Gründen auf eine Iteration verzichtet werden musste hatten die SchülerInnen ein Produkt, auf das sie stolz waren und woran sie die Sinnhaftigkeit des neu Erlernten erkannten

Das permanente Loben hat möglicherweise auch dazu geführt, dass ein sonst allgemein auffälliger Schüler bei mir tolle Beiträge lieferte und ein unproblematisches Verhalten zeigte. Positiv überrascht war ich dann aber doch, als eine Kollegin mich fragte, was ich mit meinen Schülern gemacht habe. „Wenn ich ihnen eine Aufgabe gebe“, berichtete sie, „beginnen deine Schüler zu arbeiten, während meine sich erstmal alle melden und Fragen stellen“. Offenbar führten die Projekterfahrungen zu einer nachhaltig selbständigeren Arbeitsweise.

Literaturverzeichnis

[FST13] Fuchs, A.; Stolze, K.; Thomas, O.: Von der klassischen zur agilen Softwareentwicklung. In Praxis der Wirtschaftsinformatik, 2013, 290; S. 17–26.

[HNR07] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer, Berlin, 2007.

[Ko14] Komus, A.: Status Quo Agile 2014. Zweite Studie zu Verbreitung und Nutzen agiler Methoden. Ergebnisbericht.

[KR15] Kastl, P.; Romeike, R.: Entwicklung eines agilen Frameworks mit Design Based Research. In (NN Hrsg.): INFOS 2015 - 16. GI Fachtagung Informatik und Schule, 2015.

[RG12] Romeike, R.; Göttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners’ Perspective. In (Knobelsdorf, M.; Romeike, R. Hrsg.): Proceedings of the 7th Workshop in Primary and Secondary Computing Education. ACM, New York, NY, USA, 2012; S. 48–57.

[St09] Staatsinstitut für Schulqualität und Bildungsforschung ISB Hrsg.: Informatik am naturwissenschaftlich-technologischen Gymnasium Jahrgangsstufe 10. [Erläuterungen und Materialien für Lehrkräfte ; mit CD]. Kastner, Wolnzach, 2009.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 219

Ganzjähriger Projektunterricht mit agilem Framework

Ulrich Kiesmüller[1], Petra Kastl[2] und Ralf Romeike[3]

Abstract: In diesem Beitrag werden zwei jeweils achtmonatige Unterrichtsprojekte zweier 10. Klassen eines bayerischen Gymnasiums vorgestellt. Über die gesamte Zeit entwickelten Gruppen von je fünf bis neun Schülerinnen und Schülern mit der Programmiersprache Java ihr eigenes Softwareprojekt und erarbeiteten sich dabei informatische Konzepte der objektorientierten Programmierung und Modellierung. Zur Unterstützung wurden geeignete agile Praktiken ausgewählt und jeweils zeitverzögert durch weitere ergänzt. Die vorgenommene Anpassung des agilen Modells an den Kontext, die praktische Umsetzung und Beobachtungen werden im vorliegenden Beitrag beschrieben. Sie werden kontrastiert zu den Erfahrungen aus den Vorjahren, in denen nach dem Wasserfallmodell vorgegangen wurde. Abschließend werden wesentliche Erkenntnisse und Erfahrungen, die in die Weiterentwicklung des agilen Modells fließen, zusammengestellt.

Keywords: Einsatz agiler Methoden der Softwareentwicklung im Informatikunterricht, Projektunterricht

1 Objektorientierung und Softwareentwicklung im Unterricht

Grundlagen der objektorientierten Modellierung und Programmierung sind für die naturwissenschaftlich-technologische Ausbildungsrichtung an bayerischen Gymnasien im Lehrplan der 10. Jahrgangsstufe [IS03] verankert. Als Abschluss ist dort ein kleines Softwareprojekt vorgesehen, um den Lernenden zu vermitteln, dass man umfangreiche Aufgaben nur mit sorgfältig geplanter Teamarbeit, strukturiertem Vorgehen und basierend auf fachlichem Wissen lösen kann. Hierbei geben der bayerische Lehrplan und die gängigen Schulbücher dem Wasserfallmodell den Vorzug. Über diese vom Lehrplan geforderten Ziele hinaus sind mir als Lehrkraft auch die Berücksichtigung von Aspekten wie selbstständiges Arbeiten, Kreativität und kritische Reflexion [HNR07] wichtig. Außerdem soll das Projekt Sozialkompetenzen wie Kommunikation und die Fähigkeit zur Entscheidungsfindung in einer Projektgruppe sowie einen konstruktiven Umgang mit Konflikten [Gu08] fördern. Überdies möchte ich den Lernenden mit der Projektarbeit ein modernes, zeitgemäßes Bild vom Beruf eines Informatikers vermitteln [GR13].

In der praktischen Umsetzung erlebte ich wiederholt, dass Schülerinnen und Schüler am Ende der 10. Jahrgangsstufe in Projekten zwar sehr motiviert sind, aber in den zehn Unterrichtsstunden, die der Lehrplan vorsieht, keine brauchbaren oder gar spannenden Produkte gemeinsam planen, entwickeln, vorstellen und reflektieren können. Dafür ist die

[1] Simon-Marius-Gymnasium Gunzenhausen, Simon-Marius-Straße 3, 91710 Gunzenhausen, kiesmueller@simon-marius-gymnasium.de
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, petra.kastl@fau.de
[3] Friedrich-Alexander-Universität Erlangen-Nürnberg, Didaktik der Informatik, ralf.romeike@fau.de

Zeit zu knapp bemessen. Deshalb habe ich in den letzten Jahren die Projektphase erheblich ausgedehnt und weitere Lehrplaninhalte integriert. Für den dazu notwendigen fachlichen Input wurde die Projektarbeit der Schülerinnen und Schüler regelmäßig meist zu Beginn jeder Doppelstunde unterbrochen. Problematisch bei diesem Vorgehen war, dass die Lernenden zu Beginn des Projekts weder über die fachlichen, planerischen und sozialen Fähigkeiten und Fertigkeiten verfügten, die ein lineares Prozessmodell wie das Wasserfallmodell voraussetzt, noch die Schritte einer strukturierten Vorgehensweise in der Softwareentwicklung kannten. Deshalb waren die Jugendlichen immer stark auf Unterstützung angewiesen. Für die Lehrkraft bedeutete es einen enormen Betreuungsaufwand, wobei die Unterstützung noch anspruchsvoller wurde, wenn jede Projektgruppe ihr eigenes Thema wählen konnten. Entsprechend langsam war der beobachtbare Projektfortschritt. Andererseits erhöhte ein eigenes Thema die Motivation und das Durchhaltevermögen der Lernenden insbesondere in der langen Planungsphase. Und so waren die Rückmeldungen der Lernenden am Ende des Schuljahres zwar großteils positiv und sie waren stolz auf ihr Produkt. Das lange Warten auf Unterstützung wurde aber regelmäßig von den meisten Projektbeteiligten bemängelt.

Im Einsatz agiler Methoden sah ich die Möglichkeit, es den Lernenden durch iteratives Vorgehen zu ermöglichen, mit vorhandenen fachlichen, sozialen sowie organisatorischen Fähigkeiten und Fertigkeiten, unterstützt durch ausgewählte agile Praktiken und Artefakte, selbstorganisiert loszulegen und ihre Kompetenzen in jeder Iteration auszubauen. Die Rolle der Lehrkraft wandelte sich hierbei vom „Fragenbeantworter“ und „Fehlersucher“ hin zu 50% Coach und 50% Beobachter (siehe Kapitel 5). Sehr gut konnte ich den fachlichen Lernfortschritt und die positive Entwicklung sozialer und organisatorischer Fähigkeiten bei den Schülerinnen und Schülern beobachten. Von zu langen Phasen des Wartens auf Unterstützung war nicht mehr die Rede. Obwohl ich kaum noch helfen musste, gaben die Lernenden in der Rückmeldung an, dass sie sich gut betreut fühlten.

2 Rahmenbedingungen

In diesem Artikel werden Erfahrungen aus zwei 10. Klassen eines naturwissenschaftlich-technologischen Gymnasiums (NTG) in Bayern vorgestellt. Die Klassen hatten beide ca. 25 Schülerinnen und Schüler und wurden jeweils in Doppelstunden unterrichtet, die im selben Computerraum mit 20 Einzelrechnern stattfanden. Die Größe des Raumes bot jeder Gruppe Platz für ihr Project Board sowie die davor stattfindenden Diskussionen und Planungen, ohne dass sich die Schülergruppen dabei gegenseitig störten. Die Vorkenntnisse im Bereich der Objektorientierung aus der 6. Jahrgangsstufe und und der Algorithmik aus der 7. Jahrgangsstufe waren bei den meisten Schülerinnen und Schülern gering. Allerdings verfügten beide Klassen über einige Projekterfahrung, weil sie in den vorangegangenen Jahren mehrere Projekte erfolgreich durchgeführt hatten. In der 10. Jahrgangsstufe gilt es wie oben angeführt die Grundlagen der objektorientierten Modellierung und Programmierung unter Verwendung von Java zu vermitteln. Als Entwicklungsumgebung wurde dabei BlueJ [KB09] verwendet. Als weitere Hilfsmittel standen den Lernenden das Buch *Java ist*

auch eine Insel [Ul14] zur Verfügung sowie die Java-Klasse ZEICHENFENSTER[4], die es erlaubt, einfache geometrische Objekte in einigen wenigen Farben graphisch darzustellen.

3 Einsatz und Anpassung eines agilen Frameworks

3.1 Agile Praktiken in der Vorbereitung des Projekts

Vor dem Projektstart werden die grundlegenden Voraussetzungen bezüglich der objektorientierten Modellierung und Programmierung sowie des Einsatzes geeigneter Werkzeuge vermittelt. Parallel dazu werden erste Elemente des agilen Modells für Projekte im Informatikunterricht (AMoPCE) [RG12] angepasst und in den Unterrichtsverlauf integriert. In diesem Kapitel beschreibe ich das Vorgehen und dahinter stehende Intentionen exemplarisch. Nach einem Theorie-Input zur Erstellung von Klassendiagrammen erhalten die Schülerinnen und Schüler eine Aufgabe, die sie in Kleingruppen kooperativ planen und modellieren. Da sie später im Projekt mit der Klasse ZEICHENFENSTER arbeiten werden, bieten sich Themen wie das Erstellen eines Szenenhintergrunds oder von Figuren für ein Spiel an, die aus einfachen geometrischen Objekten zusammengesetzt sind. Die Planung erfolgt in einem **Stand-Up-Meeting** vor dem **Project Board** der Gruppe, das hier nur eine freie Planungsfläche ist. Die Lernenden üben dabei im Stand-Up-Meeting Diskussionen effektiv und konzentriert zu führen, Absprachen zu treffen, Probleme zu identifizieren und sich auf einen gemeinsamen Plan zu einigen. Ihr Project Board ist für sie von Beginn an ein zentraler Arbeits- und Planungsbereich. In der Rolle eines geeignet angelegten Kunden, der **Kundengespräche** mit den einzelnen Gruppen führt, integrierte ich wesentliche Aspekte einer Anforderungsermittlung in die Planung. Aufgabe der Teams ist es, durch gezielte Fragen Kundenwünsche zu präzisieren, dem Kunden ihren geplanten Entwurf zu „verkaufen“ und die Ergebnisse in der Modellierung umzusetzen. Bei der anschließenden arbeitsteilig durchgeführten Implementierung wird in das Werkzeug BlueJ eingeführt, die Klasse ZEICHENFENSTER vorgestellt und zum ersten Mal Quelltext zusammengeführt. Im weiteren Verlauf werden Grundlagen der objektorientierten Programmierung und der algorithmischen Grundstrukturen in mehreren Kleinstprojekten vermittelt. Die Arbeitsaufträge werden in Form von **User Stories** (also Funktionalitäten aus Kundensicht) gestellt. Zu Beginn sind diese sehr präzise formuliert und durch Teilaufträge in Form von **Tasks** (also zu erledigende Aufgaben aus Entwicklersicht) ergänzt. Gegen Ende werden User Stories auch gezielt offener formuliert („Ich möchte, dass sich ein Ball quer über den Bildschirm bewegt.“, „Ich möchte, dass sich ein Kreis in einem Kreisring auf dem Bildschirm bewegt.“), um Spielraum für eigene Interpretationen zu schaffen und um die Lernenden die Tasks selbständig identifizieren und formulieren zu lassen. So führte ich sie schrittweise an den Perspektivwechsel von Kunden- zu Entwicklersicht heran. Die Bearbeitung der Aufträge erfolgte großteils in Gruppen und die Planung fand weiterhin in Stand-Up-Meetings vor dem Project Board statt. Für die Implementierung bilden die Schülerinnen und Schüler Paare und nehmen abwechselnd die Rolle des *Drivers*[5] und des *Navigators*[6]

[4] erstellt von M. Kölling, B. Quig und Ch. Heidrich

[5] bedient die Tastatur, schreibt den Code und denkt dabei laut

[6] behält das große Ganze im Auge, schlägt Alternativen vor und spricht Fehler an

ein. Aus didaktischer Sicht ist dieses Pair-Programming für mich interessant, weil Lernende dadurch passiv voneinander lernen und üben, ihre Codiertätigkeiten in Worte zu fassen und verschiedene Codierstile zu diskutieren. Themen und Aufgabenstellungen der Kleinstprojekte wurden so gewählt, dass User Stories, Tasks und erstellter Code für die Großprojekte angepasst und dort integriert werden konnten. Die in dieser Phase gebildeten Gruppen hatten stets wechselnde Zusammensetzungen, damit sich keine eingespielten Rollenverteilungen herausbildeten und die Lernenden unterschiedliche Herausforderungen kooperativen Arbeitens zu bewältigen lernten. Die Schülerinnen und Schüler konnten Kundengesprächstermine mit mir vereinbaren, wenn sie Unterstützung benötigten.

3.2 Das Projekt

Um den Lernenden spielerisch eine Idee von agiler Projektarbeit zu vermitteln und ihnen Mut zu machen, wird vor dem Einstieg ins Projekt ein „Warm-Up-Spiel“[7] durchgeführt. Hierbei erfahren die Schülerinnen und Schüler die Bedeutung von gemeinsamen Absprachen, Vorteile kurzer, iterativer Entwicklungsphasen (flexible Reaktion auf Änderungen, stete Verbesserung der eigenen Performanz, Sinn einer Reflexion) und die motivierende Wirkung von Zielsetzungen, die in kurzer Zeit erreichbar sind. Stimmen aus dem Kreis der Lernenden nach der Durchführung waren unter anderem: „Noch nie habe ich mich für etwas so reingehängt.“ „Hätte nicht gedacht, dass wir uns so steigern können.“ „Schade, dass wir nicht noch weiter gemacht haben, da wäre noch mehr drin gewesen.“ Für die sich bis zum Schuljahresende erstreckende Projektarbeit dürfen sich die Lernenden eigene Themen wählen. Die Gruppen bilden sich dann entsprechend der fachlichen Interessen.

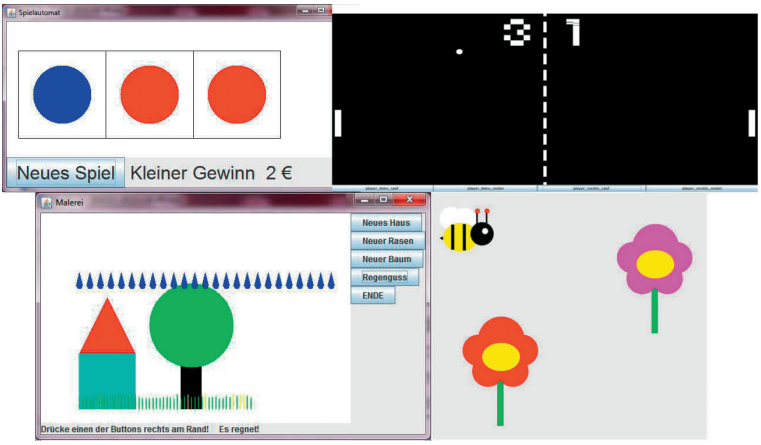


Abb. 1: Anreize zur Themenwahl: Spielautomat, Arcade-Game, bewegte Grafiken

Durch Präsentation einiger, in der zur Verfügung stehenden Zeit umsetzbarer, Beispiele (vgl. Abbildung 1) wird ihnen eine Entscheidungshilfe geboten. Nach der folgenden geheimen Abstimmung der Lernenden bezüglich eines Wunschthemengebiets ließen sich die

[7] http://borisgloger.com/scrum/materialien/tools/

Klassen jeweils problemlos in drei etwa gleich große Interessensgruppen teilen. Die themengebundene Zusammensetzung der Projektgruppen führt dazu, dass in vielen Fällen nicht nur „eingespielte Teams“ zusammenarbeiten und sich die Schülerinnen und Schüler mit für sie „neuen“ Gesprächs- und Teampartnern zurechtfinden müssen.
Die Teams treffen sich zu einer ersten Besprechung in einem Stand-Up-Meeting an ihrem Project Board, das ihnen bis Schuljahresende exklusiv zur Verfügung steht. Die Lehrkraft stößt zu jedem Team, verfolgt die Gespräche einige Zeit, schlüpft dann in die Rolle des Kunden, um die Wünsche zu kanalisieren und verdeutlicht als Berater erste erreichbare Etappenziele. Dieses Vorgehen wiederholt sich im weiteren Projektverlauf, wobei die Kontrolle über das Projekt sobald wie möglich vollständig der Gruppe überlassen wird. Anschließend greift die Lehrkraft nur noch dann ein, wenn die Schülerinnen und Schüler sich deutlich zu hohe Anforderungen stellen oder völlig falsche Wege bei der Implementierung beschreiten wollen. Die Produktfunktionalitäten, die Ziel der jeweiligen Etappe sind, werden als User Stories formuliert und in Reihenfolge ihrer Priorität mit den Namen der Bearbeitenden versehen und an das Project Board (siehe Abbildung 2) geheftet.

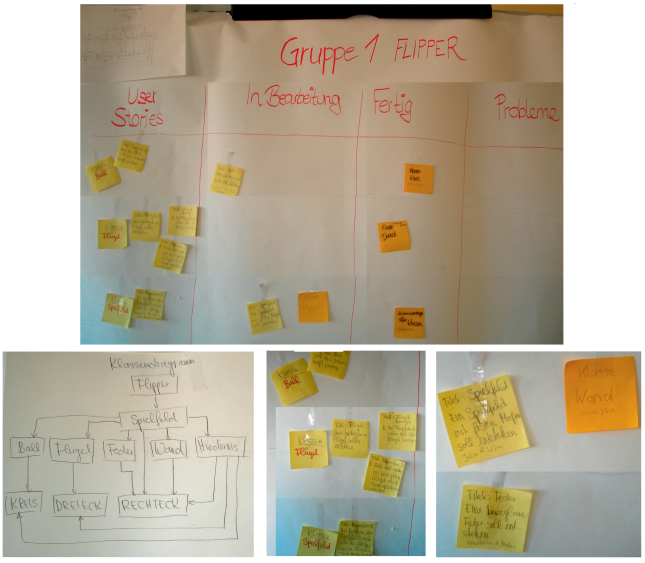


Abb. 2: Project Board „Flipper“ – Klassendiagramm – User Stories/Tasks geplant/in Bearbeitung

Dabei ergibt sich die Priorität entweder logisch aus den Funktionalitäten oder die Teammitglieder einigen sich darüber, was ihnen wichtiger und was weniger wichtig ist. Das Project Board ist ab jetzt ein Board im engeren Sinn der Softwareentwicklung, mit drei Spalten, für User Stories und dazu geplanten Tasks links, für Tasks in Bearbeitung in der Mitte und für fertige Tasks und abgeschlossene User Stories rechts. Sie bieten allen Beteiligten eine ständig aktuelle Übersicht über den Stand des Projektes. Dies gilt sowohl für die Lernenden, die sich bei auftretenden Fragen bezüglich des Verlaufs oder nächster Schritte gerne auch spontan am Board informieren, als auch für die Lehrkraft, die jederzeit einen Überblick hat, wo sich die einzelnen Gruppen aktuell im Projekt befinden und sogar, was die einzelnen Programming-Pairs gerade implementieren. Die Teams halten diese

Informationen immer aktuell. In der rechten Spalte unserer Project Boards gab es eine separate Zelle für **Probleme**. Hier heften die Schülerinnen und Schüler Zettel mit offenen Fragen und Problemen an, die sie nach der Iteration selbstständig oder mit dem Lehrer zusammen klären möchten. Bis zum Ende der Etappe ist der Ablauf nun so, dass die Teams in einem Stand-Up-Meeting in etwa die nächsten 20 Minuten planen. Die Teammitglieder wählen dazu eine oder mehrere User Stories unter Beachtung der Priorität aus, mit deren Umsetzung das Team ca. 20 Minuten beschäftigt ist und formulieren dazu Tasks. Pro Doppelstunde finden so in der Regel zwei Stand-Up-Meetings statt, auf jeden Fall aber eines zu Stundenbeginn. Die geplanten Tasks werden arbeitsteilig und im Pair-Programming implementiert. Das Project Board halten die Schülerinnen und Schüler dabei aktuell. Einen Task, den ein Pair in Bearbeitung nimmt, versieht es vor dem Umhängen mit Namen, damit die Gruppenmitglieder sich bei Fragen gezielt an das Pair wenden können. Auch die Lehrkraft kann, nach Studieren der in Bearbeitung befindlichen Tasks, die entsprechenden Schülerinnen und Schüler direkt ansprechen und sich den Arbeitsstand zeigen lassen. Treten während der arbeitsteiligen Phase dringend durch das ganze Team zu klärende Fragen oder Probleme auf, rufen die Schülerinnen und Schüler ein spontanes Stand-Up-Meeting aus. Gegen Ende der Doppelstunde wird der Code eines Teams zusammengeführt, getestet und falls nötig korrigiert. Ziel ist es, nach jeder Doppelstunde eine getestete und lauffähige Programmversion zu haben. Die nächste Doppelstunde beginnt dann wieder mit einem Stand-Up-Meeting, in dem die Teams erledigte Aufgaben der letzten Stunde rekapitulieren, eventuelle Probleme ansprechen und die nächste Arbeitsphase planen.
In diesen Ablauf wurde der Lerninhalt der 10. Jahrgangsstufe integriert. In den bisherigen Durchläufen zeigte sich, dass die einzelnen Teams trotz unterschiedlicher Themen und individuellem Arbeitstempo oft nahezu zeitgleich an bestimmte Problemstellungen gelangen, z. B. „wir bräuchten etwas, um viele gleichartige Dinge auf einmal anzusprechen“. In diesen Fällen bietet sich eine zentrale Theorie-Input-Phase durch die Lehrkraft an – im genannten Fall zum Thema „(eindimensionale) Felder“. Kommt eine Gruppe deutlich früher zu einer Problemstellung, wird ihr individuell weiter geholfen, um möglichst raschen Projektfortschritt zu gewährleisten. Themen, auf welche die Lernenden nicht von selbst stoßen, können in Kundengesprächen gezielt angeschnitten werden. Beispielsweise wird ein Kundenwunsch nach einer GUI, also einfachen Buttons und Labels mit Begeisterung aufgenommen und mit Eifer angegangen. Hierzu vermittelt die Lehrkraft die Theorie und stellt auch entsprechende Codefragmente zur Verfügung.

4 Beobachtungen und Erfahrungen

Wie in der Einleitung erwähnt, veränderte sich meine Rolle hin zum Coach und Beobachter, da die Schülerinnen und Schüler von Anfang an ihre Projektarbeit selbstständig organisierten. Zu Beginn war dies nur mit Abstrichen „zielorientiert und gründlich“, aber sie behalfen sich, wenn nötig, mit selbst einberufenen spontanen Stand-Up-Meetings und lernten rasch, worauf es bei einer Planung ankommt. Ich griff in der Anfangsphase häufiger als Kunde und/oder Berater unterstützend ein. Später konnte ich als Beobachter durch die starke Betonung der interaktiven Elemente bei den agilen Methoden beispielsweise auch soziale Kompetenzen und deren Entwicklung sehen. Defizite im Bereich der Kommunikationsfähigkeit einzelner Lernender hätte ich früher mit großer Wahrscheinlichkeit nicht

bemerkt, da ich zwar ständig bei einzelnen Teams war, jedoch jeweils nur für kurze Zeit und nur um Fragen zu beantworten. Jetzt konnte ich solche Defizite bemerken und die Entwicklung der Lernenden verfolgen. Dadurch wird die Bewertung und Einschätzung der Einzelleistungen auf einer wesentlich solideren Basis möglich und eine Rückmeldung an die Lernenden über die rein fachlichen Kriterien hinaus möglich.

Einige Tätigkeiten und Prinzipien der Softwareentwicklung, die sonst nur mit Mühe vermittelt werden können, werden von den Schülerinnen und Schülern bei der agilen, iterativen Vorgehensweise als sinnvoll und hilfreich erkannt – denn sie bedürfen keiner zusätzlichen Motivation durch die Lehrkraft. Klassendiagramme beispielsweise erstellen die Teams relativ früh freiwillig und erweitern sie, weil sie ihnen bei der Planung späterer Iterationen, bei der Implementierung sowie beim Testen helfen. Wie wichtig exakte Absprachen im Schnittstellenbereich wie z. B. Namenskonventionen für *Getter* und *Setter* sind, erkennen sie durch die arbeitsteilige Vorgehensweise im Team rasch. Sonst treten beim Testen nach der Zusammenführung Fehler auf oder es muss nachgefragt werden. „Wie heißt dein Getter“ ist zwar schnell durch den Raum gerufen, aber effektiver und ungestörter arbeiten lässt es sich mit verbindlichen Absprachen. Mit zunehmender Programmkomplexität müssen die Lernenden immer häufiger „fremden“ Code lesen und verstehen, um ihn erweitern zu können. Je besser der Code kommentiert ist, umso leichter und schneller geht das. Auch Codefragmente, die Schülerinnen und Schüler sich selbst z. B. mit Hilfe des Openbooks [Ul14] hart erarbeitet haben und die sie gerne voller Stolz weiter geben, führen bei guter Kommentierung zu weniger Nachfragen. Tests am Ende jeder Doppelstunde werden gerne durchgeführt, weil man sehen will, was die anderen Teammitglieder implementiert haben und weil man wieder ein fehlerfreies, lauffähiges Produkt haben möchte. Andere agile Praktiken werden angenommen, aber bis zu ihrer endgültigen Beherrschung benötigen die Lernenden einige Zeit. Die Erstellung kompakter User Stories, die nur aus wenigen einzelnen Tasks bestehen, sowie die Formulierung konkreter, innerhalb einer Arbeitsphase zu bewältigender Tasks, gestaltet sich für die Schülerinnen und Schüler anfangs extrem schwierig. Allerdings helfen theoretische Erläuterungen und Praxisbeispiele hier wenig; erfahrungsgemäßwird diese Technik am Besten durch das Sammeln eigener Erfahrung erlernt: Als zeitversetzt nach einigen Wochen das Planning Poker als Möglichkeit zur Aufwandsabschätzung vorgestellt wurde, hatten die Lernenden bereits ein gutes Gespür für die passende Größe von User Stories und Tasks entwickelt. Die Technik der Aufwandsabschätzung hatte für sie keinen Mehrwert und wurde deshalb verworfen. Ähnliches zeigt sich bezüglich des Planens einer Arbeitsphase. Unabhängig davon, dass mit 20 Minuten ein für Neulinge im Bereich des Programmierens überschaubares Zeitfenster gewählt wurde, gelingt vielen Schülerinnen und Schülern zunächst keine saubere Planung. Stattdessen berufen sie bis zu fünf weitere Stand-Up-Meetings ein, um noch nicht bedachte Probleme zu klären. Nach ein, zwei Doppelstunden jedoch gelingt es den Lernenden problemloser die Arbeitsphase sauber zu planen. Wenn einzelne Teammitglieder vor anderen ihre Aufgaben umgesetzt haben, es aber keine weiteren in der verbleibenden Zeit umsetzbaren Tasks gibt, verbessern die Betroffenen z. B. die Inline-Dokumentation oder testen einzelne Klassen oder das gesamte Projekt.

Etwas schwierig ist es, einen regelmäßigen Wechsel der Rollen beim Pair-Programming zu erreichen. In der Regel übernimmt die besser programmierende Person die Rolle des *Dri-*

vers und die schwächere die Rolle des *Navigators*, obwohl die Lernenden den Sinn eines Rollenwechsels einsehen. Um den Wechsel zu motivieren, wurde von mir nach etwa sechs Wochen der **Truck Factor** eingeführt und ausgerufen. In der Softwaretechnik wird vom Truck Factor gesprochen, wenn es darum geht, dass jeder Mitarbeitende bei spontanem Ausfall eines Teammitglieds vollständig über dessen Aufgaben und Arbeitsstand informiert ist und somit jederzeit die Bearbeitung übernehmen kann. Auch bei schulischen Projekten ist dieser Aspekt nicht von der Hand zu weisen. Immer wieder erkranken Lernende und ihre jeweiligen Partner können dann nicht weiterarbeiten, was zu Verzögerungen in der Projektentwicklung führt. Um die Anforderung zu prüfen, ob sich wirklich jeweils beide Partner eines Programmierteams auf dem gleichen Informationsstand befinden, kann man als Lehrkraft zu einem nicht genauer angekündigten Zeitpunkt den Truck Factor „ausrufen“. Dann wird das angesprochene Paar an zwei verschiedene Rechner gesetzt, die jeweiligen Dateien werden kopiert und beide Partner müssen getrennt voneinander weiterarbeiten. Die Lehrkraft kann sich hierbei von jedem einzelnen erläutern lassen, was seine aktuell zu bewältigenden Aufgaben sind. Anfangs standen die Schülerinnen und Schüler dieser Unterrichtsmethode skeptisch gegenüber, entwickelten aber sehr schnell den Ehrgeiz zu zeigen, dass auch nach Trennung eines Paars beide Beteiligten ohne Nachfragen weiterarbeiten können.

5 Kritischer Vergleich mit klassischen Methoden

Abschließend werden nun die Erfahrungen und Beobachtungen der drei Durchgänge mit projektbasiertem Unterricht kontrastiert mit denjenigen der Vorjahre, in denen ohne Großprojekt arbeitsgleich oder mit vorgegebenem bzw. individuellem Thema, aber nur in Zweiergruppen gearbeitet wurde.

5.1 Sicht der Lehrkraft

Arbeitsgleiches Vorgehen besitzt für die Lehrkraft den Vorteil langfristig möglicher Vorplanung. Der ständig gleiche Unterrichtsverlauf kann aber zur Abstumpfung hinsichtlich der Bedürfnisse und Probleme der Lernenden führen. Außerdem können permanente Wiederholungen demotivierende Effekte hervorrufen. Egal ob diese Methode mit Großgruppen oder mit Programming Pairs durchgeführt wird, ist der Betreuungsaufwand für die Lehrkraft nicht allzu hoch – zumindest ist er kalkulierbar. Individuelle Projekte in Zweiergruppen hingegen fordern die Lehrkraft sowohl in sportlicher als auch informatischer Sicht. Es ist nicht voraussehbar, welches Team zu welchem Zeitpunkt welche Problemstelle erreichen wird. Die Lehrkraft betreibt oft „Turnschuhdidaktik“, indem sie rastlos von einer Gruppe zur nächsten eilt und sich in kürzester Zeit in den jeweiligen Programmcode einlesen und Problemlösungen mit den Lernenden erarbeiten muss. Zusätzlich muss sie noch dafür sorgen, dass alle Theorieinhalte vermittelt werden, was oft dazu führt, dass die Schülerinnen und Schüler „auf Vorrat“ lernen müssen, da sie zum Zeitpunkt der Besprechung gewisser Inhalte diese in keiner Weise in ihrem Projekt benötigen und einsetzen können. Mit Hilfe agiler Methoden lassen sich diese Probleme vermeiden. Die individuelle Projektgestaltung führt zu einer interessanten und abwechslungsreichen, aber auf

Grund der geringen Themenzahl nicht ausufernden Arbeit für die Lehrkraft. Rasch sichtbare Ergebnisse und Anerkennung der Leistungen über die Gruppe hinaus stärken das Selbstbewusstsein der Lernenden und ihr Vertrauen in die eigenen Fähigkeiten. Sie entwickeln den Ehrgeiz, selbstständig Lösungen für ihre Aufgaben zu finden und zeigen dabei ein wesentlich höheres Durchhaltevermögen und deutlich mehr Leistungsbereitschaft als ohne agile Methoden. Auch der Betreuungsaufwand hält sich in Grenzen und ist durch die Verwendung der Project Boards gut planbar. Mit den selbst gewählten Projektthemen erreichen die Lernenden sehr schnell komplexe Fragestellungen wie zum Beispiel: „Wie kann ich eigene Farben definieren?“. Erstens ist es für die Lehrkraft zeitlich nicht möglich, all diese Problemstellungen gleich nach deren Auftreten für die Teams zu lösen. Zweitens wird auch bei Softwareentwicklungs-Großprojekten nicht jede Codezeile direkt erstellt, sondern es werden vielmehr bereits bekannte Teillösungen angepasst und integriert. Diese Technik, mit der die Lernenden bereits im Zusammenhang mit der Java-Klasse ZEICHENFENSTER bei der Erstellung kurzer Programme in Kontakt kamen, wird nun weiter vertieft. Je nach den technischen Gegebenheiten erhalten die Lernenden die Erlaubnis die Online-Version des Openbooks *Java ist auch eine Insel* von Christian Ullenboom [Ull4] zu verwenden oder es wird ihnen die entsprechende Offline-Version zur Verfügung gestellt. Sie dürfen das Buch nicht nur als Informationsquelle verwenden, sondern sich auch bei den Code-Beispielen bedienen, die sie für ihre eigenen Projekte entsprechend anpassen müssen. Die Theorieeinheiten sind zwar hinsichtlich ihres Zeitpunktes nicht langfristig zu terminieren, lassen sich aber gut über das Schuljahr verteilen, so dass auch kleine schriftliche Leistungserhebungen durchgeführt werden können. Die Beurteilung der individuellen Leistungen gestaltet sich, unterstützt durch die Eintragungen an den Project Boards und den Beobachtungen bei Kundengesprächen oder Stand-Up-Meetings, einfacher als bei den zu Beginn des Abschnitts beschriebenen Vorgehensweisen.

5.2 Sicht der Schülerinnen und Schüler

Bei der Projektarbeit in Zweiergruppen bei „klassischem Vorgehen“ kritisieren die Lernenden am meisten die langen Wartezeiten auf Betreuung durch die Lehrkraft. Sie empfinden einerseits die individuell wähl- und gestaltbaren Projektideen als sehr motivierend und insgesamt positiv, sind aber andererseits in Bezug auf den aus oben genannten Gründen sehr langsamen Projektfortschritt unzufrieden. Insbesondere ist es für die Schülerinnen und Schüler frustrierend, dass sie häufig nicht nur wochenlang kaum einen Fortschritt bei ihrer Software sehen können, sondern es bis zum Schuljahresende nicht schaffen, ein funktionsfähiges Produkt erstellt zu haben. Außerdem entstand bei einigen Lernenden der Eindruck, dass ihre individuellen Leistungsbewertung im Verhältnis zu denen ihrer Teampartner nicht gerechtfertigt sei. Auch aus Sicht der Lernenden führt der Einsatz der agilen Methoden dazu, die oben genannten negativen Aspekte zu vermeiden. Sie fühlen sich rundum betreut, sehen selbst an den nach jeder Iteration lauffähigen Programmversionen einen permanenten Projektfortschritt und empfinden die Bewertung ihrer individuellen Leistung durchweg als nachvollziehbar und korrekt. Außerdem meldeten viele Lernende zurück, dass sie durch diese Methode auch „etwas fürs Leben“ gelernt haben.

6 Fazit und Ausblick

Der Einsatz eines agilen Frameworks bei der Gestaltung des Informatikunterrichts mittels eines ganzjährigen Projekts stellt sich aus Sicht aller Beteiligten als positiv dar. Sowohl die Lehrkraft als auch die Lernenden arbeiteten hochmotiviert, die Schülerinnen und Schüler hatten viel Spaß– sicher mitbegründet durch häufige Erfolgserlebnisse – und erlernten vieles, was in den letzten Jahren unerreichbar schien oder gar nicht in Erwägung gezogen wurde. So erschlossen sich fast alle Beteiligten, wie man selbst Farben mit Hilfe der Hexadezimalcodierung definieren kann, einige lernten, wie man mehrere geometrische Grundbausteine zu grafischen Gesamtobjekten kombinieren kann, die dann zum Beispiel mit Farbverläufen gefüllt werden können. Es entstanden Stadtgrafikprojekte, bei denen sich die Sonne auf einer bogenförmigen Bahn über den Himmel bewegt und nachts dann die Lichter in den Fenstern der Häuser angehen. Die Spielautomatenprogramme wurden soweit ausgebaut, dass wirklich ein Hebel wie bei den „One-Arm-Bandits“ betätigt werden musste, um die Walzen zu starten, die während ihres Laufs wechselnde Symbole (verschiedene Früchte und nicht lediglich Kreise) zeigten. Bei Gewinn fielen dann auch Münzen aus dem Geldauswurfschacht. Für eine authentische Darstellung der Arcade-Games wurde für die Punkteanzeige eine 4x8-Punkt-Matrix eingesetzt.

Die Gestaltung des agilen Frameworks wird in weiteren Durchgängen weiter ausgebaut. Die Hilfsprogramme und Materialien für die Theorieeinheiten werden gesammelt und in den nächsten Jahren Informatiklehrkräften als „Werkzeugkasten“ zugänglich gemacht.

Literaturverzeichnis

[GR13] Göttel, T.; Romeike, R.: Agiler Projektunterricht in der Schulinformatik. In (Breier, N.; Stechert, P.; Wilke, T., Hrsg.): 15. GI Fachtagung Informatik und Schule, Praxisband. Kiel Computer Science 2013/3. Department of Computer Science, CAU Kiel, S. 151–158, 2013.

[Gu08] Gudjons, H.: Handlungsorientiert lehren und lernen: Schüleraktivierung, Selbsttätigkeit, Projektarbeit. 2008.

[HNR07] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Physica-Verlag, 2007.

[IS03] ISB München: , Lehrplan Informatik 10 (NTG), 2003. http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/index.php?StoryID=26435.

[KB09] Kölling, M.; Barnes, D. J.: Java lernen mit BlueJ: Eine Einführung in die objektorientierte Programmierung. Pearson Studium, 2009.

[RG12] Romeike, R.; Göttel, T.: Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. WiPSCE '12, ACM, New York, NY, USA, S. 48–57, 2012.

[Ul14] Ullenboom, Ch.: , Java ist auch eine Insel: Das umfassende Handbuch (Galileo Computing), 2014. http://openbook.galileocomputing.de/javainsel11.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 229

Automatox: Ein Spiel für den Informatikunterricht

Ulrike Klein[1]

Das Spiel ist der Weg der Kinder zur Erkenntnis der Welt, in der sie leben.
Maxim Gorki

Abstract: Ein Spiel für den Informatikunterricht, das Schülerinnen und Schülern ohne Vorkenntnisse einerseits eine Vorstellung von endlichen Automaten vermittelt, andererseits auch die Prinzipien der Speichersysteme Stack und Schlange verdeutlicht. Außerdem ein Spiel, das Spaß macht und die Allgemeinbildung fördert.

Keywords: Spiel, Informatikunterricht, Automat, Stack, Schlange, Allgemeinbildung

1 Spiele im Informatikunterricht

Kinder lernen durch Spielen. Das ist allgemein akzeptiert. Doch im Schulunterricht führen Spiele ein Nischendasein. Sie werden zur Abwechslung herangezogen oder als Belohnung versprochen oder um die letzte Stunde vor den Ferien auszufüllen, wenn kein „richtiger Unterricht“ sinnvoll erscheint. Dabei können Spiele durchaus gewinnbringend eingesetzt werden. Sie bringen Freude in den Unterricht, motivieren zum Lernen, förden soziale Kompetenzen, sind handlungsorientiert, unterstützen die Selbstbestimmung — kurz: sie leisten das, was man von einem modernen Unterricht erwartet. Hilbert Meyer widmet dem Spielen im Unterricht eine ganze Lektion in seinen Unterrichtsmethoden [Me87]. Er sieht aber auch einen Widerspruch: Während Spiele an sich nach seiner Charakterisierung zweckfrei sind, haben didaktische Spiele durchaus „das Ziel, die sozialen, intellektuellen und ästhetischen Kompetenzen der Schülerinnen und Schüler zu fördern.“ ([Me87, S.344] Dieses Dilemma erkennt auch Gerd Busse: „Didaktische Intentionen und das Wesen des Spiels sind weitgehend nicht deckungsgleich.“ ([Bu03]) Dass für Schülerinnen und Schüler die Begriffe „Spaß“ und „Lernen“ im Widerspruch zueinander stehen und dadurch der Spaßfaktor eines Lernspiels als Ausschlusskriterium für den Lernerfolg vonseiten der Schülerinnen und Schüler empfunden wird, stellt Nadine Hansen in ihrer Dissertation fest. ([Ha10]) Sie führt dort eine empirische Vergleichsstudie über den Lernerfolg von Lernspielen im Vergleich zum fragend-entwickelnden Unterricht an über 300 Gymnasiasten verschiedener Jahrgangsstufen durch. Ihr Ergebnis ist, dass das Lernspiel in Bezug auf Lernerfolg nicht signifikant besser abschneidet als der Frontalunterricht. Andererseits stellt sie fest, dass die „motivationalen Werte auf ein methodisches Gestaltungselement deuten, das von den Schülerinnen und Schülern bestens angenommen wird.“ ([Ha10, S.288])

[1] Bertha-von-Suttner-Schule, An den Nussbäumen 1, 64546 Mörfelden-Walldorf, ulrike_sabine.klein@stud.tu-darmstadt.de

Daraus schließe ich, dass Lernspiele im Unterricht ihren Platz haben als ein Mittel zur abwechslungsreichen Methodik und dass sie durchaus geeignet sind, verschiedene Kompetenzen der Schülerinnen und Schüler anzusprechen und sie zu motivieren. In den folgenden Abschnitten wird ein Gesellschaftsspiel beschrieben, dass ich für den Informatikunterricht entwickelt und auch ausprobiert habe.

Man kann nun aber die Frage stellen: Wenn im Informatikunterricht gespielt wird, warum ist es dann kein Computerspiel? Da kann man mit dem Edsger W. Diijkstra zugeschriebenen Satz antworten: „In der Informatik geht es genauso wenig um Computer wie in der Astronomie um Teleskope." Wie Jens Gallenbacher in seinem Buch „Abenteuer Informatik" ([Ga12]) und in der gleichnamigen Ausstellung vormacht, braucht man keine Computer, um Informatik zu lehren. Er stellt das „Begreifen" der Thematik und die „Aha"-Effekte, hervorgerufen durch Einfachheit und Verständlichkeit, in den Mittelpunkt. Sein Fazit: „Es ist nicht nur möglich, Informatik ohne Computer zu lehren, sondern entsprechende Konzepte sind in verschiedener Hinsicht sogar besser und nachhaltiger geeignet, den Kern der Wissenschaft zu vermitteln." ([Ga09, S.36])

2 Automatox — Ein Spiel für den Informatikunterricht

2.1 Spielidee

(Ausführliche Spielanleitung, Spielmaterialien und Bastelanleitung findet man unter http://www.mutam.de/automatox.)

Der Spielplan simuliert einen endlichen Automaten. Die Kreise stellen die Zustände dar. Dabei sind ein Startzustand und verschiedene Zielzustände — gekennzeichnet durch Tierbilder — ausgezeichnet. Gespielt wird mit einem einzigen Spielstein, der den aktuellen Zustand des Automaten anzeigt. Anfangs befindet sich der Automat im Startzustand. Die Pfeile mit den Buchstaben geben an, welche Zustandsübergänge möglich sind. Durch Eingabe eines Buchstabens wird der Automat in den entsprechenden nächsten Zustand überführt. Ziel des Spieles ist es, den Automaten in die verschiedene Zielzustände zu bringen.
Die Reihenfolge, in der die Ziele erreicht werden müssen, wird durch Aufdecken von Zielkarten ermittelt. Es sind immer drei Zielkarten aufgedeckt, von denen das vorderste Ziel zuerst erreicht werden muss. Wird ein Ziel erreicht, so erhält der Spieler, der das geschafft hat, diese Karte. Dann rücken die anderen Zielkarten einen Platz vor und die nächste wird aufgedeckt und hinten angelegt, was dem Prinzip einer Schlange entspricht.
Um den Zustand des Automaten zu ändern, gibt es Buchstabenkarten. Jeder Spieler hat drei Karten auf der Hand und bis zu zwei Ablage-Stacks, die er verwenden kann. Dazu legt er eine Karte in das entstehende Wort und zieht den Spielstein weiter. Wenn er nicht mehr legen kann oder möchte, legt er eine Karte auf einem seiner Stacks ab. Bei der Verwendung der Stack-Karten ist zu beachten, dass immer zuerst die oberste verwendet werden muss. Schafft es ein Spieler, alle Handkarten in das Spiel zu legen, so darf er sofort neue Handkarten ziehen und weiterspielen.

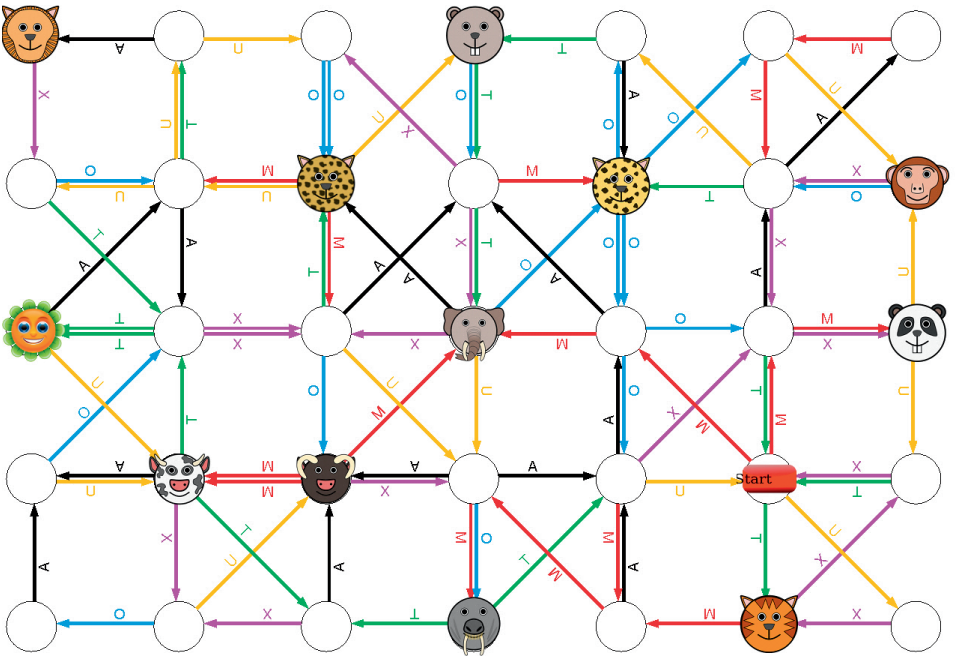


Abb. 1: Ein mögliches Spielfeld

Das Spiel endet, wenn das letzte Ziel erreicht wurde. Sieger ist, wer die meisten Zielkarten gesammelt hat.

2.2 Informatisches Konzept und Lehrplanbezug

Als Automaten bezeichnet man technische Geräte, die zu gewissen Eingaben nach einem schematischen (algorithmischen) Verfahren nach endlich vielen Schritten erwartete Ausgaben liefern. ([SS11, S.261])
Im Alltag versteht man unter einem Automaten eine Maschine, die bestimmte Abläufe selbsttätig (also: „automatisch“) ausführen kann. Beispiele sind Fahrkarten-, Kaugummi-, Spielautomaten, Ampelschaltungen, Aufzüge.
In der Informatik ist ein Automat eine abstrakte Maschine, die bestimmte Zustände annehmen kann. Durch Eingabezeichen wird der Automat abhängig vom aktuellen Zustand und der Eingabe in einen Folgezustand überführt. Es werden viele Arten von Automaten unterschieden wie Moore- und Mealey-Maschinen, Turing-Maschine, Kellerautomaten, neuronale Netze und andere mehr. Automaten spielen in der Informatik eine große Rolle, weil sie eine einfache Struktur bieten, um komplexe Zusammenhänge übersichtlich darzustellen.
Daher ist es sinnvoll, dass Automaten auch im Informatikunterricht in der Schule vorkommen. Der Lehrplan Informatik für die Oberstufe in Hessen [Le10] sieht als Thema für die Q3 „Konzepte und Anwendungen der Theoretischen Informatik“ vor. Dort werden explizit

Automaten als Unterrichtsgegenstand aufgeführt, wobei alle Kurse Endliche Automaten behandeln und die Leistungskurse auch Kellerautomaten und Turing- oder Registermaschinen. Schubert-Schwill [SS11] widmet einen ganzen Abschnitt der Automatentheorie.

Zwei andere informatische Konzepte kommen auch in dem Spiel vor: Die Speicherverwaltungen First In – First Out (FIFO) und Last In - First Out (LIFO). FIFO bedeutet, dass die Elemente, die zuerst gespeichert werden, auch zuerst entnommen werden. Man nennt das auch „Schlange“ und es handelt sich um das Prinzip, mit dem man z. B. an einer Kasse ansteht. Dagegen bedeutet LIFO, dass das Element, das als letztes in den Speicher gegeben wurde, als erstes wieder entnommen wird. Das nennt man „Stack“ oder auch „Keller“ und wird beispielsweise bei einem Bücherstapel angewandt. Diese Begriffe finden nicht nur innerhalb der Informatik Verwendung, sondern auch in der Warenwirtschaft oder der Produktionstechnik. Im Lehrplan kommen diese speziellen Bezeichnungen auch vor: Bei der Beschreibung der Q1 „Objektorientierte Modellierung“ [Le10, S.17] wird auf die Konstruktion von Klassen hingewiesen mit „Die abstrakten Datentypen Keller und Schlange können beispielsweise als spezielle lineare Listen aufgefasst und daher von einer vorgegebenen Listenklasse abgeleitet werden.“ Ferner heißt es beim Wahlthema „Betriebssysteme“ in der Q4: „Konzepte und Strategien zur Prozess-, Speicher- und Dateiverwaltung können unabhängig von einer speziellen Realisierung thematisiert werden.“ [Le10, S.26] und es wird die „prioritätsgesteuerte Warteschlange“ erwähnt. Michael Fothe hat diese Prinzipien in seinem Buch „Kunterbunte Schulinformatik“ im Zusammenhang mit dem Informatiksystem Taschenrechner direkt thematisiert und mit einer anschaulichen Grafik versehen. [Fo10, S.48]

2.3 Fachdidaktisches Konzept

Mit diesem Spiel werden informatische Prinzipien vorgestellt und zwar sehr dezent. Es sind keinerlei informatische Vorkenntnisse nötig, um das Spiel zu spielen. Man kann das Spiel auch einfach aus Spaß spielen, ohne zu merken, dass man es mit informatischen Prinzipien zu tun hat. Man kann das Spiel spielen und sich anhand des informatischen Anhangs zur Spielanleitung über die enthaltenen informatischen Zusammenhänge informieren. Beabsichtigt ist aber folgende Vorgehensweise: Man spielt das Spiel und nimmt es zum Anlass, die darin enthaltenen informatischen Prinzipien zu thematisieren und Bezüge zum Alltag zu suchen. Beispiele findet man genug, wenn man seine „Informatik-Brille“ (als Analogon zur „Mathe-Brille“ von Heinrich Winter [Wi95]) aufsetzt und sich aufmerksam in der Lebenswelt umschaut. Die im Spiel vorgeführten Prinzipien Automat, Stack und Schlange sind ja weniger kompliziertes informatisches Fachwissen als vielmehr ein Teil der Allgemeinbildung, die jeder haben sollte. Differenziert man nach Heymann ([Wi03]) genauer zwischen den Aufgaben der Allgemeinbildung, so würde man dieses Spiel der „Weltorientierung“ zuordnen: Die Jugend soll mit Wissen über die Welt ausgestattet werden. Die Fakten sollen fächerübergreifend in das Weltbild eingepasst werden. Auch zum Bereich „Stiftung kultureller Kohärenz“ lassen sich Bezüge finden, da hier zentrale Ideen angesprochen werden. Es kann auch zur „Stärkung des Schüler-Ichs“ beitragen, wenn sie Alltag die Prinzipien erkennen und dann ihren Eltern erklären können, worum es sich handelt.

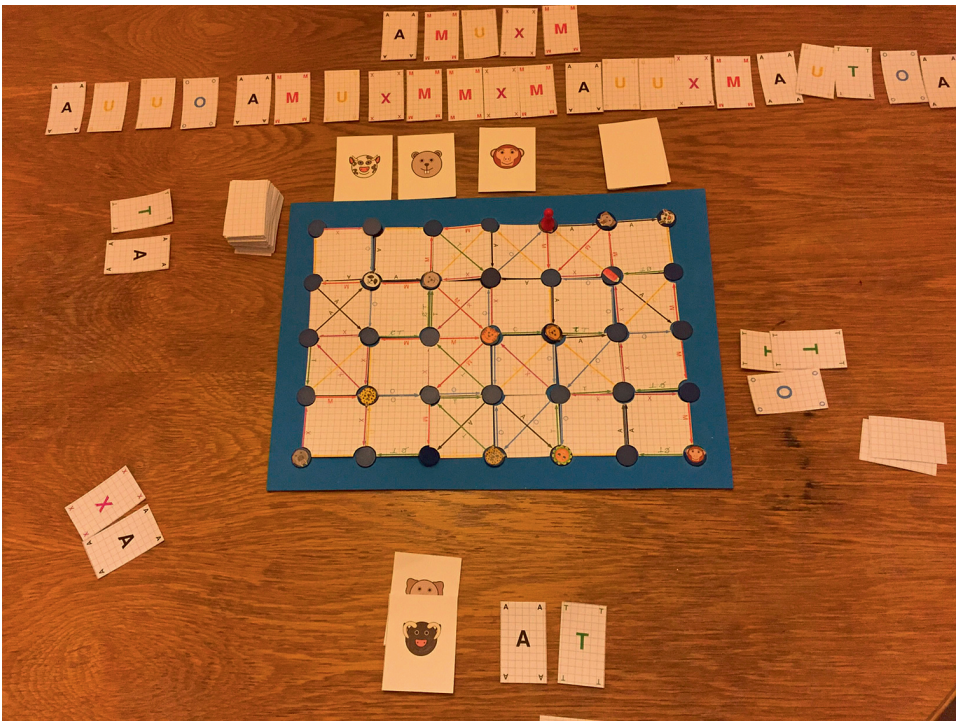


Abb. 2: Das aufgebaute Spiel im Spielverlauf

2.4 Spielkonzept

Das Spiel „Automatox“ ist ein Gesellschaftsspiel, das für Kinder ab etwa 10 Jahren geeignet ist. Die Anzahl der Spieler ist nicht festgelegt. Ich empfehle 2–6 Spieler, es kann aber durchaus auch einer mehr sein. Das verlängert nur die Wartezeiten.
Die Spieldauer beträgt ca. 30 Minuten.
Das Spiel enthält sowohl strategische als auch zufällige Elemente, wobei der Zufall überwiegt. Das Spiel ist so konzipiert, dass der Spaßcharakter im Vordergrund steht und sich die didaktischen Absichten nicht aufdrängen.
Auch wenn sich die Spielanleitung erst kompliziert anhört, so ist das Spiel doch schnell zu lernen und hat nur wenige Regeln. Die Regeln des Kartenziehens und des Ablegens auf Stacks sind an das Spiel „Skip-Bo“ angelehnt, das relativ bekannt ist.
Damit das Spielfeld nicht immer gleich aussieht, wird es aus einzelnen quadratischen Elementen zusammengebaut. Sie enthalten die Kanten des Graphen. Die Knoten des Graphen werden durch runde Spielchips dargestellt, von denen eines mit dem Startzeichen beklebt ist und zwölf andere mit Zielbildern.
Es wurden verschiedene Varianten des Spieles ausprobiert. Bei einem zu kleinen Spielfeld mit zu wenigen verschiedenen Buchstaben sind die Ziele zu leicht zu erreichen und mit ein wenig Glück kann man ganz viel auf einmal „abräumen“. Bei einem größeren Spielfeld mit zu vielen verschiedenen Buchstaben stockt das Spiel, weil man zu selten etwas

Passendes zieht. Als brauchbare Variante hat sich ein Spiel mit 24 Spielfeldkarten (also 35 Knoten) und sechs verschiedenen Buchstaben herausgestellt. Statt die ersten sechs Buchstaben A, B, C, D, E und F des Alphabets zu verwenden, heißen sie jetzt A, U, T, O, M und X. Damit erhöht sich die Chance, dass das erhaltene Wort, das von einem Ziel zum nächsten gebildet wird, auch aussprechbar ist.
Damit jede Kombination des Aneinanderlegens der Spielfeldkarten einen zum Spielen möglichen Graphen ergibt, ohne dass es Sackgassen oder unerreichbare Knoten gibt, wurde die Bedingung gestellt, dass zu jeder Ecke mindestens ein Pfeil hingehen und mindestens einer weggehen muss. Wenn man eine minimale Anzahl von Pfeilen pro Spielfeldkarte vorsieht, wurde dadurch die Anzahl der Grundmuster abgesehen von Farbe und Orientierung der Pfeile auf zwei verschiedene eingeschränkt.

3 Erprobung im Unterricht

3.1 These

Untersucht werden sollte die These: „Mit Hilfe des Spieles „Automatox“ ist es möglich, Schülerinnen und Schülern, die noch keine fachspezischen Vorkenntnisse besitzen, eine Vorstellung von den informatischen Prinzipien eines Automaten und der Speicherverwaltungen FIFO und LIFO zu vermitteln.“

3.2 Vorgehensweise der Erprobung und Fragebögen

Um diese These zu überprüfen, wurde mit Schülergruppen ein Testspiel durchgeführt. Zu Beginn erhielten die Schülerinnen und Schüler einen Fragebogen, um ihre Vorkenntnisse und die alltäglichen und intuitiven Vorstellungen herauszufinden. Dann spielten sie in Kleingruppen das Spiel, von dem ich drei Exemplare erstellt hatte. Anschließend erhielten sie einen zweiten Fragebogen, in dem abgefragt wurde, wie sich die Vorstellungen durch das Spiel verändert haben. Die beiden Fragebögen wurden von den Schülern mit einem Zeichen markiert, um sie einander zuordnen zu können.

3.2.1 Fragen des ersten Fragebogens

- In welcher Klassenstufe befindest du dich?
- Hast du Informatikvorkenntnisse (abgesehen vom letzten halben Jahr Unterricht)?
- Was verstehst du unter einem Automaten?
- Hast du schon etwas von „First In – First Out“ und von „Last In – First Out“ gehört?
  ◯ Ja ◯ Nein
- Was könnte man darunter verstehen?
- Unterstreiche die Wörter, die du in dem folgenden Graphen findest.
  ABC — CA — DAA — ADAAD – EBC — AAB

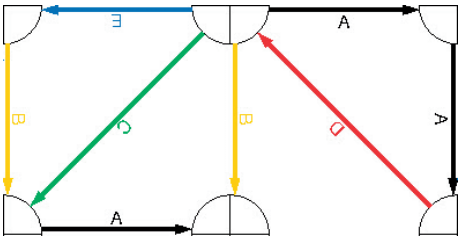


Abb. 3: Der in den Fragebögen verwendete Graph

3.2.2 Fragen des zweiten Fragebogens

- Das Spiel simuliert einen Automaten. Versuche zu formulieren, was man in der Informatik unter einem Automaten versteht. (Hinweis: Spielanleitung)
- Die Prinzipien „First In – First Out“ und von „Last In – First Out“ kommen in dem Spiel für die verschiedenen Kartenstapel (Zielkarten und Ablagestapel) vor.
  Beschreibe, wo diese Prinzipien vorkommen und was sie bedeuten.
- Suche Beispiele, wo diese Prinzipien im Alltag vorkommen.
- Schreibe fünf Wörter auf, die du in dem folgenden Graphen findest.

3.3 Durchführung der Erprobung

Die Erprobung wurde mit zwei Lerngruppen an einer integrierten Gesamtschule mit Oberstufe durchgeführt.
Die erste Lerngruppe ist ein Informatikkurs in der Einführungsphase, den ich seit einem halben Jahr unterrichte. An der Erprobung haben elf SchülerInnen (ein Mädchen und zehn Jungen) teilgenommen. Sie spielten das Spiel aufgeteilt in drei Gruppen. Die im Spiel vorkommenden Prinzipien wurden im Unterricht noch nicht behandelt.
Bei der zweiten Lerngruppe handelt es sich um Schülerinnen und Schüler einer 9. Klasse, die ich während der Projektwoche betreue. Die Teilnahme an der Erprobung erfolgte dort auf freiwilliger Basis. Nachdem recht viele den ersten Fragebogen ausgefüllt hatten, zogen es einige vor, beim Spiel nur zuzuschauen. Gespielt haben dann zwei Gruppen, eine mit vier Mädchen und eine mit drei Jungen, so dass ich hier sieben Antworten habe.
In beiden Fällen war genug Zeit vorhanden, so dass die Gruppen das Spiel bis zum Ende spielen konnten.

3.4 Ergebnisse

Bei der Frage nach der Klassenstufe stellte sich heraus, dass alle SchülerInnen der E-Phase in der 11. Klasse sind, also alle von der IGS kommen und nicht von einem Gymnasium mit G8.
Vorkenntnisse gibt es wenige: In der E-Phase gaben zwei Programmierkenntnisse an, einer

einen Kurs in Klasse 10 und zwei „Ja“ ohne nähere Angabe. In Klasse 9 wurde einmal ein Kurs besucht und einmal „Ja“ ohne nähere Angabe gesagt.
Bei der Frage nach Automaten wurden im ersten Fragebogen unabhängig von der Jahrgangsstufe verschiedene Automaten aufgezählt. Einige versuchten sich an Definitionen: „ein automatisch ablaufender Prozess“ (11), „ein Gerät, das nicht von einem Menschen bedient wird, sondern von einem Computer“ (11), „selbst arbeitendes Programm für bestimmte Aktionen und Bereiche“ (11), „Eletronische Objekte, die verschiedene Aufgaben erfüllen. Sie werden von Elektronik angetrieben.“ (9), „Ein beliebig einsetzbares Gerät, das auf maschineller Basis mittels eines Elektronengehirns arbeitet.“ (9), „etwas, das automatisch Dinge tut, wenn man zum Beispiel einen Knopf drückt und etwas passiert“ (9), „Ein Gerät, das ohne Hilfe von Menschen funktioniert, nur zum Starten oder Abschalten“ (9).
Inwieweit hat sich das Bild des Automat im zweiten Fragebogen verändert? In der Spielanleitung stand eine Definition, aber die hat niemand übernommen. In der 9. Klasse kam zum Thema Automat nichts Neues dazu. In der 11. Klasse gab es Bezüge zum Spiel und Antworten wie „Man hat vorgegebene Pfade“ (zweimal), „Ein Gerät, das über Befehle mit Schaltkreisen ein Ziel erreicht.“, „Mehrere zielführende Wege.“, „Es gibt verschiedene Zustände in einem Automat.“ (zweimal), „Um ein Ziel zu erreichen, müssen verschiedene Zustände nacheinander geschaltet werden.“ „Dass man mit verschiedenen Wegen an sein Ziel ankommt, obwohl man manchmal wieder zurück gehen muss.“
Mit der Frage nach FIFO oder LIFO konnte im ersten Fragebogen unabhängig vom Jahrgang niemand etwas anfangen. Es gab ein paar Übersetzungen, die aber keinen Bezug zu einem Inhalt hatten. Im zweiten Fragebogen wurde durch die Fragestellung schon deutlich der Bezug zum Spiel hergestellt und das Beschreiben glückte in vielen Fällen. Bei den 11-Klässlern gaben ca. die Hälfte die richtige Antwort, die anderen ließen das Feld immer noch frei. Als Beispiel für LIFO kam mehrmals der Essensautomat. Es kommt zuerst das, was vorne ist. Bei dieser Frage zeigte sich die 9. Klasse kreativer. Der Anteil, der die Frage über Karten richtig beantwortete war höher und es kamen viele Beispiele (oft nicht nach LIFO und FIFO sortiert): Abheften von Blätten im Schnellhefter, an der Kasse, Toasts im Toaster (?), Schlange im Supermarkt, Snack-Automat. Mit der 11. Klasse diskutierte ich im Anschluss an den Fragebogen noch über diese Prinzipien. Und sobald sie ein bisschen in diese Richtung gelenkt wurden, fanden auch sie eine Reihe von Beispielen, die jetzt aber nicht auf den Bögen festgehalten sind.
Hingegen war das Finden von Wörtern im Graphen für die meisten intuitiv klar und schon im ersten Fragebogen kein Problem. Die wenigen, die im ersten Fragebogen noch nichts mit der Frage anfangen konnten, waren bei der zweiten Befragung in der Lage, Wörter zu finden.

Zusätzlich zu den Fragebögen fragte ich noch nach Verbesserungsvorschlägen zum Spiel. Die meisten hatten — ihrer eigenen Aussage und meiner Beobachtung nach — Spaß am Spiel. Dabei muss man bemerken, dass bei den Neuntklässlern nur die gespielt haben, denen solche Gesellschaftsspiele Spaß an machen. Der Zufall ist bei diesem Spiel ein erheblicher Faktor. Dadurch war die Mädchengruppe bei den Neuntklässlern sehr viel schneller fertig als die Jungengruppe. Bei den Mädchen kam der Wunsch nach einem größeren Spielfeld, bei den Jungen die Bitte um mehrere verschiedene Buchstaben an einem Pfeil.

Interessant ist auch, dass ganz ohne Nachfrage über Spielstrategien reflektiert wurde: „Oft ist es sinnvoll, in die falsche Richtung zu gehen, wenn man dadurch seine Karten los wird und wieder neu ziehen darf."

4 Fazit

Das Spiel „Automatox" ist ein Spiel, das Spaß macht und informatische Prinzipien beinhaltet. Das Spielen allein bewirkt jedoch nicht, dass man hinterher die Definition für einen Automaten angeben kann. Das liegt vielleicht auch daran, dass in der Alltagssprache das Wort „Automat" immer mit einem technischen Gerät verknüpft ist, so dass die Verbindung eines abstrakten Graphen zu einem Automaten nicht gesehen wird. Das Spiel liefert aber Beispiele und hilft Vorstellungen, die vielleicht noch nicht wirklich ausformuliert sind, zu entwickeln. Wenn man dann eine Frage stellt in der Art "Wo kommt so etwas denn im normalen Leben vor?", dann werden auch Analogien gefunden. Um ein informatischen Gewinn zu ziehen, sollte man nach dem Spiel noch anregen, darüber nachzudenken, welche Prinzipien vorkamen und Beispiele zu suchen. Aus diesem Grund ist das Spiel auf jeden Fall zur Einführung in eines der genannten Themen geeignet. Ich halte es aber auch für sinnvoll, das Spiel zu spielen, wenn sich nicht direkt ein Unterricht darüber anschließt, sondern einfach aus Gründen der Motivation und der Allgemeinbildung.

Literaturverzeichnis

[Bu03] Busse, G.: Spielen im Unterricht — Ein Dilemma. Paulo Freire Verlag 3/2003, 2003. http://www.freire.de/node/69, Stand: 27.1.2015.

[Fo10] Fothe, M.: Kunterbunte Schulinformatik. LOG IN Verlag, Berlin, 2010.

[Ga09] Gallenbacher, J.: Abenteuer Informatik — „Informatik begreifen" wörtlich gemacht. Jgg. 156, Gesellschaft für Informatik, Bonn, S. 28–37, 2009. http://subs.emis.de/LNI/Proceedings/Proceedings156/28.pdf, Stand: 28.1.2015.

[Ga12] Gallenbacher, J.: Abenteuer Informatik. Springer Spektrum, Berlin Heidelberg, 2012.

[Ha10] Hansen, N.: Spielend lernen? Lernspiele in divergierendem Fächerkontext der Sekundarstufe I und II und ihre Auswirkungen auf Lernerfolg bei Kindern und Jugendlichen. Dissertation Universität Duisburg-Essen, DuEPublico, Duisburg-Essen, 2010. http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-25012/Diss_Hansen.pdf, Stand: 27.1.2015.

[Le10] Lehrplan Informatik Hessen, Gymnasialer Bildungsgang, Gymnasiale Oberstufe, https://verwaltung.hessen.de/irj/servlet/prt/portal/prtroot/slimp.CMReader/HKM_15-/HKM_Internet/med/f9a/f9a5418f-7d7a-921f-012f-31e2389e4818,22222222-2222-2222-2222-222222222222,true, Stand: 28.1.2015.

[Me87] Meyer, H.: Unterrichtsmethoden II: Praxisband. Cornelsen, Berlin, 1987.

[SS11] Schubert, S.; Schwill, A.: Didaktik der Informatik, 2. Auflage. Spektrum Verlag, Heidelberg, 2011.

[Wi95] Winter, H.: Mathematikunterricht und Allgemeinbildung. Jgg. 61. GM Gesellschaft für Didaktik der Mathematik, S. 37–46, 1995.

[Wi03] Witten, H.: Allgemeinbildender Informatikunterricht? Ein neuer Blick auf H. W. Heymanns Aufgaben allgemeinbildender Schulen. S. 53–69, 2003. http://subs.emis.de/LNI/Proceedings/Proceedings32/GI-Proceedings.32-7.pdf, Stand: 28.1.2015.

Vom Gatter zum Compiler: Im Unterricht durch sieben Abtraktionsebenen

Urs Lautebach[1]

Abstract: Vorgestellt wird ein Unterrichtsgang, der die Funktionsweise von Computern über viele Abstraktionsebenen hinweg erschließt und erlebbar macht. Die Schüler erstellen dabei auf jeder Ebene Soft- oder Hardware selber. Ausgehend von logischen Verknüpfungen bauen die Schüler (am Simulator oder mit Steckkästen) zunächst Addierwerke und andere ausgewählte Teile, soweit sie auch im Prozessorsimulator "Mikrosim" sichtbar sind. Dort werden dann Datenflusssteuerung, Neumann-Zyklus und Mikro- sowie Assemblerprogrammierung behandelt. Weil aus Schülersicht erst die Hochsprache wieder vertrautes Terrain darstellt, entsteht schließlich ein Compiler zumindest für die Sprache der arithmetischen Ausdrücke: Die Schüler simulieren dessen Syntaxanalyse zunächst "unplugged" mit einem Kartenspiel. Daran angelehnt programmieren sie den Compiler, der beliebige Ausdrücke in Assemblerprogramme übersetzt: Nach dem Assemblieren sind diese Programme im Simulator lauffähig. Der Prozessor berechnet dann vor den Augen der Schüler den Wert des Ausdrucks. Der Reiz des Unterrichtsganges ist der transparente und für die Schüler erlebbare Durchstich sozusagen vom Gatter zum Compiler.

Keywords: Logische Schaltungen, Rechenwerk, Mikroprozessor, Mikrosim, Mikrocode, Assembler, Compiler, Unterrichtseinheit, Abstraktionsebenen

1 Einleitung

Die Frage "Ja, aber wie FUNKTIONIERT denn nun ein Computer?" wird in der Schule meist nur punktuell behandelt: Der Unterricht schafft sozusagen einzelne "Inseln des Verstehens", etwa logische Schaltungen, Mikroprogrammierung oder Hochsprachen.

Der hier vorgestellte Weg hingegen baut auch zwischen diesen Inseln so viele Brücken wie möglich: Das Ziel ist ein Durchstich durch viele Abstraktionsebenen, eben "vom Gatter zum Compiler". Dieser Durchstich ist in der Breite immer unvollständig, in der Tiefe aber durchgängig, weil die Schüler der allmählich zunehmenden Abstraktion bewusst folgen können. Sie haben dabei auf jeder Ebene Gelegenheit, Komponenten selbst zu bauen und auf der nächsthöheren Ebene auch weiter zu verwenden.

Alle Elemente des Unterrichtsganges können natürlich auch einzeln unterrichtet werden. Besonders reizvoll wird der Unterrichtsgang im Zusammenhang, wenn es gelingt, einen Bogen über alle Abstraktionsebenen zu schlagen. Aber auch für sich allein sind die Elemente durchaus als allgemeinbildend anzusehen, weil alle dazu beitragen, die Arbeits-

[1] Faust-Gymnasium Staufen, Krichelnweg 1, 79219 Staufen, urs.lautebach@fgst.de
OpenPGP-Schlüsselfingerprint: 5F95 477B 22A8 2D9D 66DA 3D0A 6E23 BC3B 8CCE ACB1

weise des Computers zu entmystifizieren.

Die unterrichtliche Wiederverwendung wird auch dadurch erleichtert, dass die meisten Elemente nicht an bestimmte Werkzeuge gebunden sind. Lediglich das eigens erstellte Assemblerwerkzeug ist gezielt auf den Simulator MIKROSIM abgestimmt.

Darüber hinaus soll auch angeregt werden, die Lernenden im Informatikunterricht (mit dem genannten „Durchstich“ vor Augen) Zusammenhänge über viele Abstraktionsebenen hinweg herzustellen zu lassen. Denn ein Schlüsselmerkmal der Informatik ist ja, dass sie auf vielen solcher Ebenen „stattfindet“. Darum sollte die Fähigkeit, die jeweils passende Ebene gezielt auszuwählen, eines der Ziele jeder Informatikausbildung sein. Der geschilderte Unterrichtsgang lässt die Lernenden verschiedene Ebenen bewusst kennenlernen und gedanklich miteinander verbinden.

2 Unterrichtsgang

2.1 Logische Gatter

Anknüpfend an den aus einer Programmiersprache bekannten UND-Operator beginnt der Unterrichtsgang bei logischen Verknüpfungen, Gattern und den entsprechenden Wertetabellen. Dafür kommen der Simulator Logicsim von www.tetzl.de oder (falls verfügbar) Logikstecksysteme zum Einsatz.

Nach Einführung des Binärsystems bauen die Schüler mindestens Halb-, Volladdierer und Addierwerke; als Leitgedanke bietet sich „Rechnen mit Strom“ an (Gallenbacher, „Abenteuer Informatik“ [Ga12]). Je nach Zeitbudget entstehen auch noch andere Bauteile, die den Schülern in der nächsten Phase als fertige Bausteine innerhalb von Mikrosim wieder begegnen. Das können etwa Flipflops, Subtrahierer, Multiplexer oder umschaltbare Rechenwerke sein. Für die Arbeit mit Mikrosim muss das Hexadezimalsystem beherrscht werden; meistens ist auch eine Einführung des Zweierkomplements sinnvoll.

2.2 Prozessormodell

Diese nächste Phase stützt sich wesentlich auf „Mikrosim“, einen Simulator für einen einfachen von-Neumann-Prozessor mit 8-Bit Wort- und Adressbreite (Konrad Dammeier, Pflege mittlerweile Thomas Schaller, siehe [DS02]). Er besteht für die Schüler sichtbar aus Registern, Bussen, Steuer- und Rechenwerk, was eine nahtlose Anbindung an das zuvor Behandelte gewährleistet. Die Schüler lernen hier erst die Datenflusssteuerung innerhalb eines Prozessors kennen und automatisieren sie dann über den Mikroprogrammspeicher.

Für eine Auswahl des Maschinenbefehlssatzes bauen sie schließlich Mikroprogramme

für einfache (MOV AX, BX) und komplexere Assemblerbefehle (ADD [BX+nn]) bis hin zu bedingten Sprüngen (JPZ) und Stackbefehlen (z.B. PUSH AX, POP AX). Dieser Teil lehnt sich stark an entsprechende Materialien von M. Makowsky, K. Dammeier und anderen an.

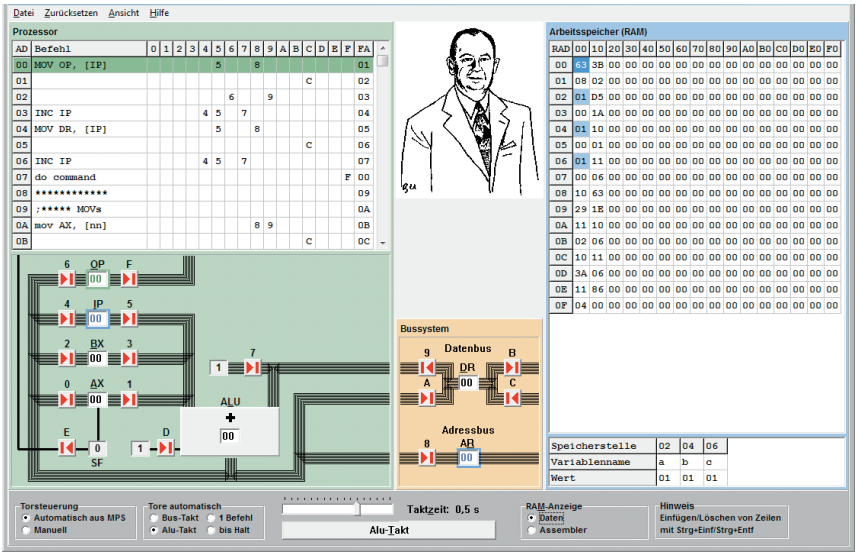


Abb. 1: Simulator „Mikrosim“. Links (grün) der eigentliche Prozessor mit Mikroprogrammspeicher und darunter Registern, Bussen, Toren und Rechenwerk (ALU). Rechts 256 Byte RAM. Der Simulator kann bis auf Ebene einzelner Tore von Hand gesteuert werden, aber auch Takte, Befehle oder ganze Programme in wählbarer Geschwindigkeit automatisch abarbeiten.

Darüber hinaus werden spezielle Stackarithmetikbefehle wie ADDSTACK („entferne die beiden obersten Stackeinträge und lege ihre Summe wieder auf den Stack“) eingeführt, um später Postfixausdrücke elegant auswerten zu können.

Auch in dieser Phase werden nur so viele Mikroprogramme erstellt, dass den Schülern klar wird, wie der Prozessor zu seinem Assemblerbefehlssatz kommt. Den vollständigen (und weitgehend fehlerfreien) Befehlssatz für die nächste Phase bekommen die Schüler gestellt. Daran lernen sie Grundzüge der Assemblerprogrammierung.

2.3 Assembler

Die Doppelverwendung des Wortes „Programm“ in Mikroprogramm(-speicher) und Assemblerprogramm ist für Schüler verwirrend, die Abgrenzung dementsprechend schwierig. Hier hilft die Klarstellung, dass der linke Teil des Prozessors „von Intel-Ingenieure entworfen“ und programmiert wird, die ihn dann fertig eingegossen und unveränderbar verkaufen; das RAM rechts hingegen ist auch für Endkunden zugänglich und veränder-, also programmierbar. An dieser Stelle wechselt der Schüler also aus der

Ingenieurs- in die Kundenrolle.

Mikrosim bietet für die Assemblerprogrammierung ein elegant integriertes Werkzeug, das RAM-Inhalte automatisch als Maschinencodes interpretiert und sofort die entsprechenden Mnemonics darstellt. Es ist unkompliziert, eignet sich aber nur für sehr kurze Programme bis etwa 5 Zeilen. Der Autor hat daher ein eigenes erstellt, das im Assemblerquelltext auch Sprunglabel, Konstanten und Kommentare verarbeiten kann; außerdem kommt es damit zurecht, dass Assemblerbefehle bei verschiedenen Schülern an unterschiedlichen Mikroadressen stehen und deswegen auch unterschiedliche Opcodes haben können.

Die Schüler sollen nach dieser Phase gerade so viel Assembler beherrschen, dass sie später Assemblerprogramme vom Compiler auch automatisch erzeugen lassen können. Dafür reicht es, sehr einfache Programme wie Zuweisungen, Entscheidungen und einfache Arithmetik von Hand assemblieren zu können:

```
if(a < 3)                          MOV AX, [var_a:]
{                                  SUB 3
    b = 5;                         JPS else:
}                                  MOV AX, 5
else                               MOV [var_b:], AX
{                                  JP ende:
    c = c + 3;               else: MOV AX, 3
}                                  ADD [var_c:]
                                   MOV [var_c:], AX
                             ende: ...
```

Abb. 2: Einfaches Java-Programm, Übersetzung in Assembler

Stärkere Schüler bekommen etwa Schleifen als Zusatzaufgabe, z.B. für eine Multiplikationsroutine. Unabhängig davon und mit Blick auf den Leitgedanken „Wechsel der Abstraktionsebenen“ sollen die Schüler beim Programmieren in Assembler aber auch spüren, dass komfortable Hochsprachen durchaus keine Selbstverständlichkeit sind.

Bis hierher hat der Unterricht zwar vorgeführt, wie man aus Gattern einen in Assembler programmierbaren Rechner baut; aber erst der Brückenschlag zur Hochsprache führt die Schüler zurück auf vertrautes Terrain und vervollständigt damit den Durchstich.

2.4 Compiler

Aus didaktischer Sicht reicht für den Compiler die Sprache der arithmetischen Ausdrücke (also „7+2“ oder „5*(6-(8-5))“): Sie ist allen vertraut, die zugehörige Grammatik ist mit nur 11 Regeln noch übersichtlich. Andererseits ist diese Sprache aber wegen der

Rekursion immerhin kontextfrei. Das ist auch ein wichtiges Merkmal realer Programmiersprachen, das die Konstruktion von Compilern maßgeblich beeinflusst.

Die Schüler wandeln solche Ausdrücke zunächst von Hand in Postfix-Schreibweise um (für die obigen Beispiele also „7, 2, +“ bzw. „5, 6, 8, 5, -, -, *“) und lernen deren Vorteile kennen: Erstens ist sie auch ohne Klammern immer eindeutig, zweitens kann der Postfixausdruck direkt als Anweisungsfolge für eine Stackmaschine gelesen werden. Mit den genannten Assemblerbefehlen zur Stackarithmetik steht eine solche Stackmaschine den Schülern bereits zur Verfügung.

Der letzte Schritt ist dann natürlich die Automatisierung der Infix-Postfix-Übersetzung. Mittlerweile händigt der Autor den Schülern die dafür fertig vorbereitete Grammatik aus: Sie erlaubt eine Analyse durch rekursiven Abstieg mit 1-Token-Vorausschau („lookahead“).

```
(1) expr       → term restExpr EOF
(2) term       → faktor restTerm
(3) restExpr   → +  term restExpr
(4)            | -  termrestExpr
(5)            | empty
(6) restTerm   → *  faktor restTerm
(7)            | /  faktor restTerm
(8)            | empty
(9) faktor     → ZAHL
(10)           | VAR
(11)           | ( expr )
```

Abb. 3: Grammatik für arithmetische Ausdrücke (nach [Gu93])

Der resultierende Parser ist ein LL(1)- oder recursive-descent-Parser. Die leider überhaupt nicht intuitive Struktur der Grammatik gewährleistet, dass Operatoren mit ihren Präzedenzen und der gewünschten Linksassoziativität richtig interpretiert werden. Man kann das kurz kommentieren, die Konstruktion der Grammatik aber weglassen. Es hat sich nicht bewährt, Linksrekursion und Eindeutigkeit an dieser Stelle auch noch zu behandeln.

Den Ablauf der Analyse („parsing“) erleben die Schüler zunächst anhand eines Kartenspiels, in dem jede der 11 Produktionen in Form von Karten vorkommt. Mit den folgenden „Spielregeln“ ergibt sich die Funktionsweise eines recursive-descent-Parsers:

- Man schaut in der Eingabe nur ein Terminal voraus, der Rest bleibt abgedeckt;
- man legt Produktionen in Form von Papierkarten auf einen Stapel;
- man markiert auf der obersten Karte jeweils den Stand ihrer Abarbeitung;
- fertig abgearbeitete Karten wirft man weg und arbeitet auf der Karte weiter, die darunter sichtbar wird, und zwar an der vorher markierten Stelle;

- man streicht „verbrauchte“ Terminale aus der Eingabe heraus.

Auf diese Weise wird die Eingabe „unplugged“ auf Gültigkeit geprüft („geparst“). Hier hat es sich bewährt, auch ungültige Ausdrücke wie „4*)5-7“ parsen zu lassen, damit klar wird, dass und wann solche Fehler bei der Analyse auffallen.

Im nächsten Schritt wird die Grammatik so erweitert, dass sie in den richtigen Momenten auch Ausgaben tätigt. Damit können die Schüler dann schon auf Papier einen „Unplugged-Compiler“ durchspielen, der Infix- in Postfixausdrücke umwandelt. Schon nach wenigen Durchläufen wird klar, dass der Vorgang tatsächlich weder weit reichende Vorausschau, noch Kreativität oder Intuition verlangt – dass er also ohne weiteres automatisierbar ist.

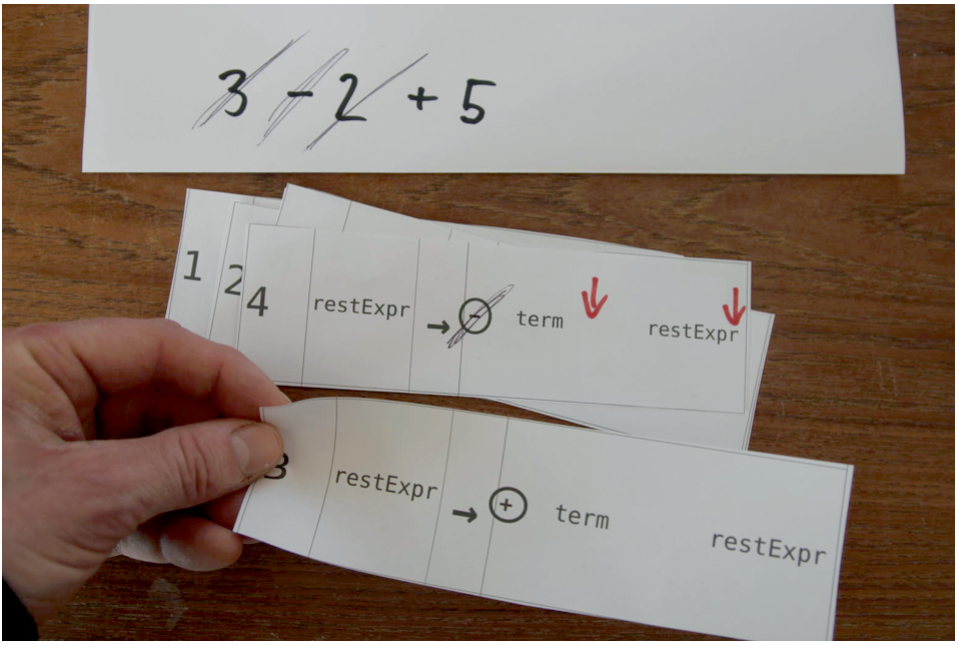


Abb. 4: Kartenspiel für den „Unplugged-Compiler“. Oben die Eingabe, aus der drei Zeichen bereits verbraucht sind. In der Mitte der Kartenstapel; die neue Karte wird gerade aufgelegt.

An diesem Punkt, bekommt die Lerngruppe den Compiler als vorbereitetes Java-Projekt. Darin sind unter anderem Datei-Ein-/Ausgabe, lexikalische Analyse und eine rudimentäre Fehlerbehandlung schon fertig programmiert; die Schüler ergänzen „nur noch“ den Kern des Compilers, also die eigentlichen Grammatikregeln mit Ausgaben (nach Assembler) exakt so, wie sie das zuvor am Kartenspiel geübt haben.

Der fertige Compiler erzeugt dann für jeden Ausdruck ein Programm in der Assemblersprache, die die Schüler zuvor am Simulator selbst gebaut haben. Nach dem Assemblieren ist es auf diesem Prozessor auch tatsächlich und beobachtbar lauffähig.

Die Einheit ist an dieser Stelle eigentlich zu Ende, das Ziel erreicht: Die Schüler haben vom AND-Gatter bis zum Compiler auf jeder Abstraktionsebene selbst Hand angelegt, nach und nach einen (im Simulator) funktionsfähigen Computer aufgebaut und sogar eine Komponente eines Betriebssystems selbst programmiert. Die Übersetzung funktioniert, man kann den übersetzten Programmen beim Ablaufen zuschauen und sie liefern das korrekte Ergebnis. In einer für die Schüler transparenten Art Weise baut all das letztlich auf die Gatter, mit denen die Einheit begonnen hat; auf dem Weg von dort bis zum Compiler erleben sie jede Menge raffinierte Informatik.

2.5 Zusammenfassung

Mit einem Umfang von gut 20 Doppelstunden ist die Einheit für Schüler aber doch schwer zu überblicken. Um die angestrebte Transparenz trotzdem zu gewährleisten, lässt der Autor sie an diesem Punkt eine Zusammenfassung erstellen, die möglichst viele Zusammenhänge innerhalb des ganzen Unterrichtsverlaufs wiedergeben soll. Einige Leitfragen helfen den Schülern bei der Strukturierung:

- Welche Bauteile setzen sich aus welchen anderen zusammen?
- Wo ist Mathematik drin, wo Informatik?
- Welche Softwarewerkzeuge sind beteiligt? Was verarbeiten sie, was liefern sie und an wen?
- Wo ist eigentlich die Grenze zwischen Soft- und Hardware?
- Wo ist menschliche Kreativität im Spiel, was kann automatisch ablaufen?

Die Schüler wählen die Form dieser Zusammenfassung selbst. Sie wurde schon als Mindmap, Flowchart, Prezi, Landkarte und mehrfach auch als handgezeichneter Comic gestaltet. Sichtung der Heftaufschriebe, Auswahl der Inhalte und Konzeption der Darstellung nehmen nochmals Unterrichtszeit in Anspruch, die hier aber gut investiert ist.

Zumindest im Nachhinein gelingt damit fast allen Schülern der gedankliche Brückenschlag über die behandelten Ebenen hinweg; er kommt sonst allenfalls im Informatikstudium zustande und dort auch nicht in dieser gebündelten Form.

2.6 Mögliche Erweiterungen

Der gesamte Unterrichtsgang wurde im vierstündigen Kurs der gymnasialen Oberstufe bisher fünf Mal gehalten und im Lauf der Zeit auf die Teile reduziert, die für den angestrebten Durchstich wesentlich sind. Natürlich gibt es viele denkbare und reizvolle Erweiterungen. So kann man die vom Compiler übersetzte Sprache auch noch schrittweise ausbauen: Statt eines Ausdrucks übersetzt er dann eine Zuweisung

```
variable = ausdruck;
```

und als nächstes eine Kette von Zuweisungen. Leistungsstarke Schüler dürfen dem Compiler eventuell noch boole'sche Ausdrücke wie

```
ausdruck < ausdruck
```

und dann Kontrollstrukturen wie

```
while(boolescherausdruck)
{
    zuweisungskette
}
```

beibringen. In dieser Bonusaufgabe spielen Theorie und Praxis auf lehrreiche Weise zusammen, weil man für jede Änderung der Grammatik die neuen first- und follow-Mengen der einzelnen Regeln ermitteln muss, um die Entscheidungen des Parsers sauber programmieren zu können (für arithmetische Ausdrücke bekamen die Schüler diese Mengen fertig vorgegeben).

Weil das mühsam ist und auch erkennbar unkreativ, also eine Tätigkeit, die man automatisieren sollte, hat der Autor in der Vergangenheit noch einen Parsergenerator vorgeführt, der genau das tut: Er transformiert eine maschinenlesbar notierte Grammatik in einen RD-Parser. Es ist gar nicht schwer, damit den Umfang der übersetzten Sprache nochmal deutlich zu erweitern, etwa um Variablendeklarationen, diverse Kontrollstrukturen und andere Sprachmittel. Das fasziniert leistungsstarke Schüler, ist aber für das übergeordnete Lernziel nicht nötig.

3 Materialien

Die verwendeten Softwarewerkzeuge, Übersichten, Beispiele, Arbeitsblätter und Musterlösungen stellt der Autor auf Anfrage per E-Mail zur Verfügung. Rückmeldungen, Vorschläge und Kritik sind willkommen.

Literaturverzeichnis

[DS02] Dammeier, K.; Schaller, T.: https://www.informatik.schule-bw.de/index.php?id=55&aid=55&mid=47.

[Ga12] Gallenbacher, J.: Abenteuer Informatik – IT zum Anfassen von Routenplaner bis Online-Banking, Spektrum, 2012.

[Gu93] Gumm, H.-P.: Vorlesungsskript Informatik IV, 1993.

Jens Gallenbacher (Hrsg.): INFOS 2015 – Informatik allgemeinbildend begreifen
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 247

Concept-Maps als Mittel zur Visualisierung des Lernzuwachses in einem Physical-Computing-Projekt

Mareen Przybylla[1] und Ralf Romeike[2]

Abstract: Im Unterricht stellt sich häufig die Frage, wie sich der Lernzuwachs der Schüler so sichtbar machen lässt, dass das tatsächliche Verständnis der erlernten Konzepte und ihrer Zusammenhänge untereinander deutlich wird. Eine Möglichkeit hierfür sind Concept-Maps. Mittels Concept-Maps lassen sich Begriffe und ihre Zusammenhänge in vernetzter Struktur visualisieren. Dies entspricht einer konstruktivistischen Sichtweise, nach der Lernen die Bildung von aufeinander aufbauenden Wissensstrukturen bedeutet. In diesem Beitrag wird der Einsatz von Concept-Maps zur Sichtbarmachung von Wissensstrukturen im Anschluss an ein Physical-Computing-Projekt geschildert, sowie der Prozess und die Ergebnisse der Analyse vorgestellt und diskutiert. Es werden Concept-Maps mit unterschiedlichen Fokusfragen verglichen, um Tendenzen des Einflusses der Fragestellung auf die Qualität der resultierenden Concepts-Maps zu gewinnen. Die Erkenntnisse aus der Analyse werden genutzt, um die Ziele des Projektes mit den tatsächlichen Resultaten zu vergleichen und Schlüsse für die Weiterentwicklung dieses sowie die Entwicklung weiterer Physical-Computing-Projekte zu ziehen.

Keywords: Concept-Maps, Visualisierung von Wissensstrukturen, Physical Computing

1 Einleitung

Im Informatikunterricht, insbesondere dann wenn die Schüler in Projekten arbeiten, stellt sich häufig die Frage, wie sich der Lernzuwachs der Schüler so sichtbar machen lässt, dass das tatsächliche Verständnis der erlernten Konzepte und ihrer Zusammenhänge untereinander deutlich wird. Eine besondere Herausforderung hierbei stellt Physical Computing als Unterrichtsgegenstand dar. Hier lernen die Schüler gleichzeitig in verschiedenen Themenbereichen: Programmierung, eingebettete Systeme, Vernetzung von interaktiven Objekten und Vieles mehr sind in diesem Themenbereich relevant. Mögliche Projekte reichen von interaktiver Kleidung und intelligenten Kuscheltieren über raumfüllende Kunstinstallationen bis hin zu nützlichen Alltagsgegenständen. Aus technischer Sicht haben all diese interaktiven Objekte und Systeme gemeinsam, dass sie Spannungswandler in Form von Sensoren und Aktoren nutzen und kontinuierlich laufen, also stetig mit ihrer Umgebung interagieren. Die Vielseitigkeit möglicher Projekte einerseits, aber auch das große Spektrum relevanter Themengebiete, macht es äußerst schwierig, in einem traditionellen Test, einer Klausur oder auch einer Präsentationsprüfung den Lern-

[1] Universität Potsdam, Institut für Informatik, Didaktik der Informatik, August-Bebel-Str. 89, 14482 Potsdam, przybyll@uni-potsdam.de
[2] FAU Erlangen-Nürnberg, Department Informatik, Didaktik der Informatik, Martensstr. 3, 91059 Erlangen, ralf.romeike@fau.de

zuwachs der Schüler zu erfassen. Eine Möglichkeit, das Gesamtbild sinnvoll zu ergänzen, bieten Concept-Maps. Hiermit lassen sich Begriffe und ihre Zusammenhänge in vernetzter Struktur visualisieren, statt nur singuläres Faktenwissen darzustellen.

2 Concept-Maps zur Einschätzung des Wissensstandes

Concept-Maps, auch „Begriffsnetze“ oder wörtlich „Konzeptkarten“ genannt, sind grafische Mittel zur Organisation und Darstellung von Wissensstrukturen. Sie bestehen aus in Form von als Knoten dargestellten Begriffen (Konzepten) und mit verbindenden Teilsätzen beschrifteten gerichteten Kanten, über welche die Konzepte miteinander zu Propositionen verbunden werden (Beispiel siehe Abb. 1). Konzepte stellen dabei Bezeichnungen für wahrgenommene Regelmäßigkeiten in Ereignissen oder Objekten dar, Propositionen treffen Aussagen über die Beziehung zwischen zwei Konzepten. Eine Proposition ergibt somit eine semantische Einheit (vgl. [NC08; St04]).

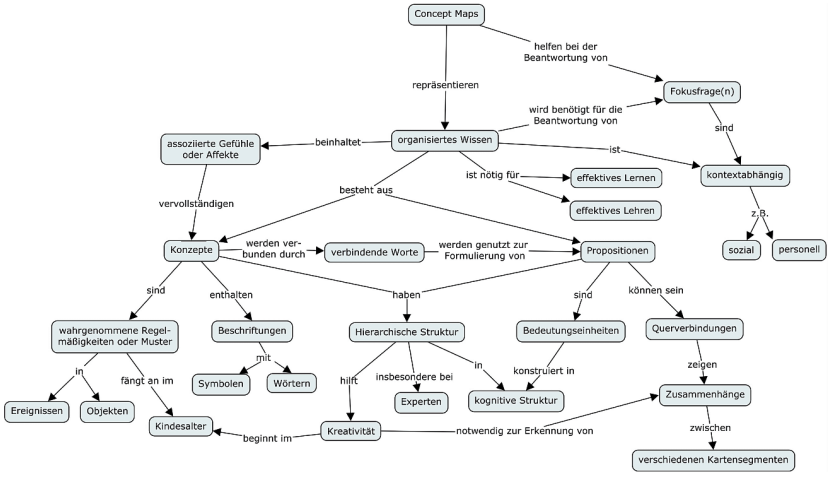


Abb. 1: Concept-Map über Concept-Maps (übersetzt von Novak & Cañas, 2008).

Mit Hilfe solcher Begriffsnetze lassen sich komplexe Sachverhalte anschaulich erklären, insbesondere die Zusammenhänge zwischen einzelnen Fakten. Concept-Maps sollten immer in Verbindung mit einer geeigneten Fokusfrage [NC08] bzw. Aufgabe [RS96] erstellt werden, welche die Denkrichtung bei der Erstellung des Begriffsnetzes lenkt. Bei der qualitativen Analyse von Concept-Maps bietet es sich an, zunächst die Struktur des Begriffsnetzes als Ganzes oder einzelne Teile näher zu betrachten. In der Literatur finden sich im Wesentlichen fünf Strukturtypen: *Kettenstrukturen*, *Ringstrukturen*, *Nabe-Speichen-Strukturen*, *hierarchische Baumstrukturen* und *Netzstrukturen* (vgl. [DSC07; KH08; RS96; Va05; Yi05]). Im Rahmen der Auswertung kann neben der Beurteilung der Struktur bezüglich konkreter Inhalte auch die Analyse der Propositionen fokussiert werden. Um die Güte der Propositionen zu beurteilen, schlägt Ruiz-Primo [Ru00] ein

fünfstufiges Bewertungssystem vor (siehe Tab. 1).

Beurteilung	Beschreibung
Akkurat Exzellent	Herausragende Beschreibung. Vollständig und korrekt. Zeigt tiefes Verständnis der Relation zweier Konzepte.
Akkurat Gut	Vollständige und korrekte Proposition. Zeigt ein gutes Verständnis der Relation zweier Konzepte.
Akkurat Schwach	Korrekte, aber unvollständige Beschreibung. Zeigt teilweises Verständnis der Relation zweier Konzepte.
Nebensächlich	Trotz ihrer Korrektheit wird durch die Proposition kein Verständnis für die Relation zweier Konzepte ersichtlich.
Ungenau/Ungültig	Falsche Proposition.

Tab. 1: Beurteilung der Qualität von Propositionen (In Anlehnung an Ruiz-Primo, 2000)

Nicoll [Ni01] kategorisiert Propositionen mit einem dreistufigen System nach *Nützlichkeit*, *Stabilität* und *Komplexität*. Insbesondere die Nützlichkeit ist ein Merkmal, welches Ausschluss über die Qualität von Concept-Maps geben kann. In dieser Kategorie wird zwischen Propositionen unterschieden, die als nützlich, falsch und unvollständig angesehen werden. Nützlichkeit wird hierbei vom Autor wie folgt definiert: „The utility of the link was interpreted as useful if the link was correct and allowed students to correctly solve chemical proplems." [ebd.] In den Informatikkontext eingebettet könnten Verbindungen also dann als nützlich angesehen werden, wenn sie korrekt sind und Lernenden ermöglichen, informatische Probleme korrekt zu lösen.

3 Verwendung von Concept-Maps im Projekt „My Interactive Garden"

Das Physical-Computing-Projekt „My Interactive Garden" (MyIG, vgl. [PR12; PR13]) wurde während des Schuljahres 2014/15 in einem geteilten Wahlpflichtkurs Informatik einer zehnten Klasse eines Gymnasiums durchgeführt. Bei diesem Kurs handelt es sich um Anfangsunterricht, etwa für die Hälfte der Schüler ist es der erste Informatikkurs. Die Teilung des Kurses hatte den Vorteil, dass die Erkenntnisse des ersten Durchlaufs beim erneuten Unterrichten mit der zweiten Teilgruppe bereits zur Verbesserung des Projektablaufs herangezogen werden konnten. Neben weiteren Evaluationsmitteln erstellten beide Schülergruppen am Ende des Projektes Concept-Maps.

3.1 Projektziele

Eines der Hauptanliegen von MyIG ist es, den Zusammenhang von Hard- und Softwarekomponenten in eingebetteten und interaktiven Informatiksystemen begreifbar zu machen. Die Schüler sollen im Projekt lernen, dass die zugrundeliegenden Prinzipien in der Funktionsweise von vielen Dingen in unserem Alltag (z.B. Parkplatzschranke, Torlinien-

technik oder Ausleihstation in der Bibliothek) nicht mystisch sind und dass mit den richtigen Werkzeugen jeder aktiv auf kreative Weise eigene interaktive Systeme gestalten kann. Im Projekt geht es darum, geeignete Sensoren und Aktoren zur Interaktion des eigenen interaktiven Objektes mit der Umwelt zu verwenden und die Funktionalität der einzelnen Bauteilgruppen erklären zu können. Im Einzelnen bedeutet das, dass die Schüler zwischen analogen und digitalen Ein- und Ausgängen unterscheiden können und diese zweckmäßig für analoge und digitale Sensoren (z.B. Lichtwiderstand, Temperatursensor, Potentiometer, Berührungssensor und Kippschalter) und Aktoren (z.B. LEDs, Standard- und kontinuierliche drehende Servomotoren und Piezo-Summer) eines vorgegebenen Physical-Computing-Baukastens (Arduino-basierte MyIG Toolbox[3]) verwenden. Hierfür müssen sie außerdem in der Lage sein, größtenteils bekannte Operatoren und Kontrollstrukturen aus einer vorangegangenen Unterrichtseinheit mit Scratch[4] (Sequenzen, Fortlaufend-Schleife, Wiederholung, bedingte Verzweigung) im neuen Kontext sachgemäß zu verwenden, sowie sich die Bedeutung und Funktion von speziellen Blöcken zur Ansteuerung der Hardwarekomponenten mit den bereitgestellten Materialien zu erschließen. Als Programmierumgebung wurde im Unterricht Snap4Arduino[5] verwendet, welches Scratch sehr ähnelt, jedoch speziell für die Verwendung mit Arduino[6] konzipiert wurde.

3.2 Ziel der Erhebung

In einem ersten Durchgang soll zunächst das Messinstrument bewertet werden, welches dann optimiert und im zweiten Durchlauf überprüft und gleichzeitig zur Erhebung der eigentlich angestrebten Daten über die Qualität des Lernzuwachses bei den Schülern verwendet wird. Daher ergeben sich für den ersten Durchlauf folgende Fragestellungen:

(I) Welche Fokusfrage bringt qualitativ hochwertige Concept-Maps hervor?
(II) Welchen Einfluss hat die Vorgabe bzw. Nicht-Vorgabe von Konzepten auf die resultierenden Concept-Maps?

In der zweiten Erhebung wird beurteilt, ob und in welchem Maße die Ziele des Unterrichts erreicht werden konnten. Hierfür sind folgende Fragestellungen leitend:

(I) Wurden die Ziele des Unterrichts erreicht?
(II) Wie wird das Zusammenwirken von Hard- und Software beschrieben?
(III) Wie werden die Fachbegriffe verwendet und zueinander in Relation gesetzt?
(IV) Haben die Schüler ein intuitives oder fundiertes Verständnis erlangt?

[3] MyIG Toolbox: www.greifbare-informatik.de
[4] Scratch: www.scratch.mit.edu - Unterrichtseinheit angelehnt an „Programmieren mit Scratch“ http://www.inf.ethz.ch/personal/gaertner/scratch/ScratchkursNicolet.pdf
[5] Snap4Arduino: www.s4a.cat/snap
[6] Arduino: www.arduino.cc

3.3 Messinstrument und Durchführung

Zunächst wurden aus einer Experten-Map erwartbare Konzepte und Propositionen generiert (vgl. Tab. 2). Diese sollen in der Analyse dazu dienen, Konzepte einzuordnen, um später dann Aussagen über die Propositionen insbesondere zwischen unterschiedlichen Kategorien treffen zu können.

Inhaltliche Kategorie	**Subkategorien**	**Sub-Subkategorie**
Bastelmaterial		
Umgebung		
Elektronische Komponenten	Sensor, Aktor, Bauteilgruppe, Mikrocontroller	
Physikalische Größe		
Interaktives Objekt		
Verhalten	Interaktionen Datenfluss	initiativ, reaktiv
Programm	Hardwaresteuerung, Variablen, Operationen, Kontrollstruktur, Daten, Methode	

Tab. 2**:** Inhaltliche Codes zur Analyse der Konzepte

Ruiz-Primo und Shavelson [RS96] beschreiben drei Arten der Variationen von Concept-Maps: Aufgabenstellung, Rahmenbedingungen und Inhaltsstruktur. Eine Einordnung der hier verwendeten Methode in dieses System ist in Tab. 3 dargestellt.

Aufgabenstellung: Eigenständiges Erstellen einer Concept-Map
Rahmenbedingungen: Begriffe teils vorgegeben, teils nicht vorgegeben; eigene Begriffe sollen hinzugefügt werden; Relationen nicht vorgegeben
Inhaltsstruktur: Teils kettenartige und ringförmige Strukturen in Programmabläufen; netzartige Strukturen beim Zusammenspiel der Komponenten zu erwarten; Hierarchien eher nicht zu erwarten, möglicherweise in der Klassifikation von konkreten elektronischen Bauteilen oder im Erläutern des Programmaufbaus denkbar

Tab. 3: Einordnung des Messinstrument anhand der Klassifikation von [9]

Es war Teil der Untersuchung herauszufinden, welche Rahmenbedingungen für die Ziele der Erhebung geeignet sind. Somit gab es in der ersten Erhebung vier verschiedene Gruppen (siehe Tab. 4).

Fokusfrage / Begriffsvorgabe	Wie interagiert Dein interaktives Objekt mit der Umgebung / Menschen?	Wie funktioniert Dein interaktives Objekt?
Begriffe vorgegeben	A	C
Keine Vorgabe	B	D

Tab. 4: Gruppen innerhalb der ersten Erhebung

Die Erhebung wurde jeweils am Ende der Projektphase, zu Beginn der ersten Unterrichtsstunde nach dem Projektende durchgeführt. Durchführungsobjektivität wurde somit gewahrt, da alle Schüler einer Gruppe unter den selben Bedingungen arbeiteten. Die Concept-Maps wurden anhand vorab festgelegter Kategorien inklusive Beschreibung und Beispiel beurteilt, so dass eine Objektivität in der Auswertung in diesem praxisrelevanten Rahmen insbesondere bei den aus anderen Untersuchungen übernommenen und diesbezüglich überprüften Kategorien angenommen werden kann. Zur Überprüfung wurde die Interrater-Reliabilität stichprobenartig kontrolliert. Dabei wurde eine durchschnittliche Übereinstimmung zweier unabhängig voneinander arbeitender Beurteiler von 70,8% gemessen. Dieser Wert kann angesichts des hohen inhaltlichen Analyseanteils als positiv eingestuft werden. Bezüglich der Validität wurde aus Gründen der Praxistauglichkeit auf einen Vergleichstest verzichtet. Stattdessen wurden die Ergebnisse der Concept-Maps mit den Eindrücken aus dem Unterricht und Ergebnissen aus einer mündlichen Befragung (außerhalb einer Prüfungssituation) verglichen. Dabei entstand der Eindruck, dass die Concept-Maps die Ergebnisse aus der mündlichen Befragung gut reflektieren.

4 Ergebnisse

Zur Analyse der Begriffsnetze wurde zunächst jeweils die gesamte Concept-Map hinsichtlich Struktur und Umfang bewertet. Anschließend fand eine Begutachtung von Inhalt und Sprache (umgangssprachlich vs. fachsprachilch korrekt, ungenau oder falsch) der Konzepte statt. Danach wurden die einzelnen Propositionen hinsichtlich der Sprache untersucht und inhaltlich beurteilt, ob Verbindungssätze Zugehörigkeiten, Hierarchien, Datenfluss oder Verhalten (Aktionen vs. Wirkweisen) beschreiben. Wie bereits oben erläutert, fand schließlich eine Analyse hinsichtlich Nützlichkeit, Komplexität und Qualität der Propositionen statt.

4.1 Erste Erhebung

In der ersten Erhebung haben die Schüler auf Grund fehlender Zeit vorab keine Einführung ins Concept-Mapping erhalten. Sie hatten daher lediglich eine kurze Erklärung und ein Beispiel auf Papier als Grundlage zum Anfertigen ihrer eigenen Begriffsnetze. Vermutlich ist dies ein Grund für die mangelnde Qualität einiger Concept-Maps, welche nicht den angegebenen Regeln folgen (z.B. fehlen Pfeilspitzen oder Beschriftungen, siehe Abb. 2 li.). Zudem standen die Schüler unter Zeitdruck, da sie eine feste Vorgabe von 20 Minuten zum Erstellen der Concept-Maps als Teil eines größeren Fragebogens bekamen. Einigen Schülern fehlte dadurch am Ende die Zeit, ihre Concept-Maps zu erstellen. Das Ziel war es, die Schüler am Beispiel ihrer selbst erstellten interaktiven Objekte das allgemeine Prinzip eben solcher erklären zu lassen. Was tatsächlich mit den verschiedenen Aufgabenstellungen erreicht wurde, war, dass die Schüler ihre konkreten Objekte erklärten und nicht abstrahierten und allgemeine Prinzipien beschrieben. Dies

wurde neben dem Gesamteindruck bei der Analyse der Begriffsnetze vor allem an zwei Merkmalen deutlich. Betrachtet man die Nennung von Bauteilgruppen im Begriffsnetz, fällt auf, dass diese in keiner der Gruppen vorkommen. Stattdessen wurden immer konkrete Bauteile benannt, wie z.B. *Schalter*, *Servo* oder *LED*. Gleichzeitig werden von lediglich zwei aus insgesamt 89 Propositionen in den Concept-Maps aller Schüler Wirkweisen beschrieben. Die überwiegende Anzahl von Verbindungssätzen erklärt Zugehörigkeiten, wie z.B. *[Safe] – hat –> [Knöpfe]* oder konkrete Aktionen, wie z.B. *[Servo] – betätigt –> [Schalter]*. Ein wahrnehmbarer Unterschied besteht zwischen den Gruppen A/C und B/D, also in Abhängigkeit davon, ob Begriffe vorgegeben wurden. Die Stabilität der Aussagen von Schülern der Gruppen A/C wurde insgesamt acht Mal (von 45; 17,8%) als vage eingestuft. Bei den Lernenden der Gruppen B/D kam dies kein einziges Mal vor. Dies deutet darauf hin, dass die Vorgabe von Konzepten Schüler dazu anregt, über bisher nicht vorhandene Verknüpfungen nachzudenken und sich den Sinn zu erschließen, während die Aufgaben ohne Konzeptvorgaben nicht dazu anregen, derartige Verknüpfungen zu erstellen.

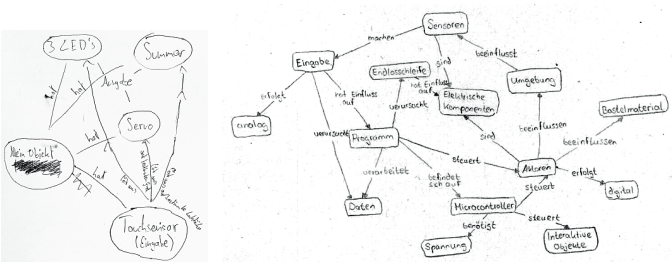


Abb. 2: Fehlerhafte (li.) und gelungene (re.) Concept-Map aus den Schülerdaten.

4.2 Zweite Erhebung

Die Ergebnisse der ersten Erhebung zeigen deutlich, dass die Aufgabenstellung nicht die gewünschten Ziele erreichen konnte. Die Schüler wurden nicht angeregt, das Gelernte in allgemeiner, abstrakter Form zu erklären, sondern blieben auf der Stufe des konkreten Objekts. Um dies zu vermeiden, wurde die Aufgabenstellung für die zweite Erhebung gänzlich überarbeitet. Dazu wurde die Fokusfrage allgemein und für alle Schüler identisch gestaltet, so dass sie ihre selbst erstellten Projekte höchstens aus eigener Initiative für beispielhafte Vergleiche heranziehen: „Wie funktionieren interaktive Objekte?“. Es zeigte sich, dass die Vorgabe von Begriffen den Prozess positiv beeinflusst, daher wurden in der zweiten Erhebung Begriffe vorgegeben. Zusätzlich bekamen die Schüler keine Zeitvorgabe, sondern konnten solange an den Begriffsnetzen arbeiten, bis sie diese abgeben wollten. Vor der Erhebung lernten die Schüler Concept-Maps an Beispielen kennen. Im Vergleich zur ersten Erhebung kann ein deutlicher Anstieg in der Anzahl der verwendeten Konzepte verzeichnet werden. Während in der ersten Erhebung durchschnittlich 5,3 Konzepte verwendet wurden, waren es in der zweiten Erhebung 11,1.

Dies kann an der verbesserten Aufgabenstellung liegen, aber auch auf die zur Verfügung stehende Zeit zurückzuführen sein. Die Zahl ungültiger/ungenauer Propositionen reduzierte sich auf 32,6% (42% in der ersten Erhebung) und es gab einen deutlichen Anstieg in der Beschreibung von Bauteilgruppen statt konkreter Sensoren und Aktoren, was darauf hindeutet, dass die Erklärungen allgemeiner gehalten sind und vom konkreten Objekt abstrahiert. Außerdem wurde mit der Aufgabenstellung erreicht, dass deutlich mehr Propositionen Wirkweisen beschreiben, als beobachtbare Aktionen. Erneut tauchten auch Konzepte und Propositionen auf, die sich bisher keiner der Kategorien zuordnen ließen. Besonders auffällig waren hier Beschreibungen von Voraussetzungen oder Abhängigkeiten, gekennzeichnet durch Worte wie *benötigt*, *benutzt*, *braucht*, etc. Hinsichtlich der oben beschriebenen Fragestellungen (vgl. Abschnitt „Ziel der Erhebung“) konnte Folgendes festgestellt werden:

Wurden die Ziele es Unterrichts erreicht? Insgesamt entsteht der Eindruck, dass die überwiegende Anzahl der Schüler in der Lage ist, zu erklären wie interaktive Objekte funktionieren. Einige auffällige Häufungen bei ungenauen oder ungültigen Beschreibungen finden sich vor allem in der Verknüpfung schwieriger Konzepte (z.B. Pulsweitenmodulation) oder solcher Konzepte, die im Unterricht keine oder nur eine untergeordnete Rolle spielten (z.B. Spannung). Das Ziel sowohl des verwendeten Baukastens, als auch der Programmierumgebung ist es, solche schwierigen Konzepte für den Anfangsunterricht auf ein einfaches Level mit geringer Einstiegshürde zu bringen, häufig durch die Verwendung von Blackboxen. Auch qualitativ als akkurat-exzellent einzustufende Propositionen, wie sie in einer Expertenmap vorhanden sein könnten, finden sich entsprechend nicht in den Daten wider.

Wie wird das Zusammenwirken von Hard- und Software beschrieben? Die Verknüpfungen zwischen Konzepten sind insgesamt überwiegend umgangssprachlich erfolgt. Sehr häufig wurden dabei fundamentale Fakten genannt, sehr selten Beispiele verwendet. Verbindungen zwischen dem Programm, welches das interaktive Objekt steuert und der Hardware, welche das interaktive Objekt ausmacht, wurden meist sehr oberflächlich behandelt (z.B. *[Mikrocontroller] – hat –> [Programm]*). Die Struktur eines solchen Programms wurde von nur wenigen Schülern erklärt, allerdings auch nur in Hinblick auf die darin vorhandene Endlosschleife, welche als Konzept vorgegeben war. Die Wechselwirkungen zwischen Programmelementen und den Hardwarekomponenten wurde lediglich von einem Schüler etwas detaillierter beschrieben: *[Mikrocontroller] – führt aus –> [Programm] – gibt Arbeitsanweisungen an –> [Aktoren] – und –> [Sensoren]*.

Wie werden die Fachbegriffe verwendet und zueinander in Relation gesetzt? Insgesamt wurde nur wenig Fachsprache in den Propositionen verwendet. Mögliche Gründe hierfür könnten darin liegen, dass die Konzepte selbst größtenteils fachsprachlich sind und sich schnell und einfach mit umgangssprachlichen Begriffen verbinden lassen. Wenn Fachsprache verwendet wurde (egal ob korrekt, ungenau oder falsch), dann typischerweise zwischen zwei Konzepten unterschiedlicher Art, z.B.: *[Sensoren] – scannen –> [Umgebung]* oder *[Programm] – verarbeitet –> [Daten]*. Die Qualität der Propositi-

onen wurde überwiegend als akkurat schwach (36%) oder akkurat gut (26%), aber auch häufig als ungenau/ungültig (33%) eingestuft. Die Nützlichkeit der Verbindungssätze wurde überwiegend als unterstützend (35%) oder fundamentale Fakten (31%), aber auch häufig als unnütz (26%) eingestuft.

Haben die Schüler ein eher intuitives oder fundiertes Verständnis erlangt? Aus den Begriffsnetzen wurde deutlich, dass die Schüler ein eher oberflächliches Verständnis erlangt haben. Sie handeln nicht rein intuitiv, sondern können die vorgegebenen Begriffe in der Mehrzahl der Fälle in korrekte Zusammenhänge zueinander bringen, jedoch kommen die Wenigsten über ein Verständnis hinaus, welches Ursache und Wirkung beschreibt. Zwar werden in den Concept-Maps Wirkweisen häufiger als konkrete Aktionen beschrieben, jedoch selten Begründungen gegeben. Dies zeigt sich auch darin, dass ein großer Anteil der als nützlich eingestuften Verbindungssätze fundamentale Fakten beschreibt (78,9%), und nur vereinzelt Verbindungen durch umliegende Verbindungen erklärt werden (12,7%). Eine weitere Ursache könnte bei einzelnen Schülern sein, dass es ihnen schwer fällt, sich sprachlich gewandt auszudrücken, was dann zu verkürzten Darstellungen führen könnte, die nicht dem tatsächlichen Verständnis entsprechen. Möglicherweise könnten die Concept-Maps und damit verbunden auch die gebildeten Wissensstrukturen eine höhere Qualität erreichen, wenn dieses Werkzeug im Unterricht regelmäßig zum Lernen und zur Evaluation genutzt werden würde.

5 Fazit und Ausblick

Zusammenfassend kann gesagt werden, dass die zweite Durchführung für die Schüler in vielerlei Hinsicht Verbesserungen brachte. Dies kann nicht ausschließlich auf Veränderungen im Unterricht zurückgeführt werden, denn auch die Fokusfrage der Concept-Maps wurde überarbeitet. Mit der entsprechenden Aufgabenstellung wurde ein Weg gefunden, komplexe und vernetzte Wissensstrukturen, wie sie Schüler in einem Physical-Computing-Projekt erlangen, sichtbar zu machen. Um die Concept-Maps auszuwerten, wurde ein umfangreiches Kategoriensystem entwickelt. Dies muss für den praxistauglichen Einsatz überarbeitet werden. Aus den Daten der Interrater-Agreement-Analyse gehen wichtige Hinweise hervor, welche Codes uneindeutig sind und überarbeitet werden müssen. Eine Analyse der Korrelationen zwischen den Kategorien wird außerdem Aufschluss über möglicherweise redundante Codes geben. Die vorhandenen Daten bieten zahlreiche Möglichkeiten zur tiefergehenden Analyse. So ist beispielsweise vorstellbar, dass durch den Vergleich der Schüler-Maps mit der Expertenmap insbesondere auf nicht vorhandene Verknüpfungen zwischen Konzepten geschaut wird, um zu sehen, welches Verständnis man als Lehrer voraussetzt, das bei den Schülern möglicherweise gar nicht vorhanden ist. Für die Weiterentwicklung des Projektes „My Interactive Garden“ kann vor allem die Schlussfolgerung gezogen werden, dass einige Inhalte, so sie denn als Lernziele verstanden werden, explizit im Unterricht besprochen werden müssen, da die Konzeptionierung des Baukastens und der verwendeten Programmierum-

gebung das Erstellen interaktiver Objekte stark erleichtern und somit das tiefere Verständnis für die zugrundeliegenden Wirkweisen nicht aus sich heraus erfordern.

Literaturverzeichnis

[DSC07] Derbentseva, N.; Safayeni, F.; Cañas, A.J.: Concept maps: Experiments on dynamic thinking. Journal of Research in Science Teaching 44, 3 (2007), S. 448–465.

[KH08] Keppens, J.; Hay, D.: Concept map assessment for teaching computer programming. Computer Science Education 18, (2008), S. 31–42.

[Mo09] Moen, P.: Concept Maps as a Device for Learning Database Concepts. Proceedings of TLAD'09, Higher Education Academy Subject Network for Information and Computer Sciences (2009), S. 29-48.

[MH12] Mühling, A.; Hubwieser, P.: Towards Software-Supported Large Scale Assessment of Knowledge Development. Proceedings of the 12th Koli Calling International Conference on Computing Education Research, (2012), 2010–2011.

[Ni01] Nicoll, G.: A three-tier system for assessing concept map links: a methodological study. International Journal of Science Education 23, (2001), S. 863–875.

[NC08] Novak, J.D.; Cañas, A.J.: The Theory Underlying Concept Maps and How to Construct and Use Them. IHMC CmapTools, (2008), S. 1–36.

[PR12] Przybylla, M.; Romeike, R.: My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education. Proceedings of Constructionism 2012, (2012), S. 395–404.

[PR13] Przybylla, M.; Romeike, R.: Physical Computing mit „My Interactive Garden". Department of Computer Science, CAU Kiel, 2013.

[RS96] Ruiz-Primo, M.A.; Shavelson, R.J.: Problems and issues in the use of concept maps in science assessment. Journal of Research in Science Teaching 33, 1996, S. 569–600.

[Ru00] Ruiz-Primo, M.A.: On the Use Of Concept Maps As An Assessment Tool in Science: What We Have Learned so Far. Revista Electrónica de Investigación Educativa 2, 1 (2000), S. 29–52.

[Ru04] Ruiz-Primo, M.A.: Examining Concept Maps as an Assessment Tool. Concept Maps: Theory, Methodology, Technology. Proc. of the First Int. Conference on Concept Mapping 1, (2004), S. 555–563.

[St04] Stracke, I.: Einsatz computerbasierter Concept Maps zur Wissensdiagnose in der Chemie. Waxmann Verlag, Münster, 2004.

[Va05] Vanides, J., et. al.: Using Concept Maps in the Science Classroom. Science Scope 28, 8 (2005), 27–31.

[Yi05] Yin, Y. et al.: Comparison of two concept-mapping techniques: Implications for scoring, interpretation, and use. Journal of Research in Science Teaching 42, 2 (2005), 166–184.

Prozessorientierte Software-Entwicklung im Informatikunterricht

Hanno Schauer [1]

Abstract: Das Thema Softwareentwicklung ist geeignet, wertvolle Schlüsselkompetenzen zu vermitteln. Es ist aber gleichzeitig ein besonders herausfordernder Unterrichtsgegenstand. Denn es gilt, die Komplexität realer Softwareprojekte im Unterricht erlebbar zu machen, ohne diesen zu überfrachten. Der Beitrag präsentiert eine speziell auf die Bedürfnisse der Schule abgestimmte Software-Entwicklungsmethode, die durch die Anwendung von Prozessmodellierungstechniken dazu beiträgt, viele der Herausforderungen deutlich abzumildern.

Keywords: Software-Entwicklung, Schulinformatik, Prozessmodellierung, Prozessorientierung.

1 Motivation

Die Softwareentwicklung ist ein ebenso reizvolles wie herausforderndes Unterrichtsthema. Sie ist reizvoll, da sich hier in besonderem Maße das Wesen der Informatik zeigt, und weil Schlüsselqualifikationen gefördert werden, die nicht nur für die Informatik von besonderem Wert sind: Das Gestalten von Technologien, das methodische Vorgehen, die prozessorientierte Sichtweise sowie das Arbeiten in Projekten.

Gleichzeitig ist die Softwareentwicklung ein herausforderndes Thema. Denn es gilt, sowohl die Komplexität realer Softwareprojekte als auch die Anwendung informatischer Planungs- und Modellierungsmethoden im Unterricht erlebbar zu machen, ohne diesen zu überfrachten. Dabei sollten charakteristische Aufgaben, Phasen und (typische) Meilensteine für den Lerner klar erkennbar und (jede Phase für sich) sinnhaft sein. Um diesen Spagat zu meistern, bedarf es geeigneter Unterrichtskonzepte.

Eine in Schulbüchern verbreitete Möglichkeit, den dargestellten Herausforderungen didaktisch zu begegnen, ist es, die Softwareentwicklung anhand innerinformatischer Problemstellungen zu behandeln: anhand eines Taschenrechners oder Fahrkartenautomaten [Hu09] oder eines Simulationsprogramms für eine Populationsentwicklung [Du09]. Ein anderer Ansatz ist der Verweis auf ein Pflichtenheft, welches bereits alle Anforderungen an eine Software beschreibt [Br09].[2]

In beiden Fällen sind die Anforderungen an eine Software weitestgehend vorgegeben. Die praktischen Übungen reduzieren sich auf die Phasen des Software-Entwurfs (mit UML-Klassendiagrammen; zuweilen auch mit Zustandsdiagrammen [Hu09]) sowie die Implementierung der Software. Insbesondere die frühen Phasen der Softwareentwick-

[1] Mons-Tabor-Gymnasium Montabaur, Von-Bodelschwingh-Straße 35, 56410 Montabaur, hanno.schauer@uni-due.de

[2] Andere Schulbücher klammern das Thema Softwareentwicklung gänzlich aus (z. B. [En06], [Ha10]).

lung, die Analyse- und frühe Entwurfsphasen, bei denen in realen Software-Projekten mit Anwendern bzw. Auftraggebern zusammengearbeitet wird, werden hierbei ausgeklammert. Dabei werden die frühen Phasen in den Einheitlichen Prüfungsanforderungen der KMK explizit für das Abitur gefordert ([KM04], S. 5): „Grundprinzip des Modellierens als zielgerichtetes Vereinfachen und strukturiertes Darstellen von Ausschnitten der Wirklichkeit, Erstellen eines Modells auf der Grundlage der Problemanalyse."

Da in vielen allgemeinbildenden Schulen Informatik das einzige Technik- bzw. Ingenieursfach ist, ist die Softwareentwicklung das Thema, bei welchem das Wesen von Ingenieursberufen i. Allg. exemplarisch vermittelt werden kann. Bei den genannten bisherigen Ansätzen für Softwareentwicklung(sprojekte) im Unterricht spielt nicht zuletzt das für Ingenieurswissenschaften typische gestaltungsorientierte Moment nur eine eingeschränkte Rolle. Insbesondere fehlt den Schülerinnen und Schülern die Erfahrung, dass die Einführung von Software die jeweilige Domäne – z. B. ein Unternehmen bzw. dessen Organisationsstruktur oder Abläufe – in deutlicher Weise verändert.

In diesem Beitrag wird eine speziell für den Einsatz im Schulunterricht entworfene Softwareentwicklungsmethode präsentiert. Gemäß der eingesetzten Modellierungssprachen und Implementierung heißt die Methode *ProKlaMation – Prozesse, Klassen, Automation*. Die Methode unterstützt speziell dabei, (auch) die frühen Phasen der Softwareentwicklung im Unterricht angemessen zu adressieren. Die Methode orientiert sich an gängigen Vorgehensweisen der Informatik-Praxis. Sie nutzt Ansätze der (Geschäfts-) Prozessmodellierung.

Der Beitrag ist wie folgt aufgebaut: Zunächst werden Prozessmodellierungstechniken allgemein und die hier genutzte Business Process Model Notation (BPMN) grundlegend vorgestellt (Kap. 2). Danach folgen ein Überblick über die Methode (Kap. 3) sowie die Anwendung der Methode an einem Beispiel (Kap. 4). Didaktische Anmerkungen (Kap. 5) sowie ein Ausblick (Kap. 6) runden den Beitrag ab.

2 Konzeptuelle Prozessmodellierung

In der Wirtschaftsinformatik werden seit geraumer Zeit konzeptuelle Prozessmodellierungssprachen entwickelt. Auch viele Vertreter der Informatik beschäftigen sich mit Fragen der Prozessmodellierung (vgl. [SSF15], S. 15). Es handelt sich hierbei um grafische Modellierungssprachen, die darauf gerichtet sind, realweltliche Abläufe – z. B. Geschäftsprozesse eines Unternehmens – semi-formal abzubilden und zu visualisieren. Die Sprachen sind so konzipiert, dass sie (1) Ablaufstrukturen formalisieren und als Planungsinstrument der Software-Entwicklung genutzt werden können. Gleichzeitig (2) sind sie so einfach gehalten, dass sie für Nicht-Informatiker gut verständlich sind. Sprachen der konzeptuellen Modellierung können somit insbesondere als Mittel der Kommunikation zwischen Domänenexperten und Software-Entwicklern genutzt werden.

Die Sprachen dienen hierbei sowohl als Instrument, um existierende Abläufe zu modellieren und zu analysieren (Ist-Prozess), als auch, um Prozesse zu reorganisieren (Soll-Prozess). Besonderes Interesse gilt hierbei auch den Veränderungen in Prozessen, die

durch eine avisierte Computerunterstützung hervorgerufen werden. Diesbezüglich sind Prozessmodelle auch ein konkretes Mittel der Technologiefolgeneinschätzung.

Typische Notationselemente einer konzeptuellen Prozessmodellierungssprache sind Aktivitäten, ein Kontrollfluss samt Verzweigungen sowie Ereignisse, die den Prozess auslösen, beeinflussen, oder (Zwischen-) Ergebnisse symbolisieren (siehe Abb. 1).

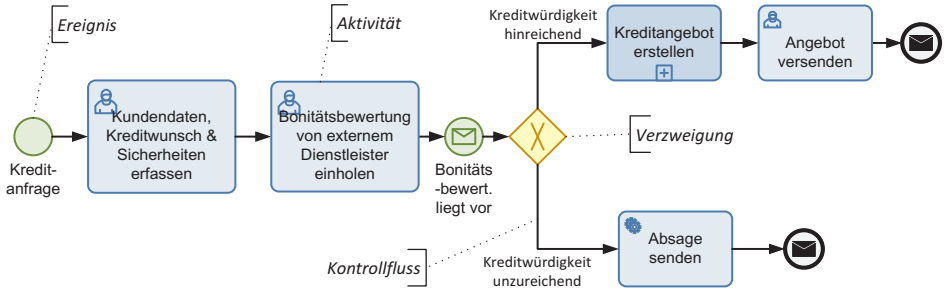


Abb. 1: Vereinfachtes Modell des Prozesses einer Kreditwürdigkeitsprüfung (in BPMN)

Die wesentlichen Strukturelemente ähneln den aus der Kerninformatik bekannten Programm-Ablauf-Plänen, weshalb sich Prozessmodellierungssprachen gut in den Unterricht integrieren lassen. Allerdings gibt es auch zentrale Unterschiede: Prozessmodellierungssprachen sind, wie bereits erwähnt, dediziert semi-formal und als Kommunikationsinstrument zwischen Entwicklern und Domänenexperten konzipiert. Daher beinhalten sie zusätzliche Elemente, die zur Abbildung realweltlicher Prozesse besonders hilfreich sind. Besonders augenfällig ist hierbei bspw. die Kennzeichnung einer Aktivität dahingehend, ob sie manuell ( ), semi-automatisch (Mensch interagiert mit Maschinen:) oder voll-automatisch abläuft ( ).

Ein deutlicher Unterschied zur softwarenahen Modellierung besteht zudem in der Art, wie die Sprachelemente typischerweise verwendet werden: Wenn z. B. ein Ablauf nicht eindeutig zu beschreiben ist, weil die handelnden Personen Freiheitsgrade haben, ist es üblich, unvollständig zu modellieren – z. B. über vage Aufgabenbeschreibungen. Auch bieten die Sprachen Notationselemente, die eine nur teilweise spezifizierte Modellierung unterstützen (z. B. unspezifische Verzweigungen). Sehr gängig in der Praxis ist es zudem, unklare oder nur aufwendig formalisierbare Tatbestände durch zusätzliche *textuelle Annotationen* natürlichsprachlich zu beschreiben.

Im Unterschied zu Klassendiagrammen (i. W. Spezifikation von Softwareschnittstellen) oder Zustandsdiagrammen (i W. Spezifikation gültiger Zustände eines Systems) sind konzeptuelle Prozessmodelle keine reduzierte, aber formale Sicht auf eine Software, sondern eine nur teilweise formalisierte Darstellung realweltlicher Prozesse. In der Software-Entwicklung sind Prozessmodelle daher Instrumente in frühen Phasen, nämlich der Analysephase und frühen Entwurfstätigkeiten (Prozess-Design).

Es gibt eine Reihe von konzeptuellen Prozessmodellierungssprachen, die alle für die hier vorgeschlagene Software-Entwicklungsmethode geeignet wären. Zu denken ist hier insbesondere an die *MEMO Process Modelling Language* (MEMO-PML) [Fr11] oder

die *Ereignisgesteuerten Prozessketten* [Sc00]. Aufgrund der Verfügbarkeit verschiedener, auch freier Modellierungseditoren wird in diesem Beitrag die *Business Process Modell Notation* (BPMN) [OM15] genutzt.

3 Die Methode ProKlaMation im Überblick

Die zentrale Herausforderung der Softwaretechnik ist – generalisierend gesprochen – die Komplexitätsreduktion. Das methodische Mittel zur Komplexitätsreduktion sind Modellierungssprachen der Software-, Projekt- oder Finanzplanung. Diesen ist gemein, dass sie die jeweilige Planung gezielt auf bestimmte Aspekte fokussieren. Die hier vorgestellte Methode ProKlaMation ist darauf gerichtet, Planungssprachen der Informatik so zu verknüpfen, dass die Phasen der Software-Entwicklung authentisch im Unterricht durchlaufen werden können, ohne diesen zu überfrachten.

Das Vorgehensmodell der Methode ProKlaMation orientiert sich grundsätzlich am Wasserfallmodell. Die Tätigkeiten und Ergebnisse der einzelnen Phasen sind hierbei klar voneinander unterscheidbar, bauen aber gleichzeitig aufeinander auf. ProKlaMation ist eine prozessorientierte Methode. Die Prozessmodellierung kommt insbesondere in den frühen Phasen der Softwareentwicklung zum Einsatz und wird (zusätzlich) mit den Modellen späterer Phasen verknüpft. Die Methode unterscheidet sich diesbezüglich in den Phasen Analyse, Prozess-Design und Software-Entwurf von den Vorgehensweisen in gängigen Schulbüchern (siehe oben), weswegen im Folgenden insbesondere auf diese Phasen eingegangen werden soll[3]:

- Gegenstand der Analyse-Phase ist es, einen oder mehrere Abläufe eines Anwendungsbereiches (einer Domäne) mithilfe einer Prozessmodellierungssprache zu modellieren (Ist-Prozessmodell) und dieses insbesondere im Hinblick auf eine Softwareunterstützung zu analysieren.
- In der *Prozess-Design-Phase* wird das Ist-Prozessmodell unter Berücksichtigung der Anforderungen aus der Domäne zu einem Soll-Prozessmodell „reorganisiert“. Dieses berücksichtigt organisationale Verbesserungen und die durch eine Softwareunterstützung induzierten Veränderungen im Prozessablauf.
- In der *Software-Entwurfsphase* werden aus dem Soll-Prozessmodell diejenigen Informationen abgeleitet, die innerhalb des Prozesses benötigt werden und damit von der zu erstellenden Software zu verwalten sind. Das Soll-Prozessmodell wird dazu durch ein UML-Klassendiagramm (Fachmodell) ergänzt. Dieses wird im nächsten Schritt angereichert mit Konzepten, welche für das Funktionieren der Software zuständig sind (z. B. Controller-Klassen oder GUI-Elemente), was zu einem Implementierungsmodell führt.

Abb. 2 zeigt das Phasenmodell und die jeweils genutzten Modelltypen.

[3] Für ein Unterrichtskonzept für die Test-Phase siehe [Sc13].

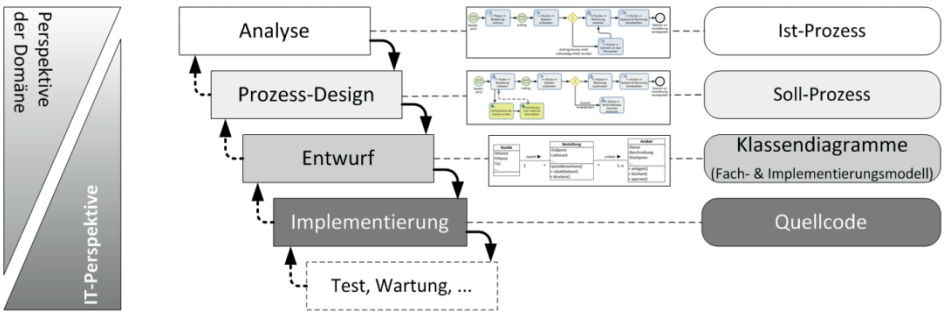


Abb. 2: Phasenmodell und jeweils genutzte Modelltypen

Das Zusammenspiel der Modelle erlaubt eine durchgängige Entwicklung und ein durchgängiges anschauliches Erleben des Softwareentwicklungsprozesses. Die weitestgehend intuitiv verständlichen Sprachelemente der Prozessmodellierung erlauben Schülern ohne besondere Voraussetzung einen schnellen Einstieg in die Softwareentwicklung.

4 Methode an einem Beispiel (Pizzadienst)

Die Anwendung und – damit einhergehend – mögliche schulische Aufgabenstellungen werden im Folgenden anhand eines Demonstrationsbeispiels gezeigt. Die Darstellung fokussiert hierbei auf die Phasen Analyse, Prozess-Design und Implementierung. Ausgangspunkt des Demonstrationsbeispiels sei das Fallbeispiel einer Pizzeria:

Die Pizzeria „Luigi" hat ihr Geschäftsfeld um einen Lieferservice erweitert. Luigi, der Inhaber, ist ein begnadeter Koch aber ein weniger guter Manager. Er organisiert den Lieferservice wie ein stationäres Restaurant: Die Speisen werden per Telefon am Tresen bestellt. Die Bestellannahme und die Lieferung werden von unterschiedlichen Personen übernommen. Rückfragen beim Gast gestalten sich daher schwierig und Liefertouren lassen sich schlecht im Vorfeld planen. Das Geschäftsergebnis erfüllt nicht die Erwartungen. Zudem häufen sich Beschwerden, weil Lieferzusagen nicht eingehalten werden.

4.1 Analyse (Ist -Prozess)

Zweck der Analysephase ist es, die Optionen einer Software-Unterstützung einer Domäne zu ergründen und Anforderungen abzuleiten. In der Analysephase würden Schüler ein Prozessmodell wie das in Abb. 3 zur derzeitigen „Bestellannahme und Speisenzubereitung" entweder selbst erstellen oder in einer Aufgabe als Ist-Prozess erhalten.

Aus dem Ist-Prozessmodell (mit der Fallbeschreibung) lassen sich Vorschläge und Ideen zur Reorganisation des Prozesses und Optionen einer Unterstützung durch Software ableiten. Die Schülerinnen und Schüler übernehmen dabei die Rolle eines externen und somit kritischen IT-Beraters oder Softwareentwicklers. Für dieses Fallbeispiel wäre bspw. festzustellen, dass (1) die Koordination zwischen Theke und Küche verbesse-

rungswürdig ist, insbesondere daraufhin, dass die Auslastung der Küche und ggf. nicht mehr vorhandenen Zutaten schon bei der Bestellung bekannt sein sollten, sowie (2) dass die vielen manuellen Tätigkeiten zu Fehlern und Medienbrüchen bei der Weitergabe der Bestellung und der Rechnungserstellung führen.

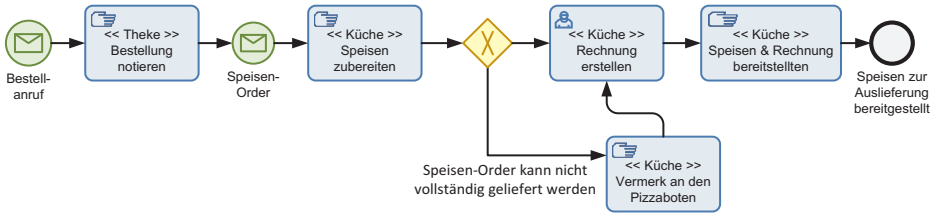


Abb. 3: Modell des Teilprozesses „Bestellannahme und Speisenzubereitung“ (BPMN)

4.2 Prozess-Design (Soll-Prozess)

Auf Basis der Problembeschreibung und Empfehlungen der Analyse-Phase ist ein reorganisierter Prozess zu modellieren (Soll-Prozess), der neben organisatorischen Verbesserungen insbesondere auch durch Software unterstützt werden kann. Abb. 4 zeigt einen möglichen Soll-Prozess unseres Pizzeria-Beispiels.

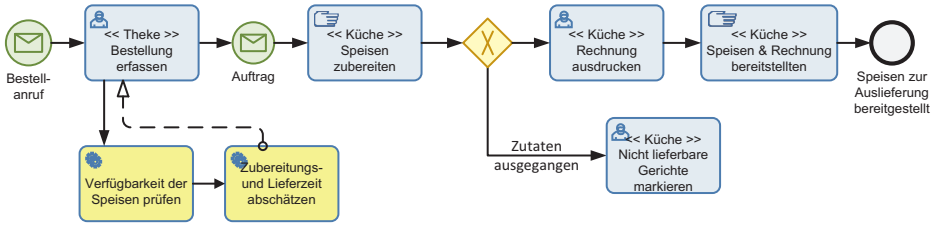


Abb. 4: „Reorganisierter“ Soll-Prozess (BPMN)

Im Unterricht ist es empfehlenswert, zu berücksichtigen, dass konzeptuelle Prozessmodellierungssprachen große Freiheit in der Modellierung erlauben (siehe oben). Es ist daher sinnvoll, alternative Entwürfe eines Soll-Prozesses zu vergleichen und sich im Diskurs auf eine Alternative zu einigen.

4.3 Software-Entwurf (Fachmodell, Implementierungsmodell)

Aus den Einzelschritten eines Prozessmodells lässt sich gut der jeweilige Informationsbedarf ableiten. Aus dem Soll-Prozessmodell sind die im Prozess durch ein Informationssystem zu verwaltenden Daten abzuleiten. Diese sind in einem UML-Klassendiagramm zu spezifizieren (*Fachmodell*). Im genannten Fallbeispiel ist ein Klassendiagramm für ein Informationssystem zu entwerfen, welches das Angebot und die Bestellungen des Pizzadienstes abbildet. Abb. 5 zeigt ein mögliches Fachmodell für den Pizzaservice. Die gestrichelten Kanten deuten hierbei an, in welcher Aktivität welche Daten

erhoben bzw. besonders benötigt werden.

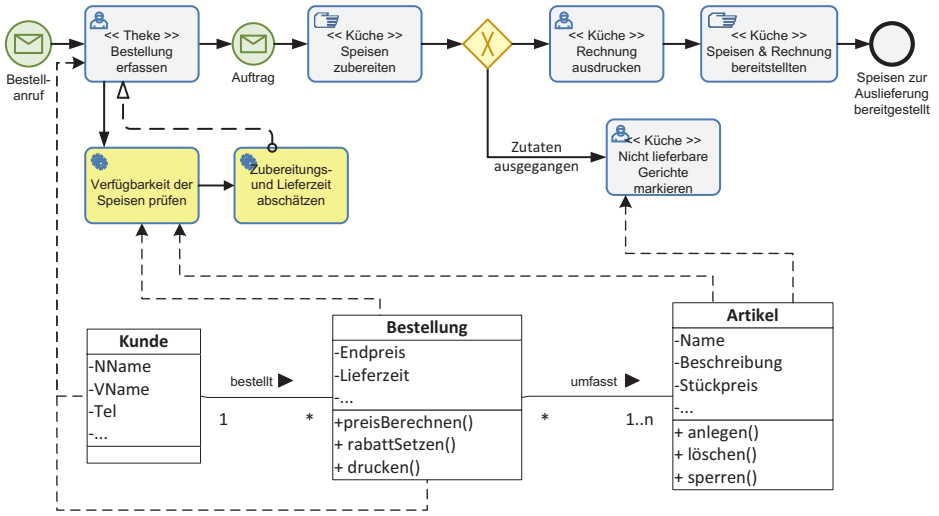


Abb. 5: Um ein Fachmodell (Klassendiagramm) erweitertes Prozessmodell

Fachmodelle sind nicht hinreichend für die Implementierung einer Software. Nun gilt es, technische Details der Implementierung zu bestimmen. Da im schulischen Kontext für gewöhnlich weder über die genutzten Technologien oder Programmiersprachen noch über die Software-Architektur zu bestimmen ist, gilt es, das Fachmodell mit zusätzlichen Software-Artefakten anzureichern, die für den Ablauf der Software von Nöten sind. Hier ist bspw. zu denken an Controller-Klassen, die den Prozessablauf steuern, oder die GUI-Komponenten einer grafischen Oberfläche (*Implementierungsmodell*).

5 Didaktische Einordnung

Die verschiedenen Aufgaben, Tätigkeiten und Ergebnisse der Methode ProKlaMation im Unterricht sind in Tab. 1 aufgelistet. Die präsentierte Methode muss dabei nicht ausschließlich an durchgängigen Beispielen behandelt werden, sondern es lassen sich in jeder Phase einzelne Aufgaben stellen – bspw. zur Modellierung von Fallbeispielen, zur Analyse gegebener Prozesse, zur Ableitung eines Fachmodells aus einer Kombination aus Prozessmodell und Fallbeispiel.

Dadurch, dass die Methode kein durchgängiges Beispiel erfordert, lassen sich Unterrichtsreihen gut phasieren. Ein zeitliches Auseinanderdriften von Schüler-Arbeitsgruppen ist nach Erfahrung des Autors diesbezüglich stets unproblematisch.

Im Folgenden werden Vor- und Nachteile bzw. Herausforderungen der Methode noch einmal eingeordnet. Diese beruhen auf Unterrichtserfahrungen des Autors einerseits sowie auf Erfahrungen und Feedback aus Lehrerfortbildungen, in denen Teile der Methode vorgestellt worden sind.

Phase	Typische Aufgaben / Tätigkeiten im Unterricht	Meilenstein / Ergebnis
Analyse	• Prozessmodelle zu Fallbeispielen erstellen • Prozessmodelle vergleichen	Ist-Prozessmodell
Prozess-design	• Analyse des Ist-Prozessmodells (und ggf. der Fallbeschreibung) hinsichtlich Reorganisation und Softwareunterstützung. • Redesign des Ist-Prozessmodells mit Blick auf dessen Unterstützung durch Software • Prozessmodelle vergleichen • Technologiefolgen an konkreten Beispielen diskutieren.	Soll-Prozessmodell (ggf. mehrere Alternativen)
Software-Entwurf	• Ableitung der Daten, die im Soll-Prozess benötigt werden, und Entwurf eines UML-Klassendiagramms • Verfeinerung des Klassendiagramms um Konzepte, die für die Software von Bedeutung sind. • Modelle vergleichen	Fachmodell Implementierungsmodell

Tab. 1: Didaktische Aufgaben, Tätigkeiten und Meilensteine der Methode ProKlaMation.

5.1 Unterrichtspraktische Anmerkungen

Zur Auswahl von Fallbeispielen:

- Die Arbeit an realistischen Fallbeispielen ist geeignet, die Software-Entwicklungs-Reihe in einen für die Schüler ansprechenden Kontext zu setzen. Allerdings ist an die Auswahl der Fallbeispiele bzw. Kontexte immer die Voraussetzung zu knüpfen, dass sie erstens mit Prozessmodellierungssprachen angemessen zu beschreiben sind, und zweitens, dass es sich um realweltliche Prozesse handelt, die durch Software unterstützt werden können. Dies gilt i. d. R. für Geschäfts-, Produktions- und Verwaltungsprozesse sowie für die Beschreibung komplexerer Software-Anwendungsszenarien (Use Cases).
- Empfehlenswert ist es, bei einem durchgehenden Fallbeispiel Musterlösungen für Meilensteine vorzuhalten, auf welche Schüler aufbauen können, die selbst keine geeigneten Zwischenergebnisse erzielen konnten. Im Unterricht des Autors hat es sich bewährt und als abwechslungsreich herausgestellt, wenn spezifische Aufgaben zu jeder Phase mit ein oder zwei durchgängigen Fallbeispielen gemischt werden.

Zum Umgang mit Prozessmodellierungssprachen:

- Prozessmodellierungstechniken sind für Schüler leicht verständlich, so dass sie grundsätzlich auch erst während der Reihe Softwareentwicklung eingeführt werden können. Es empfiehlt sich aber, die Techniken schon vorher zu nutzen. Insbesondere kann die Sprache BPMN Programm-Ablauf-Pläne ersetzen und so schon während der Algorithmik/Programmierung im Unterricht eingesetzt werden.

- In der Unterrichtspraxis erstellen Schüler häufig unterschiedliche Modelle, die auch unterschiedliche Softwarelösungen nahelegen. Hier kommt der Aufgabe, alternative Schülerlösungen im Diskurs zu vergleichen, eine wichtige Rolle zu.
- Da bei der Nutzung semi-formaler Modellierungssprachen das Abstraktionsniveau nicht strikt vorgegeben ist, ist es hilfreich, Hinweise zu geben, wie detailliert ein Modell zu erstellen ist. Eine Faustregel lautet: So detailliert modellieren, wie dies ein Domänenexperte (hier: der Pizzabäcker) für angemessen halten würde.
- Zur Modellierung von Prozessen sind verschiedene Tools erhältlich – auch kostenlose Werkzeuge (z. B. yEd oder bpmn.io). Leider unterstützen die kostenlosen Werkzeuge aktuell keine Simulation.

5.2 Didaktische Bewertung

Lernsystematisch besehen ist die Software-Entwicklung eine Vertiefung der Algorithmik, der Programmierung und der Modellierung. Schüler mit Schwächen in der Programmierung sind bei programmierintensiven Unterrichtsansätzen der Softwaretechnik schnell überfordert. Prozessmodellierungssprachen allerdings sind für Schüler gut verständlich. Sie erlauben es auch den in der Programmierung schwachen Schülern, sich am Unterricht erfolgreich zu beteiligen.

Die vorgeschlagene Methode ist eine didaktische Reduktion von in der Informatik-Praxis gängigen Ansätzen. Die Methode steht dabei authentisch für die Informatik und ist gleichzeitig ein prototypisches Beispiel für die Software-Entwicklung als solches. Durch die Nutzung verschiedener Modellierungs- bzw. Programmiersprachen in den verschiedenen Phasen der Softwareentwicklung (Prozessmodelle in der Analyse- und Prozess-Design-Phase, Klassendiagramme im Entwurf, eine Programmiersprache in der Implementierung), bleiben die Phasen und deren Herausforderungen für die Schüler klar wahrnehmbar. Gleichzeitig knüpfen die Meilenstein-Ergebnisse der einzelnen Phasen aneinander an, so dass die Durchgängigkeit der Entwicklung erkennbar wird.

Die Einführung und Nutzung einer Software in einem Anwendungskontext verändert diesen vielfach in deutlicher Weise. Abläufe und Organisationsstrukturen sind hiervon betroffen. Die Auswirkungen eines Informationssystems auf dessen Kontext müssen bei dessen Planung berücksichtigt werden. Nach den Erfahrungen des Autors erhalten Schülerinnen und Schüler über die Anwendung der Methode ein gutes Gespür für die Herausforderungen der Analysephase und für die Auswirkungen von Software (allgemeiner Technologien) auf Anwendungsfälle.

6 Zusammenfassung und Ausblick

Die Praxis der Softwaretechnik stellt nicht die Programmierung in den Mittelpunkt. Die Kernherausforderungen sind fachlicher und organisationaler Natur. Fachlich gilt es, schrittweise einen Software-Entwurf zu erstellen (Datenmodell, Klassendiagramm, Spezifikation). Dies geschieht auf Grundlage einer methodischen Analyse des Gegenstands-

bereiches, der sog. „Domäne“ einer Software. Größere Softwareprojekte umfassen viele Beteiligte mit unterschiedlichen Interessen und fachlichen Hintergründen. Organisational besteht die Herausforderung im Projektmanagement, um Einigungsprozesse zwischen den Interessengruppen herzustellen und eine klare Aufgabenverteilung zu ermöglichen. Für beide Herausforderungen stellt die Softwaretechnik Modellierungssprachen bereit, die aus verschiedenen Perspektiven die Software oder deren Domäne beschreiben.

Die vorgestellte Methode der Softwareentwicklung im Unterricht greift die zentrale Rolle der Prozessmodellierung in der betrieblichen Praxis auf. Modellierungskonzepte werden phasenübergreifend und durchgängig angewendet. Die Herausforderungen der Phasen der Softwareentwicklung werden erlebbar und die Prozessmodelle werden im Unterricht selbst als Kommunikationsmedium zum Vergleich von geänderten und durch neue Software transformierte Abläufe („mögliche Welten“) genutzt.

Literaturverzeichnis

[Br09] Brichzin, P. et.al.: Informatik Oberstufe 1 – Datenstrukturen und Softwareentwicklung. Oldenbourg: München et al. 2009.

[Du09] Duden-Verlag (Hg.): Informatik – Objektorientierte Programmierung mit BlueJ. Duden: Berlin, Mannheim, 2009.

[En06] Engelmann, L. (Hg.): Informatik – Gymnasiale Oberstufe. Duden: Berlin 2006.

[Fi06] Fischer, H. et.al.: Grundlagen der Informatik II. Oldenbourg: München et al. 2006.

[Fr11] Frank, U.: MEMO Organisation Modelling Language (2): Focus on Business Processes. ICB-Research Report Nr. 49, Institut für Informatik und Wirtschaftsinformatik, Universität Duisburg-Essen, 2011. (www.icb.uni-due.de/ fileadmin/ICB/research/research_reports/ICB-Report-No49.pdf; 31.01.2015)

[Ha10] Hattenhauer, R.: Informatik für Schule und Ausbildung. Pearson: München et al. 2010.

[Hu09] Hubwieser, P. et.al.: Informatik 4 – Lehrwerk für Gymnasien – Rekursive Datenstrukturen, Softwaretechnik. Ernst Klett: Stuttgart 2009.

[KM04] Kultusministerkonferenz (Hg.): Einheitliche Prüfungsanforderungen Informatik. Beschluss vom 01.12.1989 i. d. F. von 05.02.2004. (www.kmk.org/fileadmin/ veroeffentlichungen_beschluesse/1989/1989_12_01-EPA-Informatik.pdf; 31.1.2015)

[OM15] Object Management Group (Hg.): Business Process Model and Notation. (www.bpmn.org; 31.01.2015)

[Sc00] Scheer, A.-W.: ARIS: Business Process Modeling. 3. Aufl., Springer: Berlin et al., 2000.

[Sc13] Schauer, H.: Debugging-Aufgaben – Programmfehler als Lernchance. In: Breier, N. et.al. (Hg.): INFOS 2013: Informatik erweitert Horizonte – 15. GI-Fachtagung Informatik und Schule. Lecture Notes in Informatics Nr. P-219, Gesellschaft für Informatik, Bonn, 2013, S. 97 – 106.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 267

Programmierunterricht für Kinder und deren Lehrpersonen: Unterrichtsmaterialien, didaktische Herausforderungen und konkrete Erfahrungen

Giovanni Serafini[1]

Abstract: Der Workshop befasst sich mit einem etablierten Ansatz, um den Programmierunterricht für Kinder bis 12 Jahre altersgemäß im normalen Klassenunterricht durchzuführen. Anhand konkreter Unterrichtsbeispiele und entsprechender Lernmaterialien wird über allgemein bildende Inhalte der Informatik, Herausforderungen, Erfolgsfaktoren, Genderproblematik, Weiterbildung der Klassenlehrpersonen sowie über persönliche Erfahrungen von Referenten und Workshop-Teilnehmenden reflektiert.

Keywords: Informatik und allgemeine Bildung, Intellektuelle Arbeit, Programmiersprache, Programmierunterricht, Primarschule, Logo, Kinder, Lehrpersonen, Schweiz, Lehrplan 21.

1 Informatik und allgemeine Bildung

Die Informatik ist die Wissenschaft, die sich mit der systematischen und der automatisierten (d.h. mit der **algorithmischen**) Verarbeitung von Information, mit der Informationsspeicherung, mit deren Verwaltung und Übertragung befasst [GH12]. So wie ein Maschineningenieur physikalisch fassbare Maschinen entwickelt, wie beispielsweise einen Webstuhl, der mechanische Abläufe in der Textilindustrie automatisiert, baut auch ein Informatiker Maschinen - nur sind diese abstrakt und nicht fassbar. Die Maschinen eines Informatikers dienen der Automatisierung von intellektueller Arbeit und werden mit einem eigenen Begriff beschrieben, nämlich mit der Bezeichnung **Programm**.

Die Fähigkeit, intellektuelle Arbeit zu automatisieren (und zwar **vollständig** zu automatisieren) unterscheidet den Informatiker vom Maschineningenieur, vom Mathematiker oder vom Naturwissenschaftler. Im Informatikunterricht geht es darum, diese Fähigkeit, die dazugehörige Denkweise bzw. die damit verbundene Art und Weise, Probleme anzupacken und zu lösen, zu vermitteln. In einem vielzitierten Aufsatz beschreibt Wing diese Denkweise und den Beitrag der Informatik zur allgemeinen Bildung mit dem Konzept des **Computational Thinking** [Wi06], das sich auf Deutsch sinngemäß als **Algorithmisches Denken** anführen lässt.

Die Erziehung hin zum algorithmischen Denken ist das wichtigste übergeordnete Lernziel der Informatik. **Algorithmisch zu denken** lernt man, indem man für ein gegebenes Problem eine **automatisch auszuführende Lösungsanleitung konstruiert**, kritisch hinterfragt und präzise beschreibt, also nicht zuletzt, indem man diese **programmiert**.

[1] ETH Zürich, Departement Informatik, Universitätstr. 6, CH-8092 Zürich, giovanni.serafini@inf.ethz.ch

Eine gemeinsame Arbeitsgruppe von Informatics Europe und ACM Europe hebt im Bericht „Informatics education: Europe cannot afford to miss the boat“[Ga13] die Informatik als die Disziplin hervor, welche das wissenschaftliche Fundament für die heutige, moderne, digitale Gesellschaft zur Verfügung stellt. Die Informatik umfasst die wesentlichsten konzeptionellen Grundlagen der Informationstechnologie. Algorithmisches Denken gehört im 21. Jahrhundert zur allgemeinen Bildung und soll auf allen Schulstufen gefördert werden.

2 Der Lehrplan 21 für die Volksschule der Deutschschweiz

2.1 Der neue Bildungsraum Schweiz

Die föderalistische Natur der Schweizerischen Eidgenossenschaft legt die Hoheit über das Bildungssystem grundsätzlich in die Hauptverantwortung der einzelnen Kantone. In einer wegweisenden Volksabstimmung hat die Schweiz am 21. Mai 2006 eine Anpassung der Bundesverfassung angenommen, welche die Kantonen und den Bund dazu verpflichtet, die wichtigsten Elemente im Bildungsbereich schweizweit zu harmonisieren [Sc05]. Angestrebt wird ein Bildungssystem im sogenannten Bildungsraum Schweiz, das die Mobilität der Familien mit schulpflichtigen Kindern in der flächenmäßig kleinen Schweiz nicht unnötigerweise erschweren soll [Sc06].

Die 21 Schweizer deutsch- und mehrsprachigen Kantone haben zwischen 2010 und 2014 den gemeinsamen, kompetenzorientierten **Lehrplan 21** für die ersten elf Jahren der schulischen Grundausbildung [Dea] erarbeitet. Diese umfasst den Kindergarten, die Primarschule sowie den obligatorischen Teil der Sekundarschule. Diese drei Schulstufen werden im Kontext des Lehrplans 21 mit dem Begriff der **Volksschule** zusammengefasst.

Der Lehrplan 21 ist in drei Zyklen gegliedert:

- Der 1. Zyklus dauert 4 Jahre und umfasst den Kindergarten sowie die zwei ersten Jahre der Primarschule.
- Der 2. Zyklus erstreckt sich vom 3. bis zum 6. Jahr der Primarschule.
- Der 3. Zyklus umfasst die ersten drei Jahre der Sekundarstufe.

Die Entwicklung des Lehrplans 21 wurde von der Deutschschweizer Erziehungsdirektoren-Konferenz (D-EDK) koordiniert. Diese hat den Lehrplan 21 am 7. November 2014 freigegeben. Die druckfertige Fassung liegt seit dem 2. April 2015 vor und dient als Vorlage für die zu entwickelnden kantonalen Lehrpläne [De15].

2.2 Lehrplan 21 und Informatik

Im Rahmen der Erarbeitung des Lehrplans 21 hat die D-EDK eine Arbeitsgruppe mit der Konzeption eines Lehrplans für Informatik, Medien und Anwendungskompetenzen beauftragt. Für Medien und Informatik wurde ein gemeinsames Modul [Deb] vorgeschlagen

und entwickelt. Die Anwendungskompetenzen fließen als überfachliche Kompetenzen in die Lehrpläne aller anderen Fächer ein.

Die übergeordneten Kompetenzen, welche die Schülerinnen und Schüler in der Informatik erwerben sollen, werden von der Arbeitsgruppe wie folgt formuliert:

1. Die Schülerinnen und Schüler können Daten aus ihrer Umwelt darstellen, strukturieren und auswerten.
2. Die Schülerinnen und Schüler können einfache Problemstellungen analysieren, mögliche Lösungsverfahren beschreiben und in Programme umsetzen.
3. Die Schülerinnen und Schüler verstehen Aufbau und Funktionsweise von informationsverarbeitenden Systemen und können Konzepte der sicheren Datenverarbeitung anwenden.

Im diesem Aufsatz und im entsprechenden Workshop befassen wir uns mit der zweiten der oben angeführten Kompetenzen. Diese wurde maßgeblich von der langjährigen Erfahrung des ETH Zürich im Rahmen sogenannter Logo-Schulprojekte geprägt.

3 Einführung in die Programmierung mit Logo an Schweizer Primarschulen

Das Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich (ABZ) führt seit mittlerweile fast zehn Jahren Kinder der Primarschule und deren Lehrpersonen in die Programmierung mit Logo ein. Die Kurse finden im Rahmen sogenannter **Logo-Schulprojekte** direkt an den jeweiligen Primarschulen statt. Sie umfassen eine theoretische Weiterbildung der Klassenlehrpersonen sowie 20 Lektionen mit der Klasse, welche auf 5 bis 10 Wochen verteilt werden.

Bisher haben mehr als 2000 Kinder, hauptsächlich in der Deutschschweiz, an einem solchen Schulprojekt teilnehmen dürfen. Die eingesetzten Unterrichtsmaterialien [Ge14] liegen in den Sprachen Deutsch, Französisch, Italienisch, Englisch, Spanisch, Slowakisch und Serbisch vor. Sie entsprechen dem Inhalt der ersten sieben Kapitel des deutschsprachigen Lehrbuchs **Einführung in die Programmierung mit Logo** [Hr14].

Seit Anfang 2014 werden die Logo-Schulprojekte als Kooperation des ABZ, des Schweizerischen Vereins für Informatik in der Ausbildung und der Hasler Stiftung durchgeführt. Diese neue Projektorganisation trägt den Namen **PrimaLogo** und soll die weitere Verbreitung der Schulprojekte ermöglichen. Regional verankerte Anbieter von PrimaLogo-Schulprojekten sind derzeit die Pädagogische Hochschule Graubünden, die Pädagogische Hochschule Luzern, die Universität Bern sowie die Universität Basel. PrimaLogo-Schulprojekte haben bereits in der französisch- sowie in der italienisch-sprechenden Schweiz stattgefunden.

4 Inhalte und Ziele des Workshops

Im Rahmen des Workshops wird ein etablierter Zugang zum Programmierunterricht in Logo vorgestellt [Se11]. Die Programmiersprache Logo wurde am Ende der 60er Jahre des letzten Jahrhunderts am Massachusetts Institute of Technology vom Erziehungswissenschaftler, Mathematiker und Informatiker Seymour Papert und seinem Team entwickelt. In Logo geht es darum, eine Schildkröte auf dem Bildschirm zu bewegen und geometrische Muster zu zeichnen. Dabei lernen die Kinder auf sehr effektive Art und Weise die Grundlagen der Programmierung.

Die primäre Zielsetzung des Unterrichts besteht darin, den Kindern eine **Sprache** beizubringen, mit welcher sie die Schildkröte auf den Bildschirm steuern können. Diese Sprache, Logo, ist die **Muttersprache der Schildkröte** und besteht zunächst aus sehr wenigen vorgegebenen Wörtern. Die Kindern lernen, wie sie schrittweise das **Wortschatz** der Schildkröte erweitern können, indem sie ihr neue Wörter und deren Bedeutung beibringen, welche von ihnen in Form neuer Programme formuliert wird. Die Kinder erhalten demnach keine vordefinierte und umfangreiche Programmiersprache, sondern werden dazu befähigt, die Sprache selber zu erweitern und diese so zu gestalten, dass die Aufgaben, die ihnen gestellt werden, möglichst effektiv und klar strukturiert gelöst werden können.

Der oben beschriebene Ansatz stützt sich auf die Erkenntnisse der Cognitive Load Theory [Sw11] und fokussiert demnach darauf, die beschränkte Kapazität des Arbeitsgedächtnisses der Schülerinnen und Schüler nicht unnötigerweise mit einer Fülle an vorgegebenen Befehlen und Kontrollstrukturen zu überlasten. Die Vermeidung einer zu hohen extrinsischen kognitiven Belastung wird von der einfachen grafischen Gestaltung der eingesetzten Programmierumgebungen xLogo [LC] und deren Weiterentwicklung XLogo4Schools [Zi] unterstützt.

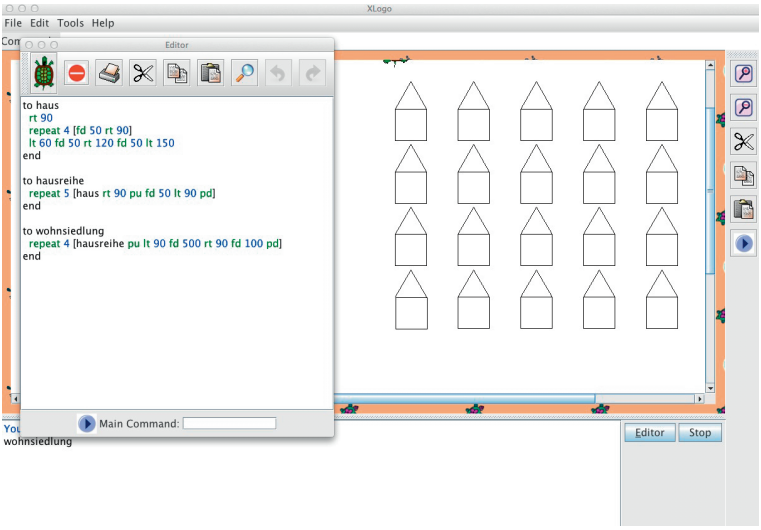


Abb. 1: Screenshot der Programmierumgebung xLogo

Die Lernmaterialien haben den folgenden didaktischen Aufbau:

- Lektion 1: Logo-Grundbefehle
- Lektion 2: Wiederholungen und der Befehl repeat
- Lektion 3: Modularer Entwurf
- Lektion 4: Vielecke, Kreise und Farben
- Lektion 5: Parameter
- Lektion 6: Parameter und Unterprogramme
- Lektion 7: Animationen

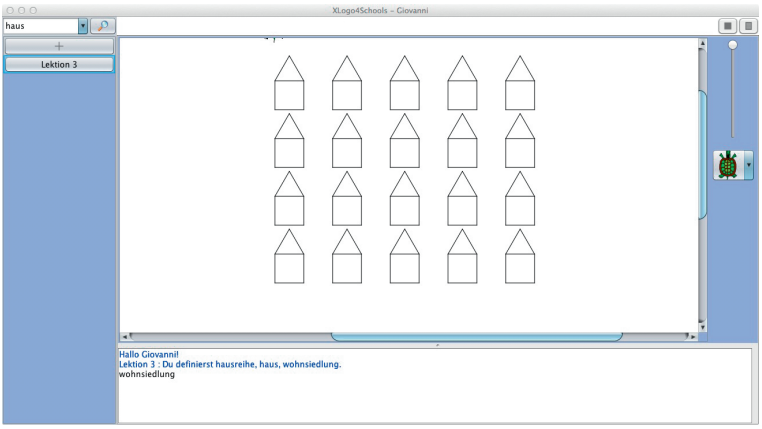


Abb. 2: Screenshot der Programmierumgebung XLogo4Schools

Nach einer kurzen Einführung durch die Referenten erhalten die Teilnehmenden die Gelegenheit, Teile der Unterrichtsmaterialien selbständig zu erproben. Im Dialog mit den Referenten reflektieren die Teilnehmenden über allgemein bildende Inhalte der Informatik, Herausforderungen, Erfolgsfaktoren und die Genderproblematik. Die Rolle der üblicherweise informatikfernen Klassenlehrperson sowie die Optionen für eine entsprechende fachliche sowie informatik didaktische Weiterbildung werden thematisiert.

Der Workshop wird mit Hinweisen zur aktuellen bildungspolitischen Entwicklung in der Deutschschweiz ergänzt. Informatik soll im Rahmen des Lehrplans 21 [Dea] als Unterrichtsubjekt in der Volksschule eingeführt werden.

Literaturverzeichnis

[Dea] Deutschschweizer Erziehungsdirektoren-Konferenz: , Lehrplan 21. `http://www.lehrplan.ch/`. Zuletzt besucht am 29. April 2015.

[Deb] Deutschschweizer Erziehungsdirektoren-Konferenz: , Medien und Informatik. `http://vorlage.lehrplan.ch/downloads.php`. Zuletzt besucht am 30. April 2015.

[De15] Deutschschweizer Erziehungsdirektoren-Konferenz: , Lehrplan 21: Druckfertige Fassung liegt vor. http://www.lehrplan.ch/medienmitteilung/lehrplan-21-druckfertige-fassung-liegt-vor, 2015. Zuletzt besucht am 29. April 2015.

[Ga13] Gander, Walter; Petit, Antoine et al.: , Informatics education: Europe cannot afford to miss the boat. http://europe.acm.org/iereport/ie.html, 2013. Zuletzt besucht am 29. April 2015.

[Ge14] Gebauer, Heidi; Hromkovič, Juraj; Keller, Lucia; Kosirova, Ivana; Serafini, Giovanni; Steffen, Björn: , Programmieren mit LOGO. http://www.abz.inf.ethz.ch/primarschulen-stufe-sek-1/unterrichtsmaterialien/, 2014. Zuletzt besucht am 30. April 2015.

[GH12] Gander, Walter; Hromkovič, Juraj: Definition von Informatik. Unveröffentlichte Unterrichtsmaterialien zur Didaktik der Informatik., 2012.

[Hr14] Hromkovič, Juraj: Einführung in die Programmierung mit LOGO - Lehrbuch für Unterricht und Selbststudium (3. Aufl.). Springer Vieweg, 2014.

[LC] Le Coq, Loïc: , xLogo. http://xlogo.tuxfamily.org/. Zuletzt besucht am 30. April 2015.

[Sc05] Schweizerische Bundeskanzlei: , Bundesbeschluss vom 16.12.2005 über die Neuordnung der Verfassungsbestimmungen zur Bildung. https://www.admin.ch/ch/d/pore/va/20060521/det522.html, 2005. Zuletzt besucht am 29. April 2015.

[Sc06] Schweizerische Bundeskanzlei: , Volksabsttimung vom 21. Mai 2006: Erläuterungen des Bundesrats. http://www.bk.admin.ch/themen/pore/va/20060521/index.html?lang=de, 2006. Zuletzt besucht am 29. April 2015.

[Se11] Serafini, Giovanni: Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects. In (Kalas, Ivan; Mittermeir, Roland, Hrsg.): Informatics in Schools. Contributing to 21st Century Education, Jgg. 7013 in Lecture Notes in Computer Science, S. 143–154. Springer Berlin Heidelberg, 2011.

[Sw11] Sweller, John: Cognitive Load Theory. Jgg. 55 in Psychology of Learning and Motivation, S. 37 – 76. Academic Press, 2011.

[Wi06] Wing, Jeannette M.: Computational Thinking. Commun. ACM, 49(3):33–35, 2006.

[Zi] Zivković, Marko: , XLogo4School. http://sourceforge.net/projects/xlogo4schools/. Zuletzt besucht am 30. April 2015.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 273

Mobiles Programmieren mit Android und Python im Informatikunterricht

Daniel Spittank[1]

Abstract: Informatik durchdringt zunehmend den Alltag in modernen Gesellschaften, besonders die Miniaturisierungs- und Mobilisierungsprozesse begünstigen dies. Besonders der Boom mobiler Informatiksysteme[2] und die Verbreitung mobiler Internetzugänge begründen gesellschaftliche Veränderungen.

Verschiedene Studien, allen voran die JIM-Studien (Jugendliche, Information, Multimedia), aber auch die des Deutschen Instituts für Vertrauen und Sicherheit im Internet (DIVSI), belegen eindrücklich die immens zunehmende Bedeutung mobiler Informatiksysteme und des immer verfügbaren Zugangs zum Internet für Schülerinnen und Schüler. Gleichsam lässt sich erkennen, dass die Bedeutung stationärer Informatiksysteme für Schülerinnen und Schüler schwindet.

Diesen Entwicklungen wird derzeit von Schulen allerdings noch wenig Aufmerksamkeit geschenkt. Einzig die Einführung von wenigen Tablet-Klassen kann hier genannt werden. Ansonsten dominieren Verbote mobiler Informatiksysteme. Die gesellschaftlichen Entwicklungen werden (noch) aus der Schule ausgeschlossen. Zu groß ist die Sorge vor den möglichen negativen Auswirkungen, die mit den Geräten in Verbindung gebracht werden (z.B. Ablenkung vom Unterrichtsgeschehen und Cybermobbing). Besonders im Informatikunterricht dominiert die, durch feste Computerräume vorgegebene, Arbeit an stationären Systemen. Dabei spricht viel für einen offeneren Umgang mit den mobilen Informatiksystemen, besonders der direkte Bezug zum Alltag der Schülerinnen und Schüler kann sich positiv auf die Motivation und die Begeisterung auswirken. Kann doch ein wesentlicher Teil des Alltags begreifbar gemacht und somit auch ein erkennbarer Vorteil für das tägliche Leben erlangt werden.

Im Rahmen seiner Arbeit zum ersten Staatsexamen entwickelte der Autor die Grundzüge für ein Unterrichtskonzept, das den Entwicklungen Rechnung trägt und versucht, die Vorteile mobiler Informatiksysteme für den Informatikunterricht nutzbar zu machen. Dazu gehören etwa die Unabhängigkeit von Computerräumen und die vielfältigeren Ansatzpunkte für den Einsatz kooperativer Methoden, die helfen können, die fachbezogene Kommunikation zu fördern. Es basiert dabei auf den positiven Erfahrungen vorangegangener Versuche mit Symbian-Smartphones[3] und konnte im letzten Jahr im Rahmen seines Referendariats mit zwei achten Klassen erstmalig erprobt werden. Dieser Beitrag soll das Konzept und die gewonnenen Erkenntnisse vorstellen.

1 Gesellschaftliche Bedeutung mobiler Informatiksysteme

Die flächendeckende Verfügbarkeit mobiler Informatiksysteme – in Form von Tablets, Smartphones und zahllosen smarten Gadgets – verändert die moderne Gesellschaft. Die

[1] Gesamtschule Uellendahl-Katernberg, Wuppertal, mobile@daniel.spittank.net

[2] Analog zu [Sp12] sind hier keine Notebooks oder Netbooks gemeint, sondern hauptsächlich Smartphones und Tablets.

[3] vgl.[He09]

ständige Verfügbarkeit von Information und die Verknüpfung mit den Daten, welche ständig durch die mobilen Systeme erfasst werden, ist einerseits ein Segen für viele Menschen, die sich schon immer einen persönlichen Assistenten herbeigesehnt haben. Andererseits erscheint dies, auf den Schutz ihrer persönlichen Daten bedachten, Charakteren wohl eher als Fluch. Insbesondere nach den diversen Datenschutz- und Spionage-Skandalen der letzten Jahre.

Diese gesellschaftlich bedeutende Debatte wird auch immer mehr öffentlich ausgetragen. Spätestens seit den Enthüllungen der digitalen Spionagetätigkeiten der NSA durch Edward Snowden ist einer breiteren Öffentlichkeit bewusst, dass die Informatisierung der Gesellschaft zwei Seiten hat.

Dennoch: Die vielfältigen nützlichen Funktionen – insbesondere in Verbindung mit Clouddiensten – erleichtern oftmals den Alltag. Reines Faktenwissen verliert durch die stetige Verfügbarkeit diverser Informationsquellen an Bedeutung, wohingegen der Bedarf an informatischer Vernunft[4] zunimmt. Mögliche Risiken können oft nicht einfach beurteilt werden, da die parallel erfolgende Simplifizierung der Benutzungsschnittstellen vieles vor dem Anwender verbirgt.

So verbleiben die vielen Funktionen und neuen Möglichkeiten, die die mobilen Geräte so schlau erscheinen lassen, für die meisten Anwender im nebulösen Feld der quasimagischen Blackbox. Dass dabei viele sehr persönliche Daten – oftmals ungeschützt – durch die halbe Welt gesendet werden, ist vielen nicht einmal klar. Genannt seien hier exemplarisch Spracherkennungen, die auf den Servern der Anbieter erfolgen, und Kommunikationsapps, die das gesamte Adressbuch in die Cloud laden, damit festgestellt werden kann, mit welchen Kommunikationspartnern man mittels der App kommunizieren kann. Im Zuge der Mobilisierungstendenzen und des Bedeutungszuwachses von Clouddiensten müssen allerdings auch Nutzungsszenarien neu bewertet werden, von denen man bisher an stationären Systemen keinerlei Gefährdung zu erwarten hatte. So etwa die Bildbearbeitung, die auf mobilen Informatiksystemen sehr beliebt ist, im Gegensatz zu stationären Bildbearbeitungsanwendungen aber oft nicht lokal, sondern in der Cloud erfolgt. Der bei mobilen Informatiksystemen deutlich stärker ausgeprägte Blackbox-Effekt sorgt hier also für eine steigende Intransparenz gegenüber den Anwendern der Geräte.

Es ergibt sich die Notwendigkeit, Menschen in die Lage zu versetzen, moderne Informatiksysteme selbstbestimmt und verantwortungsbewusst zu verwenden und die mit ihnen verbundenen Risiken und Nebenwirkungen einschätzen zu können, ohne zunächst den Informatiker ihres Vertrauens zu befragen, will man die mündige Teilhabe in demokratischen Gesellschaften sicherstellen.

2 Aktueller Stand in der Schule

Die Befähigung zur mündigen Teilhabe ist natürlich die originäre Aufgabe der Schule im Allgemeinen und des Informatikunterrichts im Speziellen, wenn es um informatische

[4] vgl. [GH05, S. 311]

Aspekte geht. Leider muss man festhalten, dass die gesellschaftliche Debatte sich in den Schulen kaum widerspiegelt, obwohl gerade die Jugendlichen und jungen Erwachsenen besonders stark von den gesellschaftlichen Auswirkungen betroffen sind, wie etwa die JIM-Studie [MP14] Jahr für Jahr erneut belegt.

Dies mag einerseits daran liegen, dass vielerorts kein echter Informatikunterricht angeboten werden kann, oder diesem kein hoher Stellenwert zukommt, weil es sich nach wie vor in den meisten Bundesländern um ein reines Wahlfach handelt. Andererseits ist der erteilte Informatikunterricht oft sehr technisch ausgerichtet und klammert gesellschaftliche Fragen aus. Wenn man sich mit gesellschaftlichen Auswirkungen beschäftigt, geht es meist um die negativen Folgen der Smartphone-Nutzung durch Schülerinnen und Schüler. Die Angst vor ständig verfügbarer Pornographie, Gewaltvideos und möglicher (Cyber-)Mobbingattacken dominiert die schulische Debatte. Auch die Ablenkung der Schülerinnen und Schüler im Unterricht ist ein beliebtes Thema. Die Folge daraus sind meist – zumindest teilweise – Verbote mobiler Informatiksysteme in der Schule, in manchen Bundesländern sogar mit Gesetzescharakter. Dass so eine tatsächliche Auseinandersetzung mit den gesellschaftsverändernden Einflüssen der mobilen Informatiksysteme unterbleibt und somit auf eine echte Vorbereitung auf die selbstbestimmte Teilhabe an der Gesellschaft in diesem Bereich verzichtet wird, wird als Kollateralschaden billigend in Kauf genommen.

Demgegenüber steht ein Trend zur – weitgehend – unkritischen Nutzung der mobilen Geräte im Unterricht (meist nicht im Rahmen der Informatik), der vor allem durch die Hersteller[5] befördert wird. Diese spiegelt letztlich nur das übliche Nutzungsverhalten innerhalb der Gesellschaft wider und bietet so zwar mediale Vorteile für den Unterricht, kann jedoch keinen größeren Beitrag zu einem mündigen, aufgeklärten Nutzungsverhalten leisten. Insbesondere der für die Informatik so wichtige Schritt der Rückführung der Erkenntnisse in die Realität, also die Umsetzung informatischer Modellierungen, oder wie Heming[He09] es beschreibt, die „Perspektive der Veränderung der Wirklichkeit“, wird so erschwert.

Der Informatikunterricht erfolgt weiterhin beinahe natürlich in Computerräumen. Die stationären Rechner erscheinen dabei als das zentrale Unterrichtsmittel und der zentrale Unterrichtsgegenstand. So ergab sich in einem Gespräch, das der Autor mit einer Gruppe von Grundschülern führte, die im Rahmen der anstehenden Anmeldungen seine Schule besuchten, dass – wie erwartet – niemand eine Vorstellung davon hatte, was sich wohl hinter seinem Fach „Sozialwissenschaften“ oder auch „Politikunterricht“ verbergen könnte. Ganz im Gegensatz dazu, hatten fast alle eine genaue Vorstellung davon, um was es in der Informatik gehen könnte, denn „das ist doch das Fach mit den Computern“. Diese eindimensionale Sicht zeigt sich nicht nur bei Grundschülern, sondern zieht sich durch die gesamte Gesellschaft.

[5] Allen voran Apple mit verschiedenen Programmen zur Förderung von „iPad-Klassen“.

3 Mobile Informatiksysteme im Informatikunterricht

Gerade auch um dieser Sichtweise auf die Informatik entgegenzuwirken, haben Carrie [Ca06] und Heming [He09] gezeigt, dass die ausschließliche Nutzung mobiler Informatiksysteme im Unterricht möglich ist. Auf Basis von Symbian S60 wurden hierzu Unterrichtskonzepte und Materialien erstellt und – insbesondere in den Pilotkursen an der Willy-Brandt-Gesamtschule Bergkamen – erprobt. Wichtig ist, dass alle folgend geschilderten Ansätze davon ausgehen, dass eine ausschließliche Nutzung der mobilen Informatiksysteme erfolgt, da ansonsten ein großer Teil der geäußerten Hoffnungen nicht erreichbar wäre und man weiterhin auf Computerräume angewiesen bliebe.

Im Folgenden soll ein – noch in Entwicklung befindliches, aber bereits erfolgreich eingesetztes – Konzept zur ausschließlichen Nutzung mobiler Informatiksysteme im Informatikunterricht vorgestellt werden, dass auch auf aktuellen Geräten nutzbar ist. Bei der Erprobung des Konzepts wurde besonders die hohe Motivation der Schülerinnen und Schüler deutlich.

Die hier vorgestellte Lösung funktioniert derzeit mit allen mobilen Informatiksystemen mit einem Android-Betriebssystem ab Version 2.2. Mit vergleichsweise geringem Aufwand können die privaten Geräte der Schülerinnen und Schüler genutzt werden. Es müssen lediglich eine App installiert und die entsprechende Bibliothek verteilt werden. Letzteres kann automatisiert mittels eines QR-Codes erfolgen.

4 Plattformauswahl

Analog zu [Ca06] sollte die Entwicklung für die mobilen Informatiksysteme mit den mobilen Informatiksystemen erfolgen. Somit fielen sämtliche Zugänge und Plattformen weg, die nur eine Entwicklung für die mobilen Systeme erlauben, hierzu aber auf weitere Informatiksysteme angewiesen sind. Diese Einschränkung war notwendig, um die aus der größeren Flexibilität und Unabhängigkeit von Computerräumen resultierenden, positiven Aspekte für den Informatikunterricht zu nutzen.

Im Wesentlichen blieb somit Android übrig[6], da alle weiteren geeigneten Plattformen bisher keinen relevanten Marktanteil erreichen konnten. Für Android ist die App QPython verfügbar (in je einer Variante für Python 2 und Python 3), die eine einfache IDE, eine simple Shell und diverse vorkonfigurierte Bibliotheken mitbringt, die etwa Zugriff auf Sensoren und Kameras erlauben. Dass durch Linkweiler in [Li02] die gute Eignung von Python für den schulischen Informatikunterricht festgestellt wurde, ist ein weiteres Argument für die Wahl der Plattform. Auch einem Verpacken der Skripte in Android-Packages und dem Vertrieb über den PlayStore steht nichts im Wege. Der Einfachheit halber begnügt

[6] Zum Zeitpunkt des Unterrichtsvorhabens galten noch die in [Sp12] genannten Einschränkungen von Apples iOS, sodass diese Plattform grundsätzlich nicht geeignet war. Inzwischen wurden jedoch wesentliche Vorgaben seitens Apple gelockert und es wurden einige Apps veröffentlicht, die die notwendigen Kriterien für den erfolgreichen Unterrichtseinsatz erfüllen. Obwohl es noch einige offene Probleme, etwa zum Dateiaustausch, zu lösen gilt, lässt insbesondere die App Pythonista hoffen, dass in Zukunft eine Umsetzung auf iOS möglich wird.

sich das Konzept bisher jedoch mit reinen Skripten und einer dialogbasierten Benutzungsoberfläche.

Über die Scripting-API ist der Zugriff auf fast alle Komponenten des Android-Systems möglich. Die diversen Sensoren lassen sich dabei ebenso einfach auslesen wie der Touchscreen, es lassen sich Bilder, Videos[7] und Audio-Clips aufzeichnen. Außerdem lässt sich natürlich auch die Position mittels GPS und Mobilfunk bestimmen und mittels Bluetooth-API kommuniziert man mit anderen Geräten. All dies funktioniert auch auf sehr alten und damit günstigen Geräten ohne Probleme.

5 Definition einer Schnittstelle

Die von Google bereitgestellte Scripting-API kapselt Remote-Procedure-Calls an einen im Hintergrund laufenden Server-Dienst. Die Folge ist, dass die API weder objektorientiert noch ansatzweise selbsterklärend ist (vgl. Listing 1). Da für den Unterricht einerseits die Objektorientierung benötigt wird, andererseits eine möglichst eindeutige und klare Schnittstelle erwünscht ist, ist es notwendig, einen entsprechenden Wrapper als Schnittstelle zu definieren, der die benötigten API-Aufrufe in Objekte kapselt (vgl. Listing 2).

Es hat sich als erfolgreich erwiesen, die Objekte im Unterricht zu entwickeln und daraus einen entsprechenden Wrapper zu erstellen, mit dem die Schülerinnen und Schüler weiterarbeiten können. Der Aufwand dafür ist nicht besonders hoch, zumal zu erwarten ist, dass sich die Modellierungen verschiedener Kurse hier nicht so gravierend unterscheiden, dass grundlegende Änderungen notwendig sind. Außerdem beschränken sich die jeweils zu ergänzenden API-Aufrufe auf wenige Zeilen.

```
1 import android
2 droid = android.Android()
3
4 droid.dialogCreateAlert("Hallo Klasse!","Bitte Ok druecken.")
5 droid.dialogSetPositiveButtonText("OK")
6 droid.dialogShow()
7 result = droid.dialogGetResponse().result
8 droid.dialogDismiss()
```

Listing 1: Hallo Klasse, API

```
1 from velamentum.alle import *
2
3 dialog = Dialog()
4
5 dialog.titel = "Hallo Klasse!"
6 dialog.nachricht = "Bitte Ok druecken."
7
8 dialog.zeige()
```

Listing 2: Hallo Klasse, objektorientiert

[7] Aufgrund geänderter Sicherheitsrichtlinien in neueren Android-Versionen ist dies nur noch nach einer Interaktion des Anwenders möglich.

Ein Beispiel für eine solche Umsetzung findet sich in Abbildung 1. Der Nutzen liegt klar auf der Hand, denn es kann direkt an die Arbeitsergebnisse angeknüpft werden, was zur Kontinuität des Unterrichts beiträgt und thematische Sprünge vermeidet. Voraussetzung dafür ist natürlich, dass die Modellierung von Objekten ausreichend eingeübt und verstanden wurde.

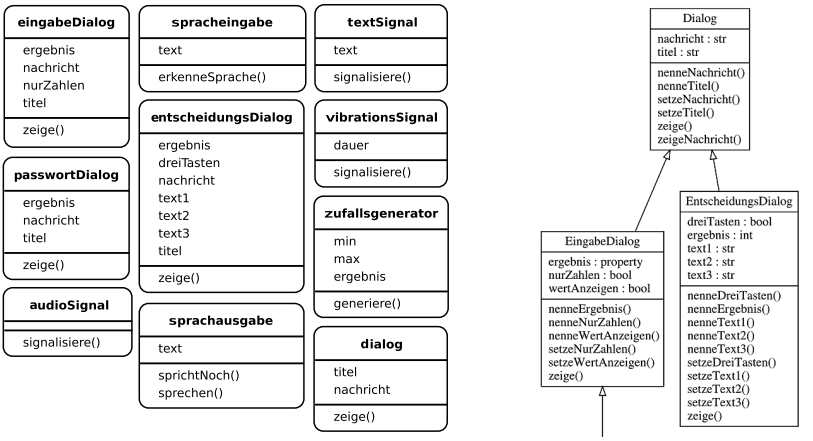
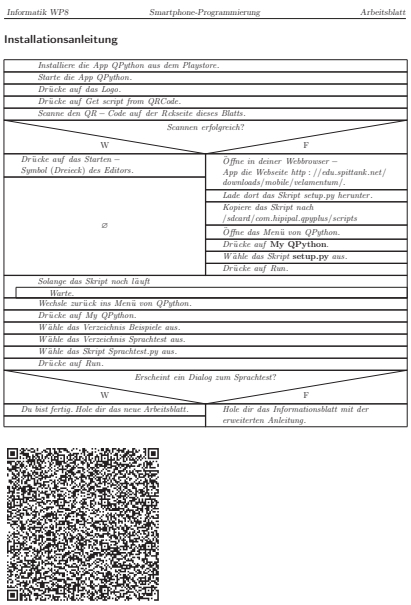


Abb. 1: Von den Schülerinnen und Schülern erarbeitete Objektübersicht und resultierendes Klassendiagramm (Auszüge)

Da dies trotzdem nicht immer möglich sein wird, befindet sich derzeit eine Referenzimplementierung für die direkte Nutzung im Unterricht in Entwicklung. Diese und die bisher erstellten Materialien sind unter [Sp15] frei verfügbar.

6 Möglichkeiten der Ersteinrichtung im Unterricht

Die erste Einrichtung der Geräte für die Programmierung ist nicht weiter schwierig. Es muss lediglich die QPython-App (bisher noch vorzugsweise die für Python 2 und nicht für Python 3) installiert und anschließend der erstellte Wrapper in das Library-Verzeichnis kopiert werden, welches sich auf der SD-Karte bzw. dem entsprechend eingebundenen, internen Speicher befindet. Dies kann manuell erfolgen, ist dann allerdings mit umständlicheren Kopier-Aktionen verbunden. Um den Zugang für die Schülerinnen und Schüler möglichst einfach zu gestalten und während der Unterrichtsreihe auch geänderte Versionen der Bibliothek verteilen zu können, ist also eine automatisierte Lösung wünschenswert. Da bis vor kurzem keine einheitliche Methode hierfür in QPython existierte (etwa die Python-Setuptools) und diese nach wie vor nicht absolut zuverlässig funktionieren, wird hierfür ein eigenes Python-Skript verwendet, das die benötigten Dateien aus dem Internet nachlädt und die Dateien installiert. Dieses Skript wird – wie es für QPython typisch ist – aus einem QR-Code geladen. Die Installationsanleitung wurde dabei zur Übung des bereits Gelernten als umgangssprachliches Struktogramm festgehalten (vgl. Abbildung 2).



Informatik WPS *Smartphone-Programmierung* *Arbeitsblatt*

Installationsanleitung

Installiere die App QPython aus dem Playstore.	
Starte die App QPython.	
Drücke auf das Logo.	
Drücke auf Get script from QRCode.	
Scanne den QR – Code auf der Rckseite dieses Blatts.	
Scannen erfolgreich?	
W	F
Drücke auf das Starten – Symbol (Dreieck) des Editors.	*Öffne in deiner Webbrowser – App die Webseite http : //edu.spittank.net/ downloads/mobile/velamentum/.*
	Lade dort das Skript setup.py herunter.
	Kopiere das Skript nach /sdcard/com.hipipal.qpyplus/scripts
ø	*Öffne das Menü von QPython.*
	Drücke auf **My QPython**.
	Wähle das Skript **setup.py** *aus.*
	Drücke auf Run.
Solange das Skript noch läuft	
Warte.	
Wechsle zurück ins Menü von QPython.	
Drücke auf My QPython.	
Wähle das Verzeichnis Beispiele aus.	
Wähle das Verzeichnis Sprachtest aus.	
Wähle das Skript Sprachtest.py aus.	
Drücke auf Run.	
Erscheint ein Dialog zum Sprachtest?	
W	F
Du bist fertig. Hole dir das neue Arbeitsblatt.	*Hole dir das Informationsblatt mit der erweiterten Anleitung.*

Informatik WPS *Smartphone-Programmierung* *Arbeitsblatt*

Eine App für die Schülerzeitung

Die Reporter der Schülerzeitung führen viele Interviews. Dabei sind Aufzeichnungen sinnvoll, damit die Interviews später korrekt abgedruckt werden können. Leider sind die Aufzeichnungsgeräte beim Einbruch vor zwei Wochen gestohlen worden. Die Reporter haben sich mit ihren Smartphones geholfen, allerdings haben die verfügbaren Rekorder-Apps einige Schwächen. So schalten sie etwa die Aufnahme ab, wenn das Display in den Stromsparmodus wechselt. Daher haben die Reporter angefragt, ob der Informatikkurs nicht eine bessere App entwickeln kann.

1. **Aufgabe** Entwickelt zunächst ein Skript, dass eine Aufnahme vom Mikrofon ermöglicht, die auf Knopfdruck beendet werden kann.

   Hinweis: Ermittelt zunächst wieder die benötigten Objekte und überlegt euch dann, wie diese zusammenspielen müssen.

2. **Aufgabe** Viele Interviewte wollen gerne noch einmal kontrollieren, was sie gesagt haben, damit sie nicht mit unangenehmen oder peinlichen Aussagen zitiert werden.

   Ergänzt eine Nachfrage nach der Beendigung der Aufnahme, die es erlaubt, die Aufnahme direkt anzuhören.

3. **Aufgabe** Als Reporter ist es wichtig, dass man das Einverständnis dazu hat, ein Interview aufzunehmen. Damit der Interviewpartner genau weiß, wann eine Aufnahme läuft, soll das Smartphone sowohl beim Starten, als auch beim Stoppen der Aufnahme einen entsprechenden Hinweis über die Lautsprecher ausgeben.

   Verwendet die Sprachausgabe, um entsprechende Hinweise auszugeben.

4. **Aufgabe** Die Sprachausgabe ist nicht auf allen Smartphones gut zu verstehen, daher soll die Ansage durch eine menschliche Stimme erfolgen. Erstellt die Ansagen und ändert euer Skript so, dass diese verwendet werden.

Abb. 2: Installationsanleitung und Arbeitsblatt

Sollte in der Schule kein WLAN mit Internetzugang zur Verfügung stehen, kann ein lokaler Server für die Verteilung verwendet werden. WLAN und Server können im Notfall auch durch ein entsprechend vorbereitetes Android-Gerät bereitgestellt werden. Ein entsprechendes Setup wurde in einem der beiden Kurse erfolgreich erprobt.

Die Verteilung der notwendigen Dateien kann auch als Wettbewerb organisiert werden. Die zu entwickelnde Verteilungsstrategie mittels Bluetooth oder Speichermedien kann dabei wiederum als Struktogramm festgehalten werden und anschließend unter Effizienzgesichtspunkten beurteilt werden.

Trotzdem ergeben sich aus der Vielfalt der von den Schülerinnen und Schülern genutzten Geräte gewisse Schwierigkeiten. So ist aufgrund von herstellerspezifischen Anpassungen die Installation auf einigen Geräten schwierig. Besonders neuere Geräte von Samsung und einzelne Geräte von HTC verhielten sich hier „zickig“. Mal fehlten Schreibrechte, mal beendeten installierte Systemdienste den API-Server. Bis alle Geräte vollständig genutzt werden konnten, vergingen rund drei Schulstunden. Danach fiel allerdings keinerlei Wartungsaufwand mehr an.

7 Weiteres Vorgehen

Die gesamte Umsetzung im Unterricht kann stark von kooperativen Methoden profitieren. Zur Entwicklung des Modells für den Wrapper wurden etwa Brainstorming und Welt-Café-Methode herangezogen, die Ergebnisse wurden dabei auf Plakaten festgehalten, die dann durch die Lehrkraft in den Wrapper überführt werden konnten. Die Plakate wurden zur weiteren Arbeit immer wieder genutzt, da sie so gleichsam die Dokumentation des Wrappers darstellten.

Am Ende des Vorhabens sollten fertige Apps und eine Projektdokumenation als Produkte einzelner Kleingruppen stehen. Um die Anforderungen zu verdeutlichen, wurde daher die Heranführung an die Programmierung mittels zweier Beispiel-Projekte durchgeführt, die gemeinsam erarbeitet wurden. Zunächst sollte das Spiel „Stein-Schere-Papier“ entwickelt werden. Hierzu wurden die Regeln, die benötigten Objekte und ein vorgesehener Programmablauf innerhalb mittels der Welt-Café-Methode ermittelt. Anschließend wurden die Skripte in Partnerarbeit implementiert und mit einem anderen Paar verglichen. Während der Arbeit war eine sehr starke, fachbezogene Kommunikation erkennbar. Besonders bei der Verifikation wurde deutlich, dass es keine künstlichen Barrieren gab. Smartphones wurden zwischen den Gruppenangehörigen ausgetauscht und die Ergebnisse verglichen. Der Wechsel zwischen Planung, Implementierung und Verifikation war dabei mühelos und wurde durch die mobilen Geräte begünstigt. Auch der Wechsel zwischen den Gruppen war zügig möglich. Leistungsstärkere Paare ergänzten das Skript dann noch um weitere Fälle. Das zweite, anspruchsvollere Skript – ein Diktiergerät für die Schülerzeitung – wurde nach demselben Muster umgesetzt (vgl. Abbildung 2).

Für die eigenen Projektideen wurden sehr früh Zielfestlegungen getroffen, die durch die Lehrkraft auf ein erreichbares Maß reduziert wurden. Die Bearbeitung erfolgte dann in kleinen Projektgruppen nach dem bekannten Verfahren: Beschreibung der Idee, Ermittlung der Objekte und des Programmablaufs und schließlich Implementierung. Die Projektdokumentationen waren dabei teilweise sehr umfassend und gingen deutlich über den erwarteten Umfang hinaus.

8 Fazit

Im Großen und Ganzen kann das Experiment als Erfolg verbucht werden. Die erhofften positiven Aspekte konnten erreicht werden, einzig die Erstinbetriebnahme gestaltete sich schleppend. Eine echte Verbesserung ist hier allerdings beim „Bring your own device“-Ansatz kaum erreichbar. Zu unterschiedlich sind die Herstellermodifikationen an Android.

Dennoch hatten außnahmslos alle Schülerinnen und Schüler sichtbar Spaß an der Entwicklung mit den Smartphones und viele haben auch außerhalb des Unterrichts und sogar nach Ende des Unterrichtsvorhabens weiter an ihren Apps gearbeitet. Eingeschobene, kurze Lernzielkontrollen zeigten, dass auch die zugrundeliegenden Konzepte verstanden wurden, sodass das Fazit durchweg positiv ausfällt, insbesondere auch von Seiten der Schülerinnen und Schüler.

Durch die Verwendung der mobilen Geräte wurde zudem tatsächlich dem einseitigen Bild der Informatik als Computerwissenschaft entgegengewirkt und eine stärkere Anbindung an den Alltag der Schülerinnen und Schüler erreicht, wie der Vergleich mit den Parallelkursen zeigte.

Literaturverzeichnis

[Ca06] Carrie, Ralph: Einsatz mobiler Informatiksysteme im Informatikunterricht der gymnasialen Oberstufe. Hausarbeit gemäß OVP, Studienseminar für Lehrämter an Schulen – Seminar für das Lehramt für Gymnasien Gesamtschulen, Hamm, 2006.

[GH05] Görlich, Christian F.; Humbert, Ludger: Open Source – die Rückkehr der Utopie? In: Open Source Jahrbuch 2005. Zwischen Softwareentwicklung und Gesellschaftsmodell. S. 311–327, 2005.

[He09] Heming, Matthias: Einsatzszenarien von Mobiltelefonen im Informatikunterricht. Masterarbeit – Master of Education, Bergische Universiät – Fachbereich Mathematik und Naturwissenschaften, Wuppertal, 2009.

[Li02] Linkweiler, Ingo: Eignet sich die Skriptsprache Python für schnelle Entwicklungen im Softwareentwicklungsprozess? Eine Untersuchung der Programmiersprache Python im softwaretechnischen und fachdidaktischen Kontext. Diplomarbeit, Universität, Fachbereich Informatik, Fachgebiet Didaktik der Informatik, Dortmund, 2002.

[MP14] MPFS: JIM 2014. Jugend, Information, (Multi-)Media. Basisuntersuchung zum Medienumgang 12- bis 19-Jähriger in Deutschland. Forschungsbericht, mpfs, Stuttgart, 2014. MPFS – Medienpädagogischer Forschungsverbund Südwest.

[Sp12] Spittank, Daniel: Auswahl und Gestaltung mobiler Informatiksysteme für den Einsatz im Informatikunterricht. Staatsexamensarbeit, Bergische Universiät – Fachbereich Mathematik und Naturwissenschaften, Wuppertal, 2012.

[Sp15] Spittank, Daniel: , https://edu.spittank.net/downloads/mobile, 2015.

Ein RFID-Projekt in der Fachinformatiker-Ausbildung unter Berücksichtigung von Threads, Software-Reviews und der Methode Webquest

Peer Stechert[1]

Abstract: Radio Frequency Identification (RFID) wird zukünftig die technische Grundlage für das „Internet der Dinge" bilden. Aus der Veränderung der Lebenswelt folgen neue Anforderungen für den allgemeinbildenden und den berufsbildenden Informatikunterricht (vgl. [Jo06], [Ko11]). In dem vorliegenden Artikel wird die erstmalige Durchführung eines Unterrichtsprojekts für den Ausbildungsberuf FachinformatikerIn[2] mit Fachrichtung Anwendungsentwicklung vorgestellt. Darin wurde die RFID-Technologie handlungsorientiert erfahrbar gemacht. Das Thema Threads wurde anhand des Philosophenproblems mittels Rollenspiel eingeführt, Software-Reviews als strukturierte Vorgehensweise eingesetzt und als Fixpunkte der Projektmethode dienten tägliche Kurz-Rückmeldungen in Anlehnung an das „Daily Scrum" der agilen Softwareentwicklung. Um die Reflexion der Schüler anzuregen, haben sie für andere Schülergruppen Webquests zur Einführung in die Thematik projektbegleitend erstellt.

Keywords: Fachinformatiker-Ausbildung, Handlungsorientierung, RFID, Software-Reviews, Methode Webquest, Threads, Daily Scrum

1 Motivation und Zielvorstellung

Zielgruppe des Unterrichtsprojekts waren angehende Fachinformatiker mit Fachrichtung Anwendungsentwicklung. Das Projektziel war die Ansteuerung eines RFID-Controllers und das Auslesen des zugehörigen Transponders (als Minimalziel) und darüber hinaus die Erstellung einer selbst gewählten RFID-Anwendung, z. B. Zutrittskontrolle oder Zeiterfassung. Dabei wurden die Themen „Nebenläufigkeit mit Threads" und „Software-Reviews" aufgegriffen. Zur Festigung der Kenntnisse über die Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS) erstellten die Schüler ein Webquest zum Einstieg in die Programmierung von RFID-Controllern: Dies erfordert einen Sichtenwechsel bei den Schülern im Sinne des „Lernens durch Lehren", denn das Webquest sollte andere Schüler unterstützen, sich in das Themengebiet einzuarbeiten.

Der Artikel ist folgendermaßen strukturiert: In Abschnitt 2 werden die Rahmenbedingungen und die Planung des Unterrichtsprojekts vorgestellt. In Abschnitt 3 wird der Unterrichtsverlauf anhand der Kernelemente skizziert. Der Artikel schließt mit einem Ausblick auf mögliche Projektvarianten (Abschnitt 4).

[1] Regionales Berufsbildungszentrum Technik Kiel, Geschwister-Scholl-Straße 9, 24143 Kiel, peer.stechert@rbz-technik.de

[2] Alle Personenbezeichnungen gelten gleichermaßen für die männliche und weibliche Form.

2 Unterrichtsplanung

2.1 Lerngruppe, technischer und zeitlicher Rahmen

Jeweils vor den Sommerferien kommen die Auszubildenden im zweiten Ausbildungsjahr für einen knapp dreiwöchigen Block in das Berufsbildungszentrum, in dem an neun Tagen die Projektarbeit in Vollzeit stattfindet. Insgesamt 10 Schüler nahmen beim ersten Durchlauf an dem RFID-Projekt teil. Die Schüler befanden sich hinsichtlich Anwendungsentwicklung auf sehr hohem, aber durch die betriebliche Ausbildung auf verschiedenen Leistungsniveaus.

Zur Verfügung gestellt wurden der RFID-Controller (ICR40 der Firma Megaset Systemtechnik) mit serieller Schnittstelle, Transpondern (Hitag 1, Hitag 2) und Dokumentation sowie eine einfache Beispielapplikation. Gemäß Schulcurriculum kennen die Schüler am Ende des 2. Ausbildungsjahres Grundlagen der Objektorientierung mit Klassen, Objekten und Methoden. Außerdem sind ihnen Datentypen und Kontrollstrukturen bekannt. Als Programmiersprache kommt am RBZ Technik Visual C# von Microsoft zum Einsatz. Die Sprache ist Java der Firma Oracle sehr ähnlich und zeichnet sich durch eine im Unterricht gut einsetzbare Entwicklungsumgebung, Visual Studio, aus. In der Express-Version ist die Software für Schüler auch zu Hause kostenlos nutzbar. Die Schüler haben bereits mit WindowsForms-Anwendungen gearbeitet. Datenbanken waren noch nicht Thema. Diese bilden aber einen möglichen Anknüpfungspunkt an die RFID-Technologie, da Vorkenntnisse zu Datenbanken aus den Betrieben zu erwarten sind.

2.2 Einordnung der Unterrichtseinheit in Lehrplan und Bildungsstandards

Zur Auswahl der Lerninhalte wurde das erweiterte Zielebenenmodell nach Hartmann, Näf und Reichert herangezogen [HNR06]. Leitidee des Unterrichts ist, dass Absolventen der Fachrichtung Anwendungsentwicklung in der Lage sein sollen, qualitativ hochwertige, komplexe Software in Teams zu entwickeln. Das Lernfeld 6 „Anwendungsentwicklung“ beinhaltet laut Rahmenlehrplan die Systementwicklung in Projektarbeit [KMK97]. Damit ist die besondere Hervorhebung der Software-Reviews begründet, die auch im Rahmenlehrplan in Form von „Testverfahren“ explizit für Schüler der Fachrichtung Anwendungsentwicklung gefordert wird [KMK97, S. 17]. Sie kann zusätzlich über die fundamentalen Ideen „Konsistenz“, „Vollständigkeit“ sowie „Verifikation“ nach Schwill legitimiert werden [SS11, S. 74]. Ein Dispositionsziel ist, dass den Lernenden das Vorhandensein unterschiedlicher Strategien zur Sicherung von Softwarequalität bewusst ist und sie solche Software-Reviews in beruflichen Situationen heranziehen. Die fundamentale Idee der Informatik „Teamarbeit“ liefert eine weitere Begründung für die Projektarbeit im Unterricht. Entsprechendes Dispositionsziel ist, dass die Schüler lernen, ihre Arbeit selbst zu organisieren, und sie dies auch beruflich für lebenslanges Lernen nutzen.

Die im Rahmenplan adressierte Komplexität von Anwendungssystemen wird in der Softwareentwicklung oft durch Aufteilung von Aufgaben auf unterschiedliche Threads bewältigt, die (quasi-) nebenläufig arbeiten. Die fundamentalen Ideen der Informatik „Parallelisierung“ und „Prozess“ [SS11, S. 74] liefern darüber hinaus eine Begründung für Threads im geplanten Unterricht.

Obgleich die Bildungsstandards der Gesellschaft für Informatik e.V. für die Sekundarstufe I entworfen wurden [GI08], kann eine Einordnung des Themas in den Inhaltsbereich „Informatiksysteme“ vorgenommen werden. Operationalisierte Unterrichtsziele sind:

Fachkompetenz: Die Schüler …

- übermitteln Kommandos zur Steuerung des RFID-Controllers über die serielle Schnittstelle.
- nutzen Threads in der Task Parallel Library in Visual C#, mit der einfache Synchronisationen unterstützt werden, da Nebenläufigkeit zu unvorhergesehenen Programmzuständen führen kann.
- lernen Software-Reviews als Mittel zur Qualitätssicherung kennen und wenden sie exemplarisch an.
- lesen und verarbeiten Daten des Transponders.
- erkennen die Notwendigkeit von Datenschutz.
- sind in der Lage, mit HTML und CSS ein Webquest zu erstellen.

Methodenkompetenz: Die Schüler …

- erkennen die Sinnhaftigkeit der Projektarbeit, festigen ihre Team-Fähigkeit und „planen Arbeitsabläufe und Handlungsfolgen.“ [GI, 2008, S. 20]
- erstellen ein eigenes Webquest als Teil der Dokumentation.
- sind in der Lage, fehlende Information der mitgelieferten Geräte-Dokumentation zu entnehmen.
- erstellen eine reflektierende Dokumentation ihrer Projektarbeit und „stellen informatische Sachverhalte unter Benutzung der Fachsprache schriftlich sachgerecht dar.“ [GI08, S. 21]
- nutzen die Lernplattform moodle.

Sozialkompetenz: Die Schüler …

- arbeiten im Team und stimmen sich zur Zusammenführung arbeitsteiliger Ergebnisse ab.
- „kommunizieren mündlich strukturiert über informatische Sachverhalte.“ [GI, 2008, S. 21]
- „benennen Wechselwirkungen zwischen Informatiksystemen und ihrer gesellschaftlichen Einbettung.“ [GI, 2008, S. 18]

Personalkompetenz: Die Schüler …

- sind in der Lage, eigenverantwortlich ihr Projekt zu organisieren, um auf weitere Projektarbeiten in der Schule und im Betrieb vorbereitet zu sein.
- reflektieren ihre Arbeit innerhalb des Projektteams.

2.3 Handlungsorientierung und Projektmethode

Ziel der Fachinformatiker-Ausbildung ist, dass die Schüler selbst aktiv werden gemäß Handlungsorientierung (vgl. [Gu01]): *„Den Ausgangspunkt des Lernens bilden Handlungen, möglichst selbst ausgeführt oder aber gedanklich nachvollzogen (Lernen durch Handeln). Handlungen müssen von den Lernenden möglichst selbständig geplant, durchgeführt, überprüft, ggf. korrigiert und schließlich bewertet werden. […] Handlungen müssen in die Erfahrungen der Lernenden integriert und in bezug auf ihre gesellschaftlichen Auswirkungen reflektiert werden“* [KMK97, S. 2].

Insbesondere für die Fachinformatiker-Ausbildung gilt, dass Programmierkenntnisse und

Projekterfahrung nur vermittelt werden können, wenn die Schüler eigenständig Programme erschaffen und ihre Kenntnisse anwenden: *„We can also distinguish between disciplines in which there is an emphasis on learning through interpreting and those in which learning is predominantly achieved through doing. […] Computing students are expected to do a lot of learning through doing, whether it is learning about software engineering by developing systems of increasing complexity, learning about networking by implementing protocols or learning about group dynamics by working in teams"* [Fu07, S. 163]. Fuller et al. schreiben dem Anwenden und Erschaffen in der überarbeiteten Bloom'schen Lernzieltaxonomie für die Informatik größte Bedeutung zu.

Der projektorientierte Unterricht ermöglicht es, dass jede Schülergruppe Verantwortung für ihr eigenes Lernen übernimmt, indem sie ihr Lerntempo und – über das Minimalziel des Projekts hinaus – selbstdefinierte Projektziele wählt. Dadurch werden Überforderung und Unterforderung vermieden. Bei der Projektmethode ist der Konflikt zwischen den pädagogischen und informatischen Zielen der Methode zu beachten: In Schulprojekten steht im Vordergrund, dass Schüler an allen Arbeiten im Team als Generalisten beteiligt sind. In informatischen Projekten steht die Effizienzsteigerung durch unabhängige Bearbeitung von Teilaufgaben und Spezialistentum an erster Stelle [SS11, S. 308]. In dem RFID-Projekt wird dieser Effekt abgemildert, da durch kleine Zweiergruppen (a) die Teammitglieder gleichberechtigt sind und (b) die Aufgabenstellung den Zwang zur Zusammenarbeit als Berührungspunkt enthält: *„Auf diese Weise werden die positiven Elemente beider Sichtweisen von Projekten, der pädagogischen wie der informatischen, nach Möglichkeit erhalten"* [SS11, S. 313].

Nach Frey sind Fixpunkte während der Projektphase notwendig [Fr07]. Deshalb wurden ein Software-Review sowie tägliche Kurz-Rückmeldungen (Daily Scrum) im Plenum in Anlehnung an das Scrum-Vorgehensmodell der agilen Softwareentwicklung eingeplant.

3 Lernphasen und Ablauf des Projekts

Projektinitiierung

Zu Beginn wurden ein zeitlicher Ablauf des Projekts mit Fixpunkten und das Ziel des Projektes vorgestellt. Von den Schülern war eine frei wählbare RFID-Anwendung mit folgenden Eigenschaften zu gestalten: (1) Es wird eine Visual Studio C#-Anwendung als WindowsForms-Projekt realisiert. (2) Die Kommunikation zwischen RFID-Controller und Anwendung geschieht über die serielle Schnittstellte mit der Komponente `SerialPort` (Anmerkung: Alternativ ist auch die Nutzung einer USB-Schnittstelle möglich. Aufgrund der vorhandenen RFID-Systeme mit serieller Schnittstelle wurde jedoch diese Konfiguration gewählt). (3) Die Anwendung verfügt über Einstellmöglichkeiten der Verbindungsparameter der seriellen Schnittstelle, z. B. Baud-Rate, Start-, Stopp- und Datenbits. (4) Außerdem müssen Threads eingesetzt werden, um die Anwendung während des Sendens und Empfangens der Daten reaktionsfähig zu halten und die getrennt laufenden Threads von Benutzungsoberfläche und

`SerialPort` zu synchronisieren.

Als Hilfestellungen wurden von den Lehrkräften zwei Einzelmodule vorbereitet: (1) Kommunikation zwischen zwei Rechnern über die Komponente `SerialPort` als Chatsystem, (2) Kommunikation mit dem RFID-Lesegerät. Die Schüler teilten sich selbständig in Zweiergruppen ein, bestimmten ein Anwendungsszenario und konnten den Weg zur Lösung selbst wählen: Gruppe 1 (Messung von Rundenzeiten), Gruppe 2 (Türöffner), Gruppe 3 (Bibliothek), Gruppe 4 (Zeiterfassung), Gruppe 5 (Gesundheitskarte).

Webquests als Unterrichtsmethode einmal anders: Lernen durch Lehren

Webquests sind eine angeleitete Methode zur (Internet-) Recherche[3]. Sie bestehen aus lokal oder im Internet verfügbaren Webseiten, die folgende Struktur aufweisen: 1) Das Thema wird eingeführt, 2) Aufgaben werden gestellt, 3) Materialien (Webseiten, aber auch Bücher, Zettel im Klassenzimmer) werden aufgelistet, 4) der Arbeitsprozess mit Hilfestellungen zu den Aufgaben wird beschrieben, 5) eine Evaluationsmöglichkeit wird gegeben, zum Beispiel zur Selbsteinschätzung für die Schüler in einer Kompetenzmatrix, und 6) eine Präsentation der Projektergebnisse wird erwartet. Letztere ist nicht mehr Teil der Webseiten. Insbesondere die Prozesshinweise sorgen für einen sehr strukturierten Unterrichtsverlauf, der laut Hattie-Studie positiv für das Lernen ist.

Eine Teilaufgabe der Schüler war es, als Dokumentation ihrer Projektarbeit ein Webquest zu erstellen, das als Unterrichtseinstieg in die RFID-Programmierung über die serielle Schnittstelle für andere Schülergruppen genutzt werden kann. Damit schlüpfen die Schüler in die Rolle von Lehrenden und müssen projektbegleitend ihr eigenes Lernen reflektieren. Nach der Projektinitiierung wurde den Schülern deshalb die Methode Webquest vorgestellt, wobei der Unterricht selbst an der Struktur eines Webquests orientiert war: Das Thema wurde in einer kurzen Präsentation aufbereitet, die in der obigen, konkreten Aufgabenstellung mündete. In dem moodle-Kurs zu dem Projekt wurden Ressourcen angeboten, die neben Beschreibungen der Methode auch Beispiele für gelungene Webquests enthielten. Nach diesem Einstieg konnten die Schüler projektbegleitend recherchieren, wobei die Lehrpersonen als „Recherche-Coach" fungierten. Webseitenerstellung mit HTML, XHTML, JavaScript oder CSS war noch nicht Thema im Unterricht, aber zum Teil aus dem Betrieb bekannt, so dass die Schüler sich im Sinne der Handlungsorientierung informieren und entscheiden mussten, welche Technologie eingesetzt wird.

Threads: Das Philosophenproblem als Rollenspiel

Als Einstieg in das Thema Threads wurde das „Philosophenproblem" von Dijkstra gewählt. Philosophen repräsentieren Threads und zeichnen sich dadurch aus, dass sie abwechselnd essen und denken. Bei fünf Philosophen, die um einen Tisch sitzen, werden fünf Gabeln zwischen den Philosophen positioniert. Diese repräsentieren die für eine

[3] http://www.webquests.de/eilige.html (Abruf: 17.04.2015)

Bearbeitung durch einen Thread notwendigen Betriebsmittel oder Daten. Ein Philosoph kann nur essen, wenn er beide Gabeln neben ihm greifen kann. Nach dem Essen werden die Gabeln wieder auf den Tisch zurückgelegt. Die Schüler wurden in Philosophen und Beobachter eingeteilt. Die Philosophen nahmen um einen Tisch mit Fruchtgummi Platz.

Dabei wurden drei verschiedene Szenarien durchgespielt. (1) Zuerst wurde erwähnt, dass ein Philosoph nur essen kann, wenn er zwei Gabeln greifen kann. Die Beobachtung war, dass die Philosophen nicht nur nach Gabeln direkt neben sich griffen. Ein Schüler fasst zusammen: *„Die Philosophen essen nacheinander, nie alle gleichzeitig“*. (2) Als zweites Szenario wurde vorgegeben, dass Philosophen erst ihre rechte, dann die linke Gabel aufnehmen. Beobachtet wurde, dass die schnellsten Philosophen zuerst aßen, nie alle gleichzeitig. (3) Drittes Szenario war, dass alle Philosophen gleich agieren sollten, analog zu programmierten Objekten. Jetzt war die Beobachtung, dass alle Philosophen gleichzeitig die rechte Gabel nahmen und keiner essen konnte. Diese Situation der „Verklemmung“ (engl.: deadlock) war durch das Rollenspiel für die Schüler direkt erfahrbar. Durch diese Einengung auf Abarbeitungsvorschriften wurde die Situation im Computer angenähert, in der Verklemmung von Threads zu vermeiden ist: Diese entsteht durch wechselseitige Abhängigkeiten bei den zugewiesenen Betriebsmitteln.

Im anschließenden, wenige Minuten umfassenden Lehrervortrag wurden (1) der Unterschied zwischen Prozessen und Threads und (2) Multithreading in C# vorgestellt. In Visual C# ergänzt die `Task Parallel Library` (TPL) die Klasse `Thread`. Mit ihr ist es möglich, Threads unter optimaler Ausnutzung der Hardware zur Laufzeit automatisch zu erstellen, d. h. wenn die Anwendung auf einem Dual-Core-Prozessor läuft, werden zwei Threads erzeugt. Dadurch wird es sogar einfacher, mit Threads zu arbeiten, was ein wichtiges Kriterium hinsichtlich einer angemessenen didaktischen Reduktion ist. Dabei wird an Vorkenntnisse zu `Delegates` aus dem Anwendungsentwicklungs-Unterricht angeknüpft, die für die TPL notwendig sind. Solche `Delegates` der ereignisgesteuerten Programmierung stellen erfahrungsgemäß eine hohe kognitive Hürde dar.

„Daily Scrum“ als Fixpunkte der Projektmethode

Beobachtungen des selbstbestimmten Arbeitsprozesses sind in projektorientiertem Unterricht besonders wichtig. Deshalb wurde vereinbart, dass die Gruppen beim jeweils nächsten Treffen als kurzes Blitzlicht zu ihrem Arbeitsstand folgende Fragen beantworten: Welche Aufgaben wurden seit dem letzten Bericht fertiggestellt? Welche Aufgaben werden bis zum nächsten Treffen bearbeiten? Gibt es Probleme, die die Gruppe bei ihren Aufgaben behindern? Diese drei Fragen sind angelehnt an den kurzen, täglichen Informationsaustausch „Daily Scrum“ im Scrum-Vorgehensmodell der agilen Softwareentwicklung [Kl09]. Die Rückmeldungen dienten einerseits als Fixpunkt im Unterricht und andererseits erlaubten sie den Lehrkräften, gezielt Hilfestellung zu geben bzw. Austausch zwischen den Projektteams zu ermöglichen. Erkennbar war, ob sich die Gruppenmitglieder dafür einsetzten, dass der Arbeitsauftrag erledigt wird und wie die gruppeninternen Regeln aussehen, z. B. zur Aufgabenverteilung. Oft wurde gerade durch

die Befragung eine konstruktive Kommunikation angeregt, Arbeitsvorschläge Einzelner wurden diskutiert und fachliche Fragen an die Lehrkräfte gestellt. Fazit ist, dass dieses kurze Blitzlicht die Reflexion über die Arbeitsweise innerhalb der Gruppe angeregt hat.

Software-Reviews als Unterrichtsmethode

Fixpunkt der praktischen Phase bildete das Software-Review eines von einer Schülergruppe erstellten Programmteils. Software-Reviews (auch „Software-Inspektion"; [SL10]) zählen zu den statischen White-box-Tests. Das Software-Review wurde ausgewählt, weil es als strukturierte Vorgehensweise unterrichtsmethodisch wertvoll ist: Mehrere Schüler werden durch Übernahme bestimmter Rollen mit unterschiedlichen Beobachtungsschwerpunkten aktiv. Dabei führte die gemeinsame Überprüfung zu einem Wissensaustausch, der die Arbeitsweisen der einzelnen Gruppen und die Qualität der erstellten RFID-Anwendungen verbessert sowie Entwicklungszeit verkürzt. Gruppe 2 (Türöffner) stellte ihre Klasse `SerialRFID` und das selbst definierte Interface `RFIDInterface` als Reviewobjekt zur Verfügung. Beide Schüler der Gruppe 2 nahmen die Rolle der Autoren ein. Außerdem war ein Schüler Moderator, einer Protokollant und alle anderen waren Gutachter, denen unterschiedliche Beobachtungsschwerpunkte zugeordnet wurden. Besonders gut lassen sich durch Reviews Abweichungen von Standards sowie Fehlerursachen finden. Darüber hinaus sind Aspekte wie Design-Fehler, unzureichende Wartbarkeit, fehlerhafte Schnittstellen und fehlende Anforderungen erkennbar. Der Quellcode wurde vorab verteilt, so dass alle Schüler sich einarbeiten konnten. Während des Reviews wurden die Gutachter nacheinander vom Moderator aufgerufen und gehört. In dem zeitgleich erstellten Protokoll wurden einzelne Befunde gewichtet, z. B. als kritischer Fehler, Hauptfehler, Nebenfehler sowie explizit als „gut". Letzteres ist aus psychologischer Sicht für die Autoren wichtig, denn der Programmausschnitt und nicht die Autoren stehen zur Diskussion. Gefundene Nebenfehler waren z. B. eine leere if-Anweisung und eine Exception, wenn ein Port nicht geöffnet ist.

Nach der Reviewsitzung wurde über die Methode reflektiert. Es wurde betont, dass niemand aus seiner Rolle gefallen ist. Als Alternative wurde ein abschnittsweises Vorgehen vorgeschlagen. Insbesondere das Konfliktpotenzial solcher Reviews wurde thematisiert: Zum Beispiel die Frage, ob und wann jemand aus der Geschäftsleitung als Verantwortlicher bei Reviews anwesend sein sollte, da immer die Gefahr von Profilierungsstreben und möglichen Auswirkungen auf das Gehalt der Mitarbeiter gegeben ist. Außerdem muss der Moderator darauf achten, dass unwichtige Details wie (Programmier-) Stilfragen nicht vom Wesentlichen ablenken.

Präsentation der Projektergebnisse

Am Ende der Projektphase präsentierten die Gruppen ihre Anwendung und das von ihnen erstellte Webquest im Plenum. Gruppe 1 (Rundenzeiten): Die Anwendung diente dazu, Rundenzeiten zu erfassen. In der Präsentation wurden drei Mitschülern RFID-Transponder für einen Lauf im Klassenraum zugewiesen. Nach jeder Runde mussten die Läufer ihren Transponder nah an das RFID-Lesegerät halten, wobei die Rundenzeiten

protokolliert wurden. Grenzen des Programms wurden thematisiert, z. B. eine Änderung der Reihenfolge der Läufer war noch nicht möglich. Besonderheit des Programms war die Nutzung des `DataGridView` mit der Thread-Synchronisation `Invoke`: Die `DataGridView.Invoke`-Methode führt einen Delegaten in dem Thread aus.

Gruppe 2 (Türöffner): In einer Türzugangs-Verwaltung wurden Transpondern Mitarbeiter mit Rechten zugeordnet, und Türen wurden bei erfolgreicher Rechteprüfung geöffnet. Dafür wurde serverseitig eine portable MySQL-Datenbank genutzt (MoWeS portable II[4]). Außerdem wurde eine kleine Anwendung entwickelt, die mit dem RFID-Controller in Verbindung steht. Über ein potentialfreies Relais (einfacher Schalter) wird ein Türöffner mit eigener Stromversorgung angeschlossen. Besonderheit hinsichtlich Threads war der Einsatz des `BackgroundWorker`-Threads. Damit ist es beispielsweise möglich, in einer Registerkarte die Initialisierung zu starten, die bei einem Wechsel der Registerkarte im Hintergrund weiter laufen kann, ohne das Programm zu blockieren.

Gruppe 3 (Bibliothek): Für die erstellte Anwendung wurden Bücher mit RFID-Transpondern ausgestattet. Darauf sind Identifikationsnummer, Autor und Titel gespeichert. Über eine Datenbank konnte der Status der Bücher erfragt werden. Zur besseren Kompatibilität gegenüber weiteren RFID-Controllern sind die Verbindungsparameter der seriellen Schnittstelle über die Software einstellbar. Threads wurden von dieser Gruppe minimalistisch eingesetzt: Um zu verhindern, dass ein Thread anfängt mit Daten zu arbeiten, die von einem anderen Thread noch nicht vollständig über die serielle Schnittstelle gesendet wurden, wird der erste Thread „schlafen gelegt“ durch `Thread.Sleep()`. Hervorzuheben ist der Einsatz des Singleton-Entwurfsmusters für den Konstruktor ihrer `RS232`-Klasse.

Gruppe 4 (Zeiterfassung): Die Anwendung setzte eine Zeiterfassung um, in der Mitarbeitern Transponder zugeordnet wurden. Zu Arbeitsbeginn und -ende müssen die Mitarbeiter sich ein- bzw. ausstempeln. Zur Datenspeicherung wird XAMPP[5] mit entsprechender Datenbank genutzt. Besonderheit bei dieser Anwendung war ein umfangreiches Exception-Handling, mit dem differenzierte Rückmeldungen an den Nutzer gegeben werden, z. B. falls ein Transponder im System noch nicht bekannt ist. Außerdem wurde mittels Threads ein Fortschrittsbalken animiert, der (zeitlich) parallel zur Initialisierung der Transponder erscheint.

Gruppe 5 (Gesundheitskarte): Die Anwendung ermöglichte das Speichern von Gesundheitsdaten wie Blutgruppe, Impfungen und Allergien auf dem Transponder und in einer Datenbank. Besonderheit war ein umfangreicher Einsatz von Delegates und Eventhandlern, die in Threads zur Benutzereingabe ausgeführt werden, z. B. mit der `boxVaccinations.Invoke`-Methode beim Setzen der medizinischen Daten.

[4] http://www.softpedia.com/get/Internet/Servers/Server-Tools/MoWeS-Portable.shtml (Abruf: 17.04.2015)
[5] https://www.apachefriends.org/de/index.html (Abruf: 17.04.2015)

4 Resümee und Ausblick

Um zu prüfen, ob das Unterrichtsprojekt in der vorgestellten Weise für die Zielgruppe geeignet ist, wurde die erste Durchführung von den beteiligten Lehrkräften durch mehrere Methoden evaluiert: Einen kompetenzorientierten Vor- und Nachtest (Situational Judgement Test), Evaluation der Projektpräsentationen und Dokumentationen mit Bewertungsbogen nach Kassner [Ka09, S. 203f], Evaluation der Webquests mittels Kriterienkatalog sowie eine Evaluation der nicht-kognitiven Kompetenzen der Schüler mittels Akzeptanzfragebogen. Da projektorientierter Unterricht aktiv durch die Schüler gestaltet wird, wurden sie auch an der Beurteilung der Mitarbeit beteiligt: Die Bewertung der Präsentationen und der Webquests wurde von Lehrern und Schülern gemeinsam vorgenommen.

Die Schüler waren motiviert und aktiv, was wir Lehrkräfte auf die Handlungsorientierung [Gu01] zurückführen: Deren Merkmal „Zielgruppenorientierung“ war durch die Berücksichtigung der Vorkenntnisse und Bedarfe der Auszubildenden mit Fachrichtung Anwendungsentwicklung erfüllt. „Ganzheitlichkeit“ durch die selbständige Wahl einer praxisrelevanten RFID-Anwendung und die Vollständigkeit der Handlung von der Planung bis zur Präsentation. „Reflexion“ wurde durch Fragebögen, Einbezug in die Bewertung sowie regelmäßige Fixpunkte zum Arbeitsstand erreicht. Außerdem forderte die Projektarbeit „Selbständigkeit“ bei der Bewältigung der problemhaltigen Aufgabe.

Das Software-Review ist darüber hinaus als schüleraktivierende Methode positiv aufgefallen, da mit ihr strukturiert Programme analysiert und verbessert werden können. Die Webquests stärkten die Reflexion über die eigene Arbeit und haben die Dokumentation bereichert. Zur Evaluation der RFID-Webquests wurde ein Fragebogen in Anlehnung an Bescherers[6] Kriterien zur Evaluation von Webquests eingesetzt. Die Evaluation zeigt, dass ausgewählte Webquests in späteren Unterrichtsprojekten für den Einstieg in die RFID-Programmierung genutzt werden können. Dabei müssten die Webquests für andere Schülergruppen, wie Berufsfachschule III (Informatikassistenten) oder Fachoberschule, etwas reduziertere Aufgabenstellungen enthalten. Besonders gelungen sind die Hinweise, die die Schüler formuliert haben, z. B. auf Software zur Überwachung der seriellen Schnittstelle.

Die interessenbezogene Vernetzung mit anderen Themenbereichen nahmen die Schüler vor, insbesondere zum Thema Datenbanken und Webentwicklung, allerdings wären auch weitere fächerübergreifende Aspekte wünschenswert, z. B. zum Datenschutz.

Alternativ zum Ablauf des Projekts, in dem interessengeleitet Anwendungskontexte selbst gesucht wurden, ist es denkbar, das Projekt als (Programmier-) Wettbewerb durchzuführen. Da beide Ansätze die Motivation der Schüler fördern können, ist dies mit Blick auf die Schülergruppe abzuwägen. Nachteil eines Wettbewerbs ist, dass

[6] http://www.bescherer.de/webquests/kriterien.pdf (Abruf: 17.04.2015)

gemeinsame Fixpunkte wie die moderierte Problemlöserunde Daily Scrum und vor allem das in dieser Unterrichtskonzeption so wichtige Software-Review, bei dem eine Gruppe ihren Quellcode zur Verfügung stellt, kaum hinein passen. Und gerade diese Plenumssitzungen wurden von den Schülern positiv aufgenommen. Außerdem ermöglichen sie eine Angleichung der Wissensstände aller Schüler im Sinne des schulischen Projekts und in Abgrenzung zum Informatik-Projekt in der Wirtschaft.

Literaturverzeichnis

[Fr07] Frey, K.: Die Projektmethode (9. Auflage). Weinheim und Basel. Beltz Verlag, 2007.

[Fu07] Fuller et al: Developing a computer science-specific learning taxonomy. In: ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education. New York, NY, USA. ACM, 2007, S. 152-170.

[GI08] Gesellschaft für Informatik (Hrsg.): Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe 1. Beschluss des Präsidiums der Gesellschaft für Informatik e. V. vom 24. Januar 2008. In: LOG IN 28 (2008), Nr. 150/151. (Beilage zum Heft; 72 Seiten).

[Gu01] Gudjons, H.: Handlungsorientiert lehren und lernen. Schüleraktivierung. Selbsttätigkeit. Projektarbeit (6. überarb. und erw. Auflage). Bad Heilbrunn. Klinkhardt, 2001.

[HNR06] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer, Berlin, 2006.

[Jo06] Johlen, D.: Zugriff auf einen RFID-Scanner mit Java über die serielle Schnittstelle (RS232) zur Verfolgung von Waren in einer Logistikkette. Lehr-/ Lernarrangement im Berufsfeld : Elektrotechnik, Informationstechnik. Oskar-von-Miller-Schule Kassel. 2006.

[Ka09] Kassner, D.: Projektkompetenz. Durchführung von Projekten in Schule, Aus- und Weiterbildung (2., überarbeitete Auflage). Bildungshaus Schulbuchverlage Westermann Schroedel Diesterweg Schöningh Winklers GmbH, Braunschweig, 2009.

[Kl09] Kleuker, S.: Grundkurs Software-Engineering mit UML. Vieweg + Teubner. Wiesbaden. 2009.

[KMK97] Kultusministerkonferenz der Länder (KMK): Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik Fachinformatiker / Fachinformatikerin vom 10. Juli 1997 nebst Rahmenlehrplan, 1997.

[Ko11] Koubek, J.: Informatik im Kontext. RFID. 2011. URL: http://medienwissenschaft.uni-bayreuth.de/informatik-im-kontext/index.php/entwuerfe/rfid/ (Abruf: 17.04.2015).

[SL10] Spillner, A.; Linz. T.: Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester: Foundation Level nach ISTQB-Standard. dpunkt.verlag GmbH, Heidelberg 2010.

[SS11] Schubert, S.; Schwill, A.: Didaktik der Informatik (2. Auflage). Heidelberg, Berlin. Spektrum Akademischer Verlag, 2011.

Grafische Programmiersprachen im Abitur

Kerstin Strecker[1]

Abstract: In diesem Artikel wird von den Erfahrungen aus dem Unterricht in einem Informatikkurs berichtet, der im Hinblick auf das Zentralabitur in Niedersachsen mit einer grafischen Programmiersprache (BYOB) vorbereitet worden ist. Daneben werden die Unterrichtserfahrungen aus einem Abiturkurs gesetzt, der nur Java als Werkzeug verwendet hat und die Erfahrungen aus einem „Bilingual"-Abiturkurs, dem BYOB und Java gleichberechtigt nebeneinander als Sprachen zur Verfügung standen.

Keywords: graphische Programmiersprachen, Sekundarstufe 2, Abitur

1 Einleitung

Grafische Programmierumgebungen finden immer mehr Verbreitung vor allem in den unteren Jahrgängen. Dies schlägt sich in Niedersachsen auch im Kerncurriculum für die Jahrgänge der Sek 1 nieder, wo es ausdrücklich heißt (S.21): „[...] entwickeln und implementieren einen Algorithmus in einer grafischen Programmiersprache […]" [KC14].

Die Gründe für die weite Verbreitung grafischer Sprachen wie Scratch [Sc15] in den unteren Jahrgängen führe ich auf die schnelle und leichte Erlernbarkeit durch intuitive Bedienung und die anschauliche Umsetzbarkeit anwendungs- und kontextbezogener Problemlösungen, z.B. durch die Integration grafischer Elemente, wie Kostümwechsel von Objekten u.a., zurück. Insgesamt führt dies nach meinen Erfahrungen zur schnellen Umsetzbarkeit eigener (kontextbezogener) Schülerideen und damit insgesamt zu einer breiten Motivation der SchülerInnen.
Dieser Artikel berichtet von dem Versuch, diese positiven Erfahrungen in der Oberstufe bis zum Abitur fortzusetzen.

Wenn die SchülerInnen in der Sek 1 mit Hilfe grafischer Programmiersprachen wie Scratch unterrichtet und vielleicht auch stärker motiviert wurden, wie geht es dann weiter? Reicht eine grafische Programmiersprache auch für das Bestehen des Abiturs aus oder bestehen die SchülerInnen das Abitur vielleicht sogar besser, wenn sie mit einer grafischen Sprache wie BYOB darauf vorbereitet werden? Um sich einer Antwort auf diese Frage zu nähern, wurde ein Informatikkurs mit BYOB auf das Zentralabitur vorbereitet. Die Erfahrungen werden im Folgenden beschrieben. Zum Vergleich werden Unterrichtserfahrungen in einem Java-Kurs und einem „Bilingual"-Kurs herangezogen.

[1] Institut für Informatik, Goldschmidtstr. 7, 37077 Göttingen, kerstin.strecker@gmx.de

Zu BYOB und dem Umgang mit BYOB verweise ich auf die Artikel von Eckart Modrow [Mo11a], [Mo11b], [Mo12] und [By15]. Beispiele für kontextbezogene Aufgaben mit Scratch, bzw. BYOB finden sich z.B. in [St14] und [St15].

2 Randbedingungen

In diesem Artikel werden drei Kurse miteinander verglichen, die im Mittel gleich viele Schüler hatten und dabei die Größe eines durchschnittlichen Oberstufenkurses. Dem Java-Kurs stand ausschließlich Java als Werkzeug zur Verfügung, der BYOB-Kurs wurde mit BYOB auf das Abitur vorbereitet und der „Bilingual"-Kurs hat sich von Anfang an gleichberechtigt mit den beiden Sprachen BYOB und Java beschäftigt. Die SchülerInnen dieses Kurses durften die Sprache jederzeit frei wählen, z.B. auch in der Klausur von Aufgabe zu Aufgabe unterschiedlich.
Da es in Niedersachsen noch kein Kerncurriculum für die Oberstufe gibt, orientiert sich der Unterricht an sogenannten Themenschwerpunkten, die jährlich etwas variieren und die Schwerpunkte des schriftlichen Zentralabiturs festlegen [NB15]. Neben Algorithmik finden sich dabei auch Themenfelder wie Technik, Datenbanken und formale Sprachen o.ä., die hier nicht betrachtet werden. Alle Klausuren inkl. Abitur werden handschriftlich geschrieben.
Die Ergebnisse der Abiture, Materialien und Aufzeichnungen über den gesamten Unterricht liegen vor und bestätigen die im Folgenden geschilderten Erfahrungen. Aufgrund der geringen Menge der SchülerInnen (insgesamt etwas über 50) werden die Ergebnisse in Kapitel 4 jedoch nur in Ansätzen statistisch ausgewertet.

3 Erfahrungsbericht

Im Java-Kurs wurden Aufgabenstellungen zur Implementierung von (eigenen) Algorithmen in Klausuren oft entweder (fast) korrekt bearbeitet oder erst gar nicht angefangen. Teillösungen fanden sich nur in ganz geringem Umfang.
Im BYOB-Kurs gab es bei ähnlichen Aufgabenstellungen in Klausuren genauso viele (fast) korrekte Lösungen, aber es konnten zusätzlich viel öfter Teilpunkte vergeben werden, weil mindestens Teillösungen der Implementierungsaufgabe vorhanden waren. Dabei unterschieden sich die algorithmischen Fähigkeiten der SchülerInnen nicht so auffallend, wie bei diesem Ergebnis zu vermuten wäre.
Aus der Beobachtung im Unterricht lassen sich dafür zwei Erklärungen dazu finden:

In jeder Lerngruppe gibt es SchülerInnen, die weniger aktiv am Unterricht teilnehmen und sich in Gruppen oder Partnerarbeit eher zurückziehen. Während sich solche SchülerInnen im Java-Kurs vermehrt in die Rolle des eher passiven Nachvollziehens der Lösungen ihrer Mitschülerinnen begeben haben, haben im BYOB-Kurs diese Schüler vermehrt aktiv an eigenen Lösungen gearbeitet. Durch die Möglichkeit, stets lauffähige

Produkte zu erzeugen, hat BYOB die SchülerInnen dazu motiviert, mit der Problemlösung überhaupt zu beginnen und so die Erfahrung vermittelt, dass eine Bearbeitung der Problemstellung nicht hoffnungslos ist. So konnten diese SchülerInnen im BYOB-Kurs später in Klausuren in viel stärkerem Maße auf eigene Erfahrungen zurückgreifen und diese Erfahrungen bei der Bearbeitung der neuen Aufgabe in der Klausur verwenden.

Eine weitere Erklärung für das Phänomen der Teillösungen sehe ich in dem Programmierverhalten in der Unterrichtszeit vor dem Abitur an sich. In Java programmieren die Schüler immer größere Blöcke auf einmal und testen diese dann, eventuell mit dem Debugger oder auch mit Testausgaben. Auch wenn die Blöcke nur wenige Zeilen umfassen, müssen sie zuvor vollständig durchdacht werden. In BYOB nutzen viele Schüler die Möglichkeit den Code stückweise zu entwickeln und dabei *gleichzeitig* zu testen. In dem laufenden System wird in BYOB der Zustand nach Ausführung eines Befehls oder Teilalgorithmus eingefroren, die Variablen z.B. behalten ihren Wert. Einzelne Befehle können so per Mausklick vor dem Einbauen in das Programm auf logische Korrektheit überprüft werden. Das Programm kann gleichzeitig Stück für Stück entwickelt und laufengelassen werden. Diese Arbeitsweise führt dann offenbar auch zu Teillösungen in Klausuren.

Zusammenfassend lassen sich meine Erfahrungen damit beschreiben, dass **die Verwendung von BYOB den Lernenden fast immer ermöglicht, wenigstens Teile der Lösung zu finden.**

Zwar hatten alle SchülerInnen Vorerfahrung im Programmieren, doch mussten einige Konzepte (Umgang mit Reihungen/Listen, eigene Methoden, OOP…) noch unterrichtet werden. Meine Aufzeichnungen zeigen hier eine Zeitersparnis im BYOB-Kurs im Vergleich zum Java-Kurs, weshalb neben anwendungsorientierten Aufgaben (s.u.) auch mehr Zeit für andere Themenbereiche (vgl. [NB15]) blieb, die dadurch etwas intensiver bearbeitet werden konnten und so vermutlich auch zu geringfügig besseren Ergebnissen im Abitur führten (vgl. Kapitel 4).

Ein oft gehörtes Argument von Lehrerseite gegen die Verwendung von BYOB ist die Schwierigkeit des Aufschreibens von BYOB-Programmen. Zwar malten die SchülerInnen in den Klausuren keine Blöcke, sondern notierten die Befehle im Klartext und markierten am Rand (der BYOB-Syntax nachempfunden) Beginn und Ende einer Schleife oder Auswahl. Allerdings wird dies durch das ausschließliche Hineinziehen der Befehle in den Arbeitsbereich mit der Maus während des Unterrichts nicht trainiert und muss also gesondert geübt werden. In Java neigen die SchülerInnen eher dazu, die Befehle selbst einzutippen, auch wenn einige Editoren bereits die Hilfe anbieten, mit einem Klick bestimmte Programmgerüste zu erzeugen. Hier fällt das Aufschreiben leichter. Auch fällt mir die Korrektur von Java-Programmen deutlich leichter als die handschriftlicher BYOB-Programme. Hier könnte natürlich Abhilfe geschaffen werden, indem das Abitur und Klausuren an Schulen, die über eine entsprechende Infrastruktur

verfügen, auch in Teilen am Rechner bearbeitet wird.

Nicht verschwiegen werden soll, dass in einem reinen BYOB-Abiturkurs aber auch einige SchülerInnen unzufrieden mit der Auswahl der Sprache sind. Die Argumente, warum sie ungern mit BYOB arbeiten, sind eher emotionaler Natur, z.B. „Java ist näher an der Maschine“ oder „näher an der echten Informatik“ (vgl. dazu auch [Mo11b]). Interessanterweise schätzten diese SchülerInnen ein Java-Programm hoch ein, das zwar ein beeindruckendes Ergebnis lieferte, intern jedoch fast ausnahmslos aus dem Aufruf von Methoden verschiedener Java-Bibliotheken bestand und algorithmisch wenig anspruchsvoll war.
Die Akzeptanz von BYOB im Bilingual-Kurs war dagegen überhaupt kein Thema, die obigen Aussagen kamen von diesen SchülerInnen nicht.
Eine Erklärungsmöglichkeit ist folgende: Im Bilingual-Kurs wurde von Anfang an stets jedes Konzept in beiden Sprachen nebeneinander demonstriert und so immer wieder indirekt darauf hingewiesen, dass es einen Unterschied gibt zwischen dem Konzept an sich und der Umsetzung in einer beliebigen Sprache. Da BYOB und Java an vielen Stellen eine unterschiedliche Syntax besitzen, aber nebeneinander verwendet wurden, wurde deutlich, dass sprachspezifische Details oder die „Schreibweise“ eine andere Kategorie hatten als das algorithmische Problemlösen oder die in algorithmischen Grundstrukturen an sich. Dies konnte an den Nachfragen und Aussagen der Schülerinnen festgemacht werden, die anders als in anderen Kursen Fragen stellten wie: „wie mache ich Vererbung noch mal in Java?“

Meine Erfahrung lässt sich folgendermaßen zusammenfassen: **Im Unterricht ist im Großen und Ganzen ein Zusammenhang festzustellen zwischen der Akzeptanz von BYOB und der Trennung zwischen Konzept und Sprache.**

Durch die Trennung von Konzept und Umsetzung zeigte sich, dass der Großteil der SchülerInnen des Bilingual-Kurses reflektiert die Sprache nach Art der Aufgabe gewählt und bei Bedarf gewechselt hat. Während BYOB in Aufgaben, in denen es um mathematische Rechnungen ging, sehr umständlich ist und hier von mehreren SchülerInnen Java gewählt wurde, wählten die SchülerInnen z.B. zur Erzeugung der Kochkurve alle BYOB. Dies Verhalten zeigte sich übrigens auch in den Klausuren des Bilingual-Kurses, in denen teilweise aufgabenweise die Sprache gewechselt wurde.
Die Fähigkeit zur Unterscheidung zwischen Konzept und sprachlicher Umsetzung habe ich vor dem Unterricht im Bilingual-Kurs nie so intensiv wahrgenommen, und ich werte diese Kompetenz sehr hoch, auch für den weiteren Lebensweg der SchülerInnen, da sich Sprachen schnell verändern, Konzepte seltener.
Die Kompetenz, zwischen Konzept und Umsetzung zu unterscheiden und aufgabenspezifisch die geeignetere Sprache zu wählen, ist meiner Meinung nach mit zwei textbasierten Sprachen in dieser Form nicht so einfach zu erreichen, da allein das Erlernen der Sprachen zu viel Zeit in Anspruch nehmen und die SchülerInnen wahrscheinlich mit der Syntax durcheinander kommen würden.

Die rückblickende Durchsicht der Kurshefte zeigte für mich überraschend, dass meine Aufgabenstellungen im BYOB-Kurs und Java-Kurs unterschiedlich waren. Beispielsweise sehen die thematischen Schwerpunkte [NB15] die Implementation eines Stapels vor. Während ich im Java-Kurs den Anwendungsbezug eher vernachlässigt habe, weil die Implementation eines dynamischen Stapels sehr viel Zeit in Anspruch genommen hat, habe ich im BYOB-Kurs mit Hilfe der dynamischen Liste den Stapel nur nebenbei als Bestandteil des Lindenmayersystems eingeführt. Durch die grafischen Elemente (hier die Zeichenbefehle) konnte dies schnell umgesetzt werden und die Zeit, die vorher für die Java-Implementation erforderlich war, wurde im BYOB-Kurs für die Erzeugung künstlicher Pflanzen genutzt.

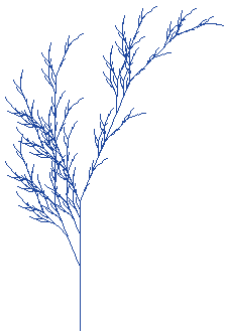


Abb. 1: Erzeugung künstlicher Pflanzen

Während im Java-Kurs viele klassische Algorithmikaufgaben behandelt worden sind, habe ich in BYOB durch die integrierten grafischen Elemente (Kostüme, Kostümwechsel, und das Hinterlassen von Abdrücken in diesem Kostüm…) eher Anwendungs- und kontextbezogene Aufgaben gewählt, beispielsweise die Generierung eines EAN-Strichcodes mit Prüfsumme im Themengebiet „Codierung und fehlererkennende Codes“ in BYOB statt dem Anhängen eines Paritäts“bits“ an eine Zeichenkette aus 0en und 1en in Java.
Dieselbe Erfahrung zeigte sich auch im Wahlverhalten der SchülerInnen des Bilingual-Kurses. Trotz ausgewogener Anteile bei der Wahl von Java oder BYOB im Unterricht zuvor, haben in einer Phase, wo eigene Projekte umgesetzt werden konnten, die meisten SchülerInnen ihre Ideen in BYOB umgesetzt, sei es die Programmierung von Geschicklichkeitsspielen oder eine physikalische Simulation, weil insgesamt die Visualisierung hoch gewertet wurde.

Meine Erfarungen zeigen also, dass sich **anwendungsorientierte Aufgaben mit BYOB oft leichter umsetzen lassen.**

Als Konsequenz müssten sich dann auch Klausuraufgaben verändern, da sich „alte“ Aufgaben schwer wiederverwenden lassen: Ich hatte für alle Kurse dieselbe Aufgabe ausgewählt, in der eine leicht schräge Kugelrinne beschrieben wurde, in die zufällig Kugeln verschiedener Farben geworfen werden. Liegen drei Kugeln derselben Farbe nebeneinander, darf der Werfer sie entfernen und bekommt einen Punkt. Die Kugelrinne wurde mit einer Reihung von Zeichenketten (als Farben) beschrieben. Es galt, alle vorhandenen Dreiergruppen zu entfernen.
Während sich alle SchülerInnen im Java-Kurs darauf konzentrierten, dass eine Reihung durchlaufen werden musste und Zeichenketten auf eine bestimmte Eigenschaft hin untersucht werden mussten, gab es im BYOB-Kurs auch SchülerInnen, die damit starteten, die Kugelrinne und den Wurf einer Kugel zu simulieren bzw. zu implementieren, um diese zunächst zu füllen.

Das oben beschriebene Beispiel der Lindenmayersysteme zeigt auch, wie im BYOB-Kurs mit Objekten umgegangen worden ist.
Zunächst bestand das Programm zur Erzeugung künstlicher Pflanzen aus mehreren nacheinander ausgeführten eigenen Methoden zur iterativen Berechnung der Ableitungen und dem Zeichnen der Pflanzen. Die Funktionalität des Stapels wurde mit einer Liste umgesetzt, auf der jeweils wieder als Liste x-Koordinate, y-Koordinate und Richtung abgelegt wurden. Im zweiten Schritt wurde ein Stapelobjekt erzeugt u.a. mit den klassischen Methoden pop und push und einer internen Liste. Dieser Stapel wurde nun in das „Hauptprogramm“ eingebunden und sogar in späteren Projekten wiederverwendet. Anschließend wurde ein weiteres Objekt „Koordinate“ erstellt, das eigene Variablen für x-Koordinate, y-Koordinate und Richtung besaß. Dieses Objekt, der Prototyp für alle vorkommenden Koordinaten, konnte unabhängig von den anderen Objekten getestet werden. Da je nach Schachtelungstiefe vor Ausführung des Programms unbekannt ist, wie viele Koordinaten man zur Erzeugung künstlicher Pflanzen braucht, wurde dem Koordinatenobjekt eine weitere Methode hinzugefügt, die einen Klon dieses Objekts zurückgegeben hat. Auch dies wurde ins Hauptprogramm eingebunden, es wurden nur noch Koordinaten auf den Stapel gelegt und man konnte bei der Ausführung mitverfolgen, wie die Koordinaten dynamisch erzeugt wurden und ihre Individualität im Wortsinn „sichtbar“ wurde.
Hier konnte also der Weg vom konkreten Objekt zur Verallgemeinerung gegangen werden. In Java ist nur eine Generierung eigener Klassen möglich. Egal ob später nur ein Objekt erzeugt wird oder eine unbekannte Zahl von Objekten erzeugt werden, man muss den Weg über eine Klasse gehen, bevor Objekte erzeugt werden können. Didaktisch gesehen gefällt mir die Umsetzung des OOP-Konzepts in BYOB auf diese Weise etwas besser. Es kann in den Unterrichtsgang gut integriert werden, zumal man alle (dynamisch erzeugten) Objekte nach dem Programmablauf einzeln untersuchen kann.
Im Vergleich zum Klassenkonzept von Java können so auch wieder die Konzepte der OOP und die sprachspezifische Umsetzung getrennt und auch hier eine Vermischung vermieden werden.

Bei mir selbst habe ich außerdem beobachtet, dass ich durch die Möglichkeit des schnellen Zusammenschiebens von Teilalgorithmen Konzepte oft mit einer kurzen BYOB-Sequenz veranschaulicht, ausgeführt und damit geeigneter visualisiert, in einem reinen Java-Kurs hingegen in solchen Fällen oft nur Teilcode an die Tafel geschrieben habe.

4 Zusammenfassung und Auswertung meiner Erfahrungen

Grafische Programmiersprachen sind (meiner Meinung nach) eine echte Bereicherung für die Schule, auch in der Oberstufe. SchülerInnen, die mit BYOB auf das niedersächsische Zentralabitur vorbereitet wurden, haben dies im Mittel eher besser bestanden als reine Java-Kurse. Auch wenn kaum statistische Aussagen möglich sind, lässt sich sagen, dass sie den Aufgaben gewachsen waren und ihnen keine Konzepte oder Kompetenzen fehlten. Für weniger leistungsstarke SchülerInnen ist BYOB ein besonderer Gewinn, weil Teillösungen einfacher zu realisieren scheinen.
Im Detail konnten im Mittel im Java-Kurs 58% der Punkte im Teilbereich Algorithmik und 82% in allen anderen Teilbereichen des Abiturs erreicht werden.
Im BYOB-Kurs konnten im Mittel 77% der Punkte im Teilbereich Algorithmik des Abiturs erreicht werden und 85% der Punkte in allen anderen Teilbereichen. Für die Vorbereitung der anderen Teilbereiche stand im BYOB-Kurs (vgl. Kapitel 3) etwas mehr Zeit zur Verfügung. Es ergibt sich ein Vergleich wie folgt:

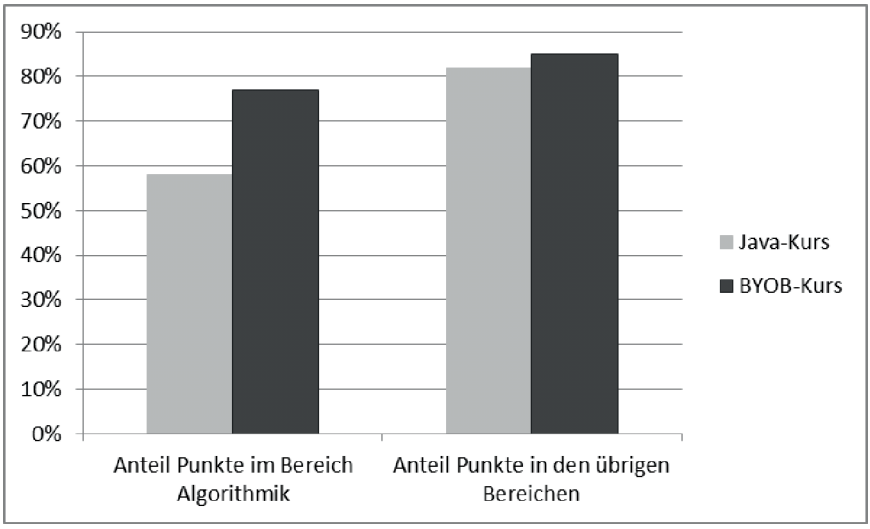


Abb. 2: Abiturergebnisse Java- und BYOB-Kurs

Die grafischen Elemente von BYOB verlieren bei der Umsetzung anwendungsorientierter Aufgaben auch in der Oberstufe nicht ihren Reiz. Weiterhin scheinen anwendungsbezogene Aufgaben auch für OberstufenschülerInnen motivierend zu sein, in eigenen Projekten wählen sie selbst anwendungsbezogene Beispiele und die scheinen sich besser mit BYOB umsetzen zu lassen.

Auch wenn Unterrichtende an textbasierten Sprachen festhalten möchten, ist der zusätzliche Einsatz von BYOB ein Gewinn für die SchülerInnen. Aus meinen Aufzeichnungen geht hervor, dass es kaum mehr Zeit erfordert, neben z.B. Java die Konzepte auch noch in BYOB darzustellen. Man gewinnt aber die Möglichkeit, den SchülerInnen den Unterschied zwischen Konzept und sprachspezifischer Umsetzung zu verdeutlichen. Die Kompetenz, sich aufgabenbezogen reflektiert für die in diesem Fall geeignetere Sprache zu entscheiden, kann kaum anders vermittelt werden.

Literaturverzeichnis

[By15] BYOB, URL: http://snap.berkeley.edu/old-byob.html, Zugriff: 26.1.2015.

[KC14] Niedersächsisches Kultusministerium: „Kerncurriculum für die Schulformen des Sekundarbereichs I. Schuljahrgänge 5-10. Informatik“, http://db2.nibis.de/1db/cuvo/datei/kc_informatik_sek_i.pdf, Zugriff: 26.1.2015.

[Mo11a] Modrow, Eckart; Mönig, Jens; Strecker, Kerstin: „Wozu Java? Plädoyer für grafisches Programmieren“, Zeitschrift LOG IN, Heft 168, LOG IN-Verlag, 2011.

[Mo11b] Modrow, Eckart: „Visuelle Programmierung – oder: Was lernt man aus Syntaxfehlern?“, in Marco Thomas (Hrsg.): „Informatik in Bildung und Beruf“, INFOS 2011, Lecture Notes in Informatics - Proceedings, Gesellschaft für Informatik, 2011.

[Mo12] Modrow, Eckart: „Objektorientiertes Programmieren mit BYOB“, Zeitschrift LOG IN, Heft Nr. 171, LOG IN-Verlag 2011/2012.

[NB15] Zentralabitur Niedersachsen, URL: http://www.nibis.de/nibis.php?menid=1395, Zugriff: 30.4.2015.

[Sc15] Scratch, URL: http://scratch.mit.edu/, Zugriff: 26.1.2015.

[St14] Strecker, Kerstin: „kontextbezogene Aufgaben“, Zeitschrift LOG IN, Heft Nr. 176/177, LOG IN-Verlag 2014.

[St15] Strecker, Kerstin: „Informatik konkret: 28 Anwendungsbeispiele. Lehrplanthemen erarbeiten – in Scratch programmieren – auf medizinische Kontexte anwenden“, AOL-Verlag, 2015.

Das Kreativlabor als generationsverbindendes Angebot im Bereich der praktischen Informatik

Katharina Weiß[1], Torben Volkmann[2] und Michael Herczeg[3]

Abstract: Dieser Beitrag präsentiert ein Konzept zur nachhaltigen Förderung des Interesses an der Informatik, wobei die Zielgruppen sowohl Schülerinnen und Schüler als auch Eltern und Großeltern darstellen. Es wird die Frage adressiert, wie das einmal geweckte Interesse an Informatik über den einzelnen Impuls hinaus aufrechterhalten und gefördert werden kann. Hier spielen Akzeptanz, Interesse und Unterstützung im familiären Umfeld eine wichtige Rolle. Ein Mehr-Generationen-Workshop ermutigt die Teilnehmenden, sich im Rahmen eines Kreativlabors gemeinsam an verschiedenen Stationen dem Themenfeld der praktischen Informatik anzunähern. Unterschiedliche gestalterische Methoden, wie Design Thinking und Prototyping treffen auf die Erprobung von Fertigkeiten, wie z.B. die Programmierung von Microcontrollern oder die Verarbeitung von Stoffen und leitfähigen Garnen für Wearables. Ziele des Kreativlabors sind zum einen eine motivierende Lernumgebung zu gestalten, innerhalb derer bereits erworbene Kompetenzen aller Generationen ineinandergreifen und zum anderen eine nachhaltiger gestützte Motivations- und Förderstruktur für eine weiterführende Auseinandersetzung mit dem Themenfeld Informatik zu schaffen.

Keywords: Praktische Informatik, Kreativ-Workshop, generationsübergreifend, nachhaltige Förderung

1 Einleitung

Frühzeitig Interesse wecken und Möglichkeiten bieten, individuelle Begabungen zu entdecken und auszubauen, sind wichtige Eckpfeiler für die Förderung von Jugendlichen im Bereich der Informatik und ihren Querschnittsbereichen [La10]. Im Rahmen der Schülerakademie der Universität zu Lübeck, bietet die LIaS-Initiative (Lübecker Informatik an Schulen) seit vielen Jahren ein breites Spektrum an Formaten vom motivierenden Schnupperkurs bis hin zur intensiven fachlichen Förderung von interessierten Jugendlichen. Inwieweit diese Motivations- und Förderstrukturen zur Aufnahme eines Studiums im Bereich der Informatik führen, scheint in einem großen Maße auch von der Unterstützung und Akzeptanz aus dem privaten und sozialen Umfeld der Jugendlichen abhängig zu sein [Pu05], [Pu11]. Für den Prozess der Berufsorientierung spielen Eltern nach wie vor als „Orientierungsinstanz“ ihrer Kinder

[1] Universität zu Lübeck, Institut für Multimediale und Interaktive Systeme (IMIS), Ratzeburger Allee 160, 23562 Lübeck, weiss@imis.uni-luebeck.de
[2] Universität zu Lübeck, Institut für Multimediale und Interaktive Systeme (IMIS), Ratzeburger Allee 160, 23562 Lübeck, volkmann@imis.uni-luebeck.de
[3] Universität zu Lübeck, Institut für Multimediale und Interaktive Systeme (IMIS), Ratzeburger Allee 160, 23562 Lübeck, herczeg@imis.uni-luebeck.de

eine wesentliche Rolle [AHQ10]. Hier stellt das weitverbreitete und auch falsche Bild der Trennung zwischen „Digital Natives“ und „Digital Immigrants“ [Pr11] eine ungünstige Voraussetzung für den Aufbau einer Motivations- und Förderungsstruktur im familiären Umfeld dar. Weniger Berührungsängste im Umgang mit neuen Medien auf Seiten der Jugendlichen führen auf Seiten der Eltern oder Großeltern oftmals zu einer automatischen Zuschreibung von Anwendungskompetenzen, die als Gegensatz zu eigenen Erfahrungswerten und Kompetenzbereichen wahrgenommen werden [AW13]. Ziel des Kreativlabors ist es, den so entstehenden Unsicherheiten und Vorurteilen auf beiden Seiten zu begegnen und in einer motivierenden und praxisorientierten Lernatmosphäre eine gemeinsame Basis für den generationenübergreifenden Dialog zu schaffen.

2 Workshop-Konzept: Kreativlabor

Das hier beschriebene Format des Kreativlabors basiert auf der Idee, den unterschiedlichen Vertretern der Zielgruppen an verschiedenen inhaltlich teilweise aufeinander aufbauenden Kreativ-Stationen einen Einblick in unterschiedliche Themenbereiche der praktischen Informatik zu ermöglichen.

2.1 Idee und Struktur des Workshops

Abb. 1: Workshop- Impressionen

Mit dem Kreativlabors wird ein Lernraum geschaffen, innerhalb dessen Fertigkeiten praktisch erworben und schon erworbene Fähigkeiten ausgebaut werden können und gleichzeitig die Gelegenheit besteht, sich über Studienangebote und fachliche Inhalte auszutauschen. Eltern oder Großeltern übernehmen während des gesamten Workshops eine unterstützende Rolle und erhalten durch die aktive Teilnahme einen Einblick in Methoden und Fragestellungen der Informatik, die nicht nur den Lebensalltag der Jugendlichen betreffen, sondern eine gesamtgesellschaftliche Herausforderung bilden. Ein besonderer Fokus bei der Konzeption des Workshops wurde auf die Gestaltung einer motivierenden Lernumgebung gelegt, die kreatives Arbeiten und Kommunikation

fördert. Die Projektarbeit an einzelnen Stationen spiegelt diesen Ansatz wider.

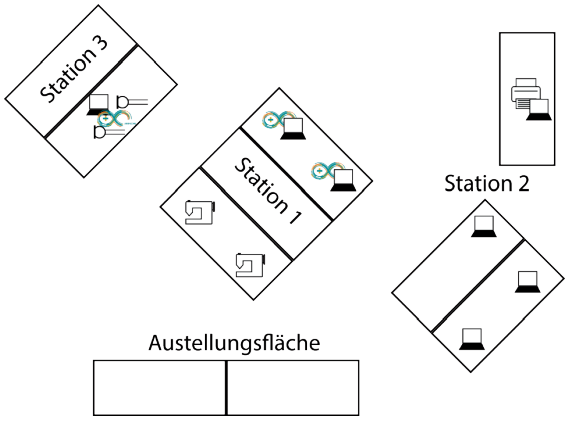


Abb. 2: Raumsituation im Kreativlabor

2.2 Stationen des Kreativlabors

Die einzelnen Lernstationen bauen inhaltlich aufeinander auf, lassen aber ebenso eigene interessengeleitete Lernpfade der einzelnen Teilnehmenden zu. In den bisher durchgeführten Workshops wurden drei verschiedene Stationen angeboten, die unterschiedliche Themenbereiche der Informatik in einer „begreifbaren“ Anwendung darstellen.

- Bei der Station „Papercraft“ geht es um die Transformation digital vorliegender Informationen in die reale Welt. Hierfür werden verschiedene dreidimensionale Modelle mit Hilfe des Programms Blender vorbereitet, die durch die Software von den Jugendlichen und deren Eltern oder Großeltern individualisiert werden können. Die kostenlos verfügbare und quelloffene Software Blender wird verwendet, da auch eine Nutzung außerhalb des universitären Kontextes möglich ist. Abhängig von dem Erfahrungslevel der Teilnehmenden besteht die Möglichkeit, Modelle vollständig individuell zu erstellen oder die vorweg erstellten Modelle durch die in dem Programm integrierten Modi farblich zu gestalten. Im Anschluss werden die Modelle mit einem herkömmlichen Drucker auf Papier gedruckt und können mit einer Schere ausgeschnitten und mit Klebestift zusammengesetzt werden. Um das “Papercrafting” in Blender zu integrieren, wird ein Plug-in eingebunden, das Falz- und Klebekanten vollautomatisiert erstellt. Auf die Benutzung von 3D-Druckern wird im Rahmen dieses Formats bewusst verzichtet, da diese verglichen mit herkömmlichen Druckverfahren um ein Vielfaches teurer und langsamer sind. Auch wurde der zeitliche Aufwand, die von den Teilnehmenden angefertigten Modelle für den 3D-Druck vorzubereiten und anzupassen im Vorwege als zu groß betrachtet.

Weiterhin wurde es als wichtig angesehen, dass die eigenen gefertigten Modelle für weitere Verwendungen in der zweiten Station eingesetzt werden können.

- Bei der Station „Monster und Musik“ haben die Teilnehmenden die Möglichkeit, vorher erstellte Modelle so anzupassen, dass mit ihnen Musik abgespielt werden kann. Hierzu wurde in der Vorbereitung ein Arduino-Microcomputer mit Hall-Sensoren ausgestattet und programmiert, sodass die Magnetfeldstärke von Magneten gemessen werden kann. Die von den Sensoren empfangenen Daten werden ausgewertet, werteabhängig einer bestimmten Tonhöhe zugeordnet und von Lautsprechern ausgegeben. Den Teilnehmenden wird die Möglichkeit gegeben, die vorher erstellten oder bereitgestellten Monster mit Magneten auszustatten und gemeinsam zu musizieren. Zudem dient diese Station dazu, langsam an die Microcontroller-Programmierung sowie an allgemeine Grundlagen der Schaltungstechnik heranzuführen. Die Arduino-Umgebung bietet die Programmierung in einer vereinfachten C-Syntax an. Im Gegensatz zu den gebräuchlichen Microcontroller-Programmiersprachen C und Assembler bietet die Arduino-Syntax gerade Anfängern einen leichten Einstieg in die Welt der Microcomputer. Außerdem finden sich in Online-Plattformen zahlreiche Tutorials und Foren, die sich mit dem Arduino-System beschäftigen.
- Bei der dritten Station durchlaufen die Teilnehmenden einen Design-Thinking-Prozess. Es werden vielfältige Materialien, wie Stoffe oder Knöpfe, zur Verfügung gestellt, die für die Generierung erster Ideen für eigene “Monster” genutzt werden können. Mit Hilfe von diversen elektronischen Bauteilen sollen die Monster mit Funktionen ausgestattet werden. Dazu werden mittels Lillypads verschiedene Sensoren und Aktuatoren, LEDs und leitfähiges Garn bereitgestellt. Vor dem eigentlichen Produktionsprozess werden konzeptionelle Fragestellungen zur Spezifizierung des Prototypen (Farbe, Form, Funktion) und zur Einschätzung der Realisierbarkeit des Vorhabens erörtert. Während der Umsetzungsphase wird das aufgestellte Konzept fortlaufend überprüft und gegebenenfalls an neue Anforderungen angepasst. Die Entstehung der Monster läuft in mehreren Etappen ab:
  - Auswahl von Formen, Farben und Funktionen des Monsters und Festhalten in eigenen Zeichnungen ggf. Schnittmustern;
  - Einführung in die Funktionsweise der Nähmaschinen und Vorbereitung der Zuschnitte;
  - Programmierung der Microcontroller und Zusammenführung mit dem Monstermodell;
  - Funktionstest des gesamten Monsters.

Ziel des Workshops ist es, einen gemeinsamen Design-Thinking-Prozess zu durchlaufen, der von der eigenen Idee hin zur Realisierung eines ersten Prototyps führt. Dabei helfen unterschiedliche Kompetenzen der Teilnehmenden (Programmierung der

Microcontroller, Umgang mit elektrotechnischen Bausteinen, Bedienen einer Nähmaschine) das selbst gesteckte Ziel mit unterschiedlichen Lösungsansätze zu erreichen und gleichzeitig Einblicke in neue, bisher unbekannte Themengebiete zu erhalten.

Abb. 3: Design Thinking und Prototyping im Kreativlabor

3 Erwartungen und Beobachtungen

Bei der Ausarbeitung des Konzeptes für das Kreativlabor als Mehrgenerationen-Workshop standen einige zentrale Überlegungen zur didaktischen Ausgestaltung im Mittelpunkt. Bei der Durchführung eines Workshops mit mehreren Generationen zu unterschiedlichen Themenbereichen der Informatik wurden als potentiell kritische Aspekte das Vorwissen und die Motivation der Teilnehmenden, das stimmige Verhältnis zwischen Dialog und Produktion innerhalb eines zeitlich begrenzten Rahmens sowie das tatsächliche Ineinandergreifen von unterschiedlichen Kompetenzen der Generationen zur Erreichung eines gemeinsamen Zieles betrachtet. Die Methode des Stationenlernens wurde eingesetzt, um eine Vielfalt möglicher Zugänge zu den Themengebieten der Informatik zu gewährleisten [Dö03].

3.1 Vorwissen und Motivation

Die Erfahrung aus der langjährigen Durchführung von Projekten und Veranstaltungen zur Förderung des Interesses an Informatik und ihren Querschnittsbereichen zeigt, dass das Vorwissen von Jugendlichen selbst innerhalb einer Altersstufe meist sehr unterschiedlich ist. Für die Durchführung eines Mehrgenerationen-Workshops wurde dieser Aspekt als noch entscheidender eingeschätzt. Hier galt es bereits im Vorfeld auf eine ausreichend flexible Aufgabenstellung zu achten, um sowohl die Teilnehmenden mit fortgeschrittenen Kenntnissen als auch Anfängern gerecht zu werden und die Motivation zur aktiven Teilnahme aufrecht zu erhalten. Aus diesem Grund wurde mit

der Methodik des Stationenlernens die Möglichkeit für die Lernenden geschaffen, ihren Lernweg entsprechend ihrer Interessen und Fähigkeiten selbst zu steuern. Während des Workshops zeichnete sich ab, dass sich die Teilnehmenden selbstständig Aufgaben annahmen, die ihren unterschiedlichen Kompetenzen entsprachen und sich gegenseitig bei der Ausarbeitung von Lösungsansätzen unterstützten.

3.2 Dialog und Produktion

Dass die Idee des Kreativlabors aufging, durch gemeinsame Projektarbeit in einem eher informellen Rahmen auch Kommunikationsräume zu eröffnen, zeigte sich an zahlreichen Diskussionen über Herausforderungen und Risiken, Chancen und Potentiale des Umgangs mit neuen interaktiven Medien im Alltag zwischen den Teilnehmenden auch unterschiedlicher Zielgruppen. Gesprächsanlässe wurden durch den Einstieg über die praktische und kollaborative Projektarbeit initiiert und führten zu einer offenen Gesprächsatmosphäre innerhalb derer viele fachspezifische Fragen zu informatischen Themen aber auch zum Studium und Berufsperspektiven aufkamen. Die aktive Teilnahme von Studierenden der Informatik an den einzelnen Stationen bot die Gelegenheit, am Geschehen unterstützend teilzunehmen und gleichzeitig eine beobachtende Rolle einzunehmen. Die im Vorfeld erwarteten wichtigen, vor allem auch kritischen Aspekte fanden hierbei besondere Berücksichtigung und wurden in einem abschließenden Gespräch gemeinsam ausgewertet.

3.3 Ineinandergreifen von Kompetenzen

Das Konzept des Workshops, den verschiedenen Generationen an Hands-On-Stationen Möglichkeiten zu bieten, sich mit eigenen Kompetenzen in den Gestaltungsprozess einzubringen, führte in vielen Situationen zu einer zuvor erwarteten Aufgabenteilung. Jugendliche zeigten wesentlich weniger Berührungsängste im Umgang mit digitalen Technologien [MFS12], während Eltern und Großeltern eher Arbeiten, wie die Bedienung der Nähmaschine oder auch das Zuschneiden von Schnittmustern übernahmen. Hier zeigten sich zunächst in vielen Situationen Unsicherheiten bezüglich der eigenen Kompetenzen und die daraus resultierende Hemmschwelle, eine neue Fertigkeit praktisch zu erproben. An der gemeinsamen Entwicklung einer Idee und der Anfertigung eines ersten Papierprototypens waren allerdings alle Generationen in gleichem Maße beteiligt. Das Produkt wurde damit zum gemeinsamen Erfolg.

4 Fazit und Ausblick

Das Kreativlabor-Format stellt nach unseren ersten Untersuchungen einen geeigneten und niedrigschwelligen Einstieg für eine vertiefende Förderung im Bereich der Informatik dar. Die Vertreter der Zielgruppen Schüler, Eltern und Großeltern erlebten und thematisierten im Rahmen des Workshops, welche Herausforderungen und

Potentiale digitale Technologien für verschiedene Generationen mit sich bringen. Die Atmosphäre und die Struktur des Workshops erwiesen sich als förderlich, um unterschiedliche Diskussionen zu initiieren. So reichen Dialoge über einzelne Themenbereiche der Informatik hinaus und stellen relevante Bezüge zu Alltagserfahrungen der unterschiedlichen Teilnehmenden dar. Das Kreativlabor wird dem im Vorfeld erhobenen Anspruch gerecht, keine reine Informationsveranstaltung für Eltern zu sein, sondern eine lernförderliche gemeinsame Arbeitsumgebung zu schaffen, die den Austausch von Eltern oder Großeltern und Jugendlichen „auf Augenhöhe" ermöglicht, während gemeinsam an informatischen Projekten gearbeitet wird. Für die Eltern und Großeltern stehen dabei gar nicht primär die Jugendlichen als „Projekt" im Fokus, sondern die gemeinsame Projektarbeit, die eine Basis für gegenseitiges Verständnis und Anerkennung bilden kann und somit eine gute Voraussetzung für eine nachhaltige Motivations- und Förderstruktur für informatische Systeme im familiären Rahmen bietet.

Literaturverzeichnis

[AHQ10] Albert, M.; Hurrelmann, K.; Quenzel, G.: Jugend 2010. Shell Jugendstudie, Hamburg, 2010.

[AW13] Arnold, P.; Weber, U.: Die „Netzgeneration". Emprirische Untersuchungen zur Mediennutzung bei Jugendlichen. Lehrbuch für Lernen und Lehren mit Technologien, 2. Auflage, 2013.

[Dö03] Dörig, R.: Handlungsorientierter Unterricht.Ansätze, Kritik und Neuorientierung aus bildungstheoretischer, curricularer und instruktionspsychologischer Perspektive , Stuttgart, Wiku-Verlag, 2003.

[La10] Lasen, M.: Education and career pathways in information communication technology: What are schoolgirls saying? Computers & Education, 54(4), 1117-1126, 2010.

[MFS12] Medienpädagogischer Forschungsverbund Südwest: JIM 2012: Jugend, Information, (Multi-)Media. Basisstudie zum Medienumgang 12-bis19-jähriger in Deutschland, Stuttgart, 2012.

[Pr11] Prensky, M.: Digital Natives, Digital Immigrants, in: On The Horizon, MCB University Press, 9(5), 2001.

[Pu05] Puhlmann, A.: Die Rolle der Eltern bei der Berufswahl ihrer Kinder, Bonn, 2005.

[Pu11] Puhlmann, A.: Berufsorientierung junger Frauen im Wandel. Abschlussbericht, Bonn, 2011.

Jens Gallenbacher (Hrsg.): INFOS 2015 - Informatik allgemeinbildend begreifen,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 309

Punkt, Punkt, Semikolon, Strich – Grafikorientierte Einführung in die Programmierung mit Processing

Daniel Wunderlich[1]

Abstract: Die Programmiersprache Processing ist darauf spezialisiert, einen möglichst einfachen Einstieg in die textuelle Programmierung zu bieten. Seit Beginn der Entwicklung 2001 am MIT Media Lab wird der Fokus hierbei auf die Erzeugung von Grafiken und Animationen gelegt. Processing erweitert hierzu Java um einfache Methoden zum Einbezug von Benutzereingaben durch Maus und Tastatur, wobei die Syntax und grundlegende Konzepte der Sprachen identisch sind. Gleichzeitig entfallen viele Aspekte der Java-Programmierung, die gerade beim Einstieg in die Programmierung von Lernenden nur schwer nachvollzogen werden können.

Der Workshop stellt einen Lehrgang zur Einführung in die Programmierung mit Processing in Sekundarstufe II vor, der im Rahmen des Referendariats mit ausführlicher Dokumentation entstand.

Keywords: Processing, Programmierung, Einführung, Grafik, Java, Sekundarstufe II

1 Programmierung in der Schule

Die Programmierung stellt, wie das Verfassen von Texten in sprachlichen Fächern und Naturgesetze in der Physik, ein zentrales Konzept der Informatik dar. Wichtig ist, dass sie im Kontext der schulischen Ausbildung insbesondere zu Beginn nicht als Produktschulung einzelner (meist unnötig komplexer) Entwicklungsumgebungen (IDEs – engl. *integrated development environment*) und im Hinblick auf programmiertechnische Finessen aufgefasst und gelehrt wird. Vielmehr sollten allgemeine und grundlegende Prinzipien der Informatik vermittelt werden [RNH01, S. 1 f.]. Zur Implementierung von Algorithmen ist sie außerdem Teil der *fundamentalen Ideen* der Informatik nach SCHWILL [Hu07, S. 82 ff.].

Gleichzeitig wird durch die Programmierfähigkeit die Kompetenz gefördert, die Arbeitsweise eines Computers zu verstehen – eine Grundlage zur „kritischen und verantwortungsvollen Nutzung von informationstechnischen Hilfsmitteln“ [Mi04, S. 438], wie sie z. B. im baden-württembergischen Bildungsplan 2004 gefordert wird. Diese Fähigkeit gewinnt gerade in einer zunehmend von Software geprägten Welt ständig an Relevanz.

Aufgrund ihrer Relevanz findet sich die Programmierung deshalb auch im Inhaltsbereich *Sprachen und Automaten* der Bildungsstandards der Gesellschaft für Informatik e. V. (GI) [Ge08] und der *2. Leitidee „Algorithmen und Daten“* des baden-württembergischen Bildungsplans 2004 wieder [Mi04, S. 439]. Nach letzterem sollen die SchülerInnen insbesondere „elementare Datentypen und Strukturen zur Ablaufsteuerung anwenden“ und „Algorithmen entwerfen und in Programme umsetzen“ können.

[1] Gymnasium Walldorf, Schwetzinger Str. 95, 69190 Walldorf, daniel.wunderlich@gymnasium-walldorf.de

2 Processing

Processing ist eine Programmiersprache mit zugehöriger IDE, mit deren Entwicklung 2001 von BEN FRY und CASEY REAS am *MIT Media Lab* (Cambridge, USA) begonnen wurde [FR]. Processing ist freie Open-Source-Software, sodass sich die Entwicklergemeinde stetig vergrößert. Die Programmiersprache wurde von Beginn an mit der Absicht konzipiert, in der Tradition von LOGO [Lo] und BASIC [Ho] als „erste Programmiersprache" einen einfachen Einstieg in die Programmierung zu ermöglichen. Gleichzeitig weist sie viele Parallelen zu gegenwärtig populären IDEs für den schulischen Einsatz wie *Scratch* [MI], *Kara* [Sw], *Greenfoot* [Pr] und *BlueJ* [LP] auf. Der Schwerpunkt von Processing liegt hierbei auf der Erzeugung von interaktiven, animierten Grafiken. Die erste bedeutende Version 1.0 erschien nach siebeneinhalbjähriger Entwicklung im November 2008 [Wi]. Mit Version 2.0 wurde 2012 eine umfassende Überarbeitung der Sprache veröffentlicht. Aktuell ist Version 2.2.1 (Stand: April 2015).

Heute erfreut sich Processing großer Beliebtheit. Eine aktive Community von Entwicklern, Künstlern und Wissenschaftlern verwendet die Programmiersprache für diverse Zwecke. Darüber hinaus steht mittlerweile eine Vielzahl von Erweiterungen zur Verfügung, z. B. Schnittstellen zu Arduino, JavaScript, Python, Microsoft Kinect oder OpenCV [FR].

Processing erweitert Java um diverse Funktionen [Sh08], z. B. das Zeichnen von Grafikprimitiven[2], die Interaktion mit Maus und Tastatur, das Verarbeiten und Anzeigen von Texten, Bildern und Videos und 2D-/3D-Transformationen. Syntax und zentrale Sprachelemente (insb. Variablen, Kontrollstrukturen, Methoden und Objektorientierung) sind mit Java identisch und auch fortgeschrittene Programmiertechniken (z. B. Objektorientierung) können angewandt werden. Beim Starten eines Programms wird der Processing-Code intern zuerst in Java-Code übersetzt und anschließend kompiliert. Dies führt dazu, dass man theoretisch sämtliche Processing-Programme problemlos in „reines" Java übersetzen kann.

Die bereits vorhandene Funktionalität ist auf die Gestaltung von animierten und interaktiven Medien ausgelegt. Dies zeigt sich deutlich im *Processing-Prinzip* (s. Abb. 1): Die Benutzereingabe erfolgt nicht wie in „traditionellen Einstiegsprogrammen" durch Zeicheneingabe, sondern ähnlich den oben aufgeführten IDEs über Maus und/oder Tastatur. Auch die Ausgabe erfolgt nicht textuell, sondern durch eine Animation der Grafik. Klickt man im gezeigten Beispiel z. B. in den hellen Kreis, verändert sich die Hintergrundfarbe von schwarz zu weiß.

Die schlichte Processing-IDE (auch PDE – engl. *processing developement environment*) ist in Abb. 2 zu sehen. Im weißen Bereich wird der Code verfasst, hierbei farblich hervorgehoben und automatisch eingerückt. Über die Schaltfläche ▶ wird das Programm kompiliert und ausgeführt, über ■ gestoppt. Fehler- und Statusmeldungen werden in einer Statuszeile ausgegeben, eine Konsole zur Textausgabe steht zur Verfügung.

Programme in Processing werden – in Anlehnung an den künstlerischen Ursprung – *Skizzen* (engl. *sketch*) genannt. Dieser Begriff wird sowohl für die Quelltextdatei als auch für

[2] Als *grafische Primitiven* bezeichnet man in der Computergrafik elementare geometrische Formen. Meist zählen hierzu Punkte, Strecken, Polygone, Kreise und Ellipsen.

Traditionell

1. *Programmierung*: Textuell
2. *Eingabe*: Textuell

```
> java Eingabebeispiel
> Wie heißt du? Horst_
```

3. *Ausgabe*: Textuell

```
> Hallo Horst!
```

Processing

1. *Programmierung*: Textuell
2. *Eingabe*: Durch Maus/Tastatur

3. *Ausgabe*: Visuell – Animation

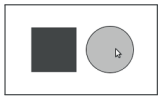
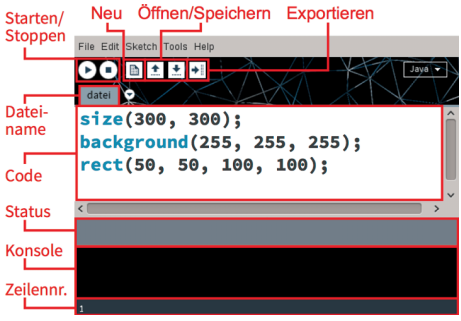
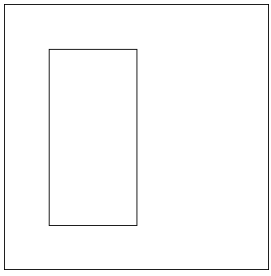


Abb. 1: Benutzerinteraktion traditionell und in Processing



(a) IDE mit Programm (Skizze)



(b) Resultierende Grafik

Abb. 2: Processing IDE mit erstem Programm

die resultierende Grafik verwendet. Im Folgenden werden die Begriffe *Programm* und *Skizze* synonym verwendet. Bevor eine Skizze gestartet werden kann, muss sie gespeichert werden. Für jede Skizze legt Processing hierzu ein Verzeichnis an, in dem der Code in einer Textdatei (Endung `*.pde`) abgelegt wird.

3 Motivation

ARNULF MESTER verwendete Processing bereits vor mehreren Jahren in der Lehre der DHBW Mosbach und stellte die Programmiersprache am Informatiktag 2010 der Fachgruppe der *Informatiklehrerinnen und -lehrer in Baden-Württemberg* der GI in einem Workshop vor [Me10]. Nach einer Einarbeitungsphase im Rahmen des Referendariats waren es insbesondere die folgenden drei Aspekte, welche Processing für den schulischen Einsatz besonders attraktiv erscheinen ließen:

1. *Processing ist einfach zu erlernen*. Dies betrifft insbesondere den Einstieg in die Programmierung. Selbst SchülerInnen ohne Programmiervorkenntnisse erleben relativ schnell Erfolge und werden nicht durch unverständliche Code-Fragmente abgeschreckt. Aufgrund der minimierten Menge an Code reduzieren sich syntaktische Fehler und auch die IDE überzeugt durch ihren Minimalismus.

2. *Processing ist ansprechend*. Bereits in der dritten Doppelstunde sind SchülerInnen in der Lage, Benutzereingaben zu verarbeiten und hierdurch interaktive dynamische Skizzen zu erstellen. Nach der vierten/fünften Doppelstunde können SchülerInnen einfache Spiele erstellen. Berücksichtigt man den gegenwärtigen Boom der Videospielbranche, ist deshalb mit hoher Motivation der SchülerInnen zu rechnen, sie werden in ihrer Lebenswelt abgeholt.

3. *Processing ist anders*. Zum einen entfallen die zu Beginn typischen auf Textausgabe abzielenden Aufgabenstellungen traditioneller Programmierlehrgänge („Gib die Quadratzahlen von 1 bis 20 aus.“), welche häufig mathematische Probleme lösen und für SchülerInnen meistens unattraktiv sind. Zum anderen fördert und fordert Processing Kreativität.

4 Unterrichtsform

Selbstständiger und handlungsorientierter Unterricht stellt offensichtlich eine Notwendigkeit beim Erlernen des Programmierens dar, weshalb die Unterrichtseinheit diesen Ansatz verfolgt und hierbei auf die Definition von HILBERT MEYER zurückgreift: „Handlungsorientierter Unterricht ist ein ganzheitlicher und schüleraktiver Unterricht.“ [Me87]

Die Doppelstunden der Unterrichtseinheit wurden deshalb grundlegend in die folgenden vier Phasen gegliedert. Damit der zeitliche und inhaltliche Umfang der jeweiligen Phasen nicht zu groß wird, wurden die Doppelstunden mehrmals in diese vier Phasen geteilt.

1. *Einstieg*: Die SchülerInnen bekommen die Notwendigkeit des neuen Unterrichtsinhaltes als Motivation aufgezeigt. Wenn möglich, wird dies durch ein kurzes passendes Spiel in Kleingruppen ermöglicht. Außerdem wird als zusätzliche Motivation ein Processing-Programm demonstriert, das nach der Doppelstunde (theoretisch) von den SchülerInnen programmiert werden könnte.

2. *Klassenunterricht*: Grundlegende Aspekte des Themas werden gelenkt durch den Lehrer in der Klasse erarbeitet. Die Beteiligung der SchülerInnen richtet sich hierbei nach dem Inhalt und die Dichte an neuen Informationen ist verhältnismäßig hoch.

3. *Einzelarbeit*: Die SchülerInnen üben die behandelten Inhalte und erarbeiten anschließend neue Aspekte. Dies geschieht durch Aufgaben auf einem Arbeitsblatt. Wenn möglich, werden Inhalte ohne Einsatz des Rechners gelöst. Diese Phase stellt den Schwerpunkt des Unterrichts dar.

4. *Sicherung*: Die Ergebnisse der Einzelarbeit werden gemeinsam besprochen und/oder demonstriert. Ggf. aufgekommene Fragen werden geklärt.

5 Vorhandenes Material und Literatur

Zu Beginn der Unterrichtseinheit konnten keine Materialien ausfindig gemacht werden, die nicht ohne intensives Aufbereiten einen für die Kursstufe angemessenen Programmiereinstieg ermöglicht hätten. Mittlerweile sind mit *Creative Coding* von TILL NAGEL [Na] und *Processing – Programmieren lernen* von MICHAEL KIPP [Ki] zwei vielversprechende Einführungskurse online verfügbar.

Die Auswahl an passender Literatur für Processing hält sich ebenfalls in Grenzen. In deutscher Sprache sind gegenwärtig lediglich zwei Bücher verfügbar: Das sehr kompakte *Processing – eine Einführung in die Programmierung* von ANDRES WANNER [Wa11] eignet sich eher zum schnellen Nachschlagen. Sehr viel ausführlicher, ansprechend gestaltet und empfehlenswert ist *Processing* von ERIK BARTMANN [Ba10]. Als Grundlage der Unterrichtseinheit diente dennoch ein englischsprachiges Werk: *Learning Processing* von DANIEL SHIFFMAN [Sh08]. Zum einen verfolgt SHIFFMAN konsequent das erläuterte Processing-Prinzip, indem bereits sehr früh Benutzerinteraktion durch Maus und Tastatur thematisiert wird. Diese Eingabemethoden werden in den folgenden Themen verwendet und z. B. Verzweigungen durch Farbänderungen aufgrund der Mausposition behandelt. Zum anderen beinhaltet das Buch viele Übungsaufgaben. Diese wurden zwar nur selten direkt übernommen, inspirierten jedoch sehr.

6 Folien und Arbeitsblätter

Der Klassenunterricht wird durch Folien unterstützt. Diese dienen insbesondere als Schulbuchersatz. Die Folien beinhalten zentrale Punkte des jeweiligen Themas, deren eigenständige Erarbeitung nur bedingt sinnvoll ist. Häufig werden Sachverhalte hierbei durch Grafiken veranschaulicht. Um den Klassenunterricht möglichst knapp halten zu können, werden viele Inhalte auf Arbeitsblätter ausgelagert und eigenständig erarbeitet.

Die Einzelarbeitsphasen werden durch Arbeitsblätter organisiert. Alle Teile beginnen mit einer einfachen Einstiegsaufgabe, in der z. B. ein Programm beschrieben, vorgegebene Änderungen am Code vorgenommen oder Code vom Blatt übertragen werden muss. Der Anspruch der folgenden Aufgaben steigt sukzessive: Werden zuerst noch einzelne Schritte vorgegeben, ist in späteren Aufgaben lediglich die zu implementierende Funktionalität beschrieben (ergebnisorientiert). Beendet werden die Einzelarbeitsphasen durch Zusatzaufgaben zur Differenzierung für schnelle SchülerInnen.

7 Themenverteilungsplan und Lernziele

Der vorgesehene Themenverteilungsplan ist in Tab. 1 dargestellt. Der Reihenfolge in [Sh08] folgend, wird den SchülerInnen bereits in der dritten Doppelstunde ermöglicht, Benutzereingaben in Programmen zu verarbeiten. Die Aufgaben in folgenden Doppelstunden greifen ausnahmslos hierauf zurück.

Die Unterrichtseinheit verfolgt folgende übergeordneten Lernziele:

Stunde(n)	Thema
1/2	Grafiken und Farben
3/4	Erste Schritte in Processing
5/6	Interaktivität in Processing
7/8	Variablen
9/10	Verzweigungen
11/12	Schleifen
13/14	Methoden

Tab. 1: Themenverteilungsplan der Unterrichtseinheit

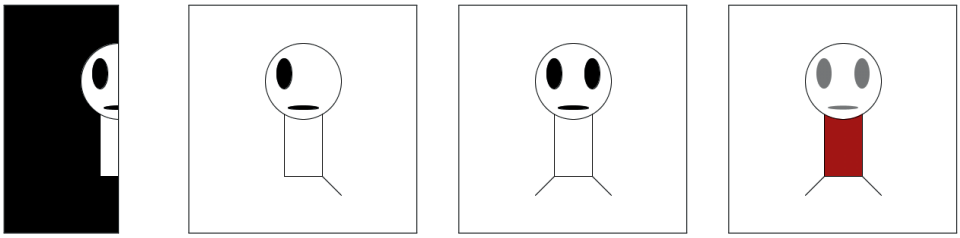
- Die SchülerInnen lernen einen zentralen Bereich der Informatik kennen.
- Die SchülerInnen erarbeiten angeleitet aber eigenständig neue Inhalte.
- Die SchülerInnen erstellen erste einfache Programme.
- Die SchülerInnen realisieren vorgegebene Sachverhalte in Programmen.
- Die SchülerInnen verbalisieren informatische und algorithmische Gegebenheiten.

8 Exemplarische Unterrichtsinhalte

8.1 Erste Schritte in Processing – Hallo Walli!

In der zweiten Doppelstunde machen die SchülerInnen zum ersten Mal Bekanntschaft mit *Walli*. Angelehnt an *Zoog* aus [Sh08], ist Walli eine einfache, achsensymmetrische Figur, welche durch die wichtigsten grafischen Primitive in Processing erzeugt werden kann. Die SchülerInnen erhalten auf dem Arbeitsblatt Quelltext, den sie in Processing übertragen sollen und der Abb. 3a erzeugt. So geben sie ihre ersten Processing-Befehle ein und haben ggf. gleichzeitig die Möglichkeit, syntaktische Fehler zu korrigieren, welche hierbei auftreten. Angeleitet sollen nun zuerst die Fenstergröße und anschließend die Hintergrundfarbe durch vorgegebene Änderung eines Parameters angepasst werden. Letzteres geschieht experimentell – die SchülerInnen sollen sich hierbei an die RGB-Spezifikation von Farben aus der vorangegangenen Doppelstunde erinnern und diese auf einen Grauwert übertragen. Abb. 3b zeigt das Ergebnis dieser Aufgabe.

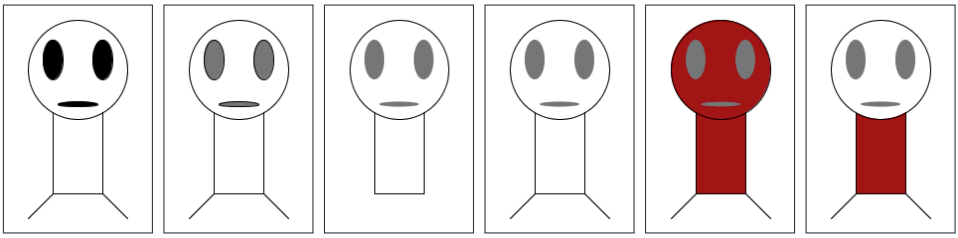
In der folgenden Aufgabe geht es darum, die ersten grafischen Primitive selbst zu programmieren. Walli soll zuerst mit einem zweiten Auge versehen werden. Die Parameter der Methode `ellipse()` werden hierzu noch auf dem Arbeitsblatt vorgegeben. Da bereits ein Auge vorhanden ist, muss nur der Mittelpunkt angepasst werden, die Symmetrie der Figur vereinfacht die Aufgabe zusätzlich. Dann wird es etwas schwieriger: Ein zweites Bein soll ergänzt werden. Zwar ist bereits ein Bein vorhanden, die Bedeutung der Parameter müssen sich die SchülerInnen jedoch selbst herleiten. Das Ergebnis dieser Aufgabe ist in Abb. 3c zu sehen.



(a) Ausgang (b) 2. Aufgabe (c) 3. Aufgabe (d) 4. Aufgabe

Abb. 3: Erste Schritte in Processing mit der Figur *Walli*

In einer weiteren Aufgabe sollen schrittweise die wichtigsten Methoden im Zusammenhang mit Farben behandelt und in der Skizze mit Walli ergänzt werden (s. Abb. 4). Die SchülerInnen machen sich hierbei zuerst mit der Processing-Referenz vertraut und entnehmen ihr die Funktion der Methode `fill(g)`. Durch `fill()` sollen dann Wallis Augen durch den Wert 128 grau gefärbt werden. Die Änderung der Füllfarbe lässt die Strichfarbe der Augen (Schwarz) als Umrandung sichtbar werden. Das Zeichnen der Striche (insb. der Umrandung gefüllter Primitiven) soll im Anschluss durch die Methode `noStroke()` deaktiviert werden. Hierdurch werden auch Wallis Beine nicht mehr gezeichnet, was durch `stroke(0)` vor dem Zeichnen der Beine rückgängig gemacht werden muss. Danach soll Wallis Körper in einer beliebigen Farbe gefärbt werden. Der Aufruf von `fill()` vor dem Zeichnen des Körpers hat hierbei aufgrund der Reihenfolge der Methodenaufrufe (bewusst) zur Konsequenz, dass auch der Kopf gefärbt wird. Ein eingefügtes `fill(255)` vor dem Zeichnen des Kopfes vollendet Walli.



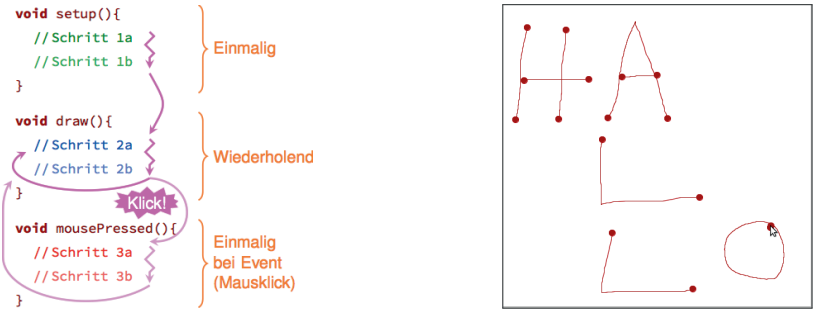
(a) Ausgang (b) `fill(128)` (c) `noStroke()` (d) `stroke(0)` (e) `fill(col)` (f) `fill(255)`

Abb. 4: Schrittweises Färben von *Walli*

8.2 Paint Light – Interaktive Skizzen durch Events

Die dritte Doppelstunde thematisiert interaktive Skizzen in Processing. Abb. 5a zeigt den schematischen Ablauf eines Processing-Programms. Nachdem einmalig die `setup()`-Methode durchlaufen wurde, springt Processing in die `draw()`-Methode, welche als Endlosschleife wiederholend abgearbeitet wird. Spezielle Methoden (z. B. `mousePressed()` oder `keyPressed()`) werden einmalig ausgeführt, wenn das entsprechende Event eintritt.

Bereits am Ende der dritten Doppelstunde sind die SchülerInnen in der Lage, ein einfaches Zeichenprogramm – *Paint Light* – zu programmieren (s. Abb. 5b). Dieses kann bei gedrückter Maustaste den Weg der Maus nachzeichnen und die Leinwand durch einen Tastendruck der Tastatur löschen.



(a) Programmfluss mit Event (b) Paint Light mit Events

Abb. 5: Events in Processing

8.3 Push the button – Erleuchtende Verweigung

Die Aufgabe *Push the button* der Doppelstunde über Vezweigungen ist als fortgeschrittene Aufgabe ergebnisorientiert und mit Zusatzteil konzipiert. Sie zeigt außerdem, wie einfach in Processing Zusatzaufgaben zur Differenzierung erstellt werden können:

Aufgabe: Push the button *In Skizze* `sketch_13_push_the_button_a.pde` *siehst du – mit etwas Phantasie – einen grünen Schalter in einem dunklen Raum. Dieser Schalter soll, sobald er mit der Maus angeklickt wird, das Licht im Raum einschalten (also den Hintergrund weiß färben). Das Licht soll brennen, bis er erneut angeklickt wird.*

1. *Implementiere diese Funktion unter Verwendung der vorhandenen Variablen.*
2. Zusatz: *Erweitere die Skizze um einen sog. Hover-Effekt des Schalters. D. h. der Schalter soll heller gefärbt werden, wenn sich der Mauszeiger darüber befindet.*

Processing stellt die Systemvariablen `mouseX` und `mouseY` zur Verfügung. Neben Variablen für die Position und Größe des Schalters (`buttonX` etc.) ist außerdem die boolesche Variable `lightOn` bereits vorgegeben. Mithilfe einer Vezweigung und logischer Operatoren kann in `mousePressed()` der Wert von `lightOn` wie in List. 1 invertiert und innerhalb von `draw()` der Hintergrund der Skizze entsprechend gefärbt werden.

```
if (mouseX > buttonX && mouseX < buttonX + buttonW &&
  mouseY > buttonY && mouseY < buttonY + buttonH) {
  lightOn = !lightOn;
}
```

List. 1: Mausklick auf „Taste“ durch Verzweigung überprüfen

9 Erfahrungen

Die Unterrichtseinheit wurde im Schuljahr 2013/2014 ursprünglich für einen zweistündigen Informatikkurs im ersten Jahr der Kursstufe an einem allgemeinbildenden Gymnasium (G8) konzipiert. Der Kurs bestand aus 14 SchülerInnen und wurde in einem Informatik-Raum mit fest installierten Rechnern unterrichtet.

Im gleichen Schuljahr wurde die Unterrichtseinheit in einem Parallelkurs unterrichtet. Im Schuljahr 2014/2015 wurde sie in einem weiteren zweistündigen Informatikkurs erneut erprobt, ergänzt und verbessert.

Insbesondere zwei negative Aspekte offenbarten sich beim Einsatz von Processing: Zum einen war – insbesondere zu Beginn der Unterrichtseinheit – die korrekte Bestimmung von Koordinaten der nötigen Grafikprimitiven sehr zeitintensiv und zeigte aufgrund des Trial-and-Error-Vorgehens der meisten SchülerInnen nur einen geringen Lerneffekt. Zum anderen zeigte sich, dass der grafische Ansatz von Processing beim Erlernen oder Überprüfen neuer Inhalte auch im Weg stehen kann. Man muss beim Erstellen neuer Aufgaben gezielt darauf achten, dass dies nicht der Fall ist. Sollen die SchülerInnen z. B. beim Erlernen von Verzweigungen innerhalb der Verzweigung ein Grafikprimitiv zeichnen, kann die Zeichnung für sie anspruchsvoller als das eigentlichen Thema sein. Somit entsteht ein falscher Fokus. Entgegenwirken kann man dieser Problematik, indem man sie bei der Aufgabengestaltung jederzeit berücksichtigt und ggf. nötige Teile des Quelltextes vorgibt. Gänzlich beheben kann man sie jedoch nicht.

Ebenfalls zeigte sich, dass die Zeit für die Themen in Abschnitt 7 sehr knapp bemessen ist. Der Kurs der ersten Durchführung arbeitete sehr gewissenhaft und auf hohem Niveau, sodass die Verteilung insgesamt eingehalten werden könnte. Bei den folgenden Durchläufen war dies (auch aufgrund der nicht optimalen Ausstattung durch einen Laptopwagen und dem resultierenden Zeitverlust) nicht möglich. Durch die Einteilung der Themen in mehrere Phasen ist eine flexible Aufteilung der Inhalte jedoch leicht möglich.

Alle drei Durchläufe der Unterrichtseinheit offenbarten vor allem sehr großes Interesse der SchülerInnen. Die im Abschnitt 3 erhofften Vorteile Processings bestätigten sich größtenteils. Über die gesamte Unterrichtseinheit arbeiteten die SchülerInnen motiviert an den vorgelegten Aufgaben und im Klassenunterricht mit. Sicherlich ist dies auch auf die allgemeine Lernatmosphäre des jeweiligen Kurses zurückzuführen. Häufige Äußerungen einzelner SchülerInnen während des Unterrichts zeigten jedoch, dass sie sich mit Freude mit den neuen Inhalten beschäftigen. Abschließende Klausuren und über Moodle durchgeführte Umfragen über die Unterrichtseinheit fielen ebenfalls größtenteils positiv aus. Insgesamt verlief die Unterrichtseinheit erfolgreich.

Literaturverzeichnis

[Ba10] Bartmann, Erik: Processing. O'Reilly, 1. Auflage, 2010.

[FR] Fry, Ben; Reas, Casey: Processing. `http://www.processing.org`.

[Ge08] Gesellschaft für Informatik (GI) e. V.: Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I. 2008. http://www.sn.schule.de/~istandard/docs/bildungsstandards_2008.pdf.

[Ho] Hoffmann, Reiner: BASIC. http://www.madeasy.de/2/basic.htm.

[Hu07] Hubwieser, Peter: Didaktik der Informatik – Grundlagen, Konzepte, Beispiele. 3. Ausgabe. Springer, 2007.

[Ki] Kipp, Michael: Processing – Programmieren lernen. http://michaelkipp.de/processing.

[Lo] Logo Foundation: Webpräsenz. http://el.media.mit.edu/logo-foundation.

[LP] La Trobe University, Melbourne, Australien; Programming Education Tools Group, University of Kent, Canterbury, England: BlueJ. http://www.bluej.org.

[Me87] Meyer, Hilbert: UnterrichtsMethoden, Jgg. 2. 1987.

[Me10] Mester, Arnulf: Programmieren lernen mit Processing. Workshop des Informatiktags der ILL-BW, 2010.

[MI] MIT Media Laboratory, Cambridge, USA: Scratch. http://scratch.mit.edu.

[Mi04] Ministerium für Kultus, Jugend und Sport Baden-Württemberg: Bildungsplan 2004 – Allgemein bildendes Gymnasium. 2004. http://www.bildung-staerkt-menschen.de/service/downloads/Bildungsplaene/Gymnasium/Gymnasium_Bildungsplan_Gesamt.pdf.

[Na] Nagel, Till: Creative Coding. http://btk.tillnagel.com.

[Pr] Programming Education Tools Group, University of Kent, Canterbury, England: Greenfoot. http://www.greenfoot.org.

[RNH01] Reichert, Raimond; Nievergelt, Jürg; Hartmann, Werner: Programming in schools – why, and how? In (Hartmann, Werner; Näf, Michael; Reichert, Raimond, Hrsg.): Enseigner l'informatique, S. 143–152. Springer, 2001. http://www.swisseduc.ch/informatik/karatojava/docs/programming_why_how.pdf.

[Sh08] Shiffman, Daniel: Learninig Processing – A Beginner's Guide to Programming Images, Animation, and Interaction. 1. Ausgabe. Morgan Kaufmann, 2008.

[Sw] SwissEduc: Kara. http://www.swisseduc.ch/informatik/karatojava/kara.

[Wa11] Wanner, Andres: Processing – eine Einführung in die Programmierung. lulu.com, 1.1. Auflage, 2011.

[Wi] Wikipedia (de): Processing. https://de.wikipedia.org/wiki/Processing.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Lan-guage to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Business 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Er-fahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT:
Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting. CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik –
Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I^{2}CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-160 Gregor Engels, Markus Luckey, Alexander Pretschner, Ralf Reussner (Hrsg.)
Software Engineering 2010 – Workshopband
(inkl. Doktorandensymposium)

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

P-164 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2010: Biometrics and Electronic Signatures Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-165 Gerald Eichler, Peter Kropf, Ulrike Lechner, Phayung Meesad, Herwig Unger (Eds.)
10th International Conference on Innovative Internet Community Systems (I^2CS) – Jubilee Edition 2010 –

P-166 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
3. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-167 Robert Krimmer, Rüdiger Grimm (Eds.)
4th International Conference on Electronic Voting 2010
co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-168 Ira Diethelm, Christina Dörge, Claudia Hildebrandt, Carsten Schulte (Hrsg.)
Didaktik der Informatik
Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik

P-169 Michael Kerres, Nadine Ojstersek Ulrik Schroeder, Ulrich Hoppe (Hrsg.)
DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V.

P-170 Felix C. Freiling (Hrsg.)
Sicherheit 2010
Sicherheit, Schutz und Zuverlässigkeit

P-171 Werner Esswein, Klaus Turowski, Martin Juhrisch (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2010)
Modellgestütztes Management

P-172 Stefan Klink, Agnes Koschmider Marco Mevius, Andreas Oberweis (Hrsg.)
EMISA 2010
Einflussfaktoren auf die Entwicklung flexibler, integrierter Informationssysteme
Beiträge des Workshops der GI-Fachgruppe EMISA (Entwicklungsmethoden für Informationssysteme und deren Anwendung)

P-173 Dietmar Schomburg, Andreas Grote (Eds.)
German Conference on Bioinformatics 2010

P-174 Arslan Brömme, Torsten Eymann, Detlef Hühnlein, Heiko Roßnagel, Paul Schmücker (Hrsg.)
perspeGKtive 2010
Workshop „Innovative und sichere Informationstechnologie für das Gesundheitswesen von morgen“

P-175 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 1

P-176 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 2

P-177 Witold Abramowicz, Rainer Alt, Klaus-Peter Fähnrich, Bogdan Franczyk, Leszek A. Maciaszek (Eds.)
INFORMATIK 2010
Business Process and Service Science – Proceedings of ISSS and BPSC

P-178 Wolfram Pietsch, Benedikt Krams (Hrsg.)
Vom Projekt zum Produkt
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschafts-informatik (WI-MAW), Aachen, 2010

P-179 Stefan Gruner, Bernhard Rumpe (Eds.)
FM+AM`2010
Second International Workshop on Formal Methods and Agile Methods

P-180 Theo Härder, Wolfgang Lehner, Bernhard Mitschang, Harald Schöning, Holger Schwarz (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW)
14. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme“ (DBIS)

P-181 Michael Clasen, Otto Schätzel, Brigitte Theuvsen (Hrsg.)
Qualität und Effizienz durch informationsgestützte Landwirtschaft, Fokus: Moderne Weinwirtschaft

P-182 Ronald Maier (Hrsg.)
6th Conference on Professional Knowledge Management
From Knowledge to Action

P-183 Ralf Reussner, Matthias Grund, Andreas Oberweis, Walter Tichy (Hrsg.)
Software Engineering 2011
Fachtagung des GI-Fachbereichs Softwaretechnik

P-184 Ralf Reussner, Alexander Pretschner, Stefan Jähnichen (Hrsg.)
Software Engineering 2011
Workshopband
(inkl. Doktorandensymposium)

P-185 Hagen Höpfner, Günther Specht, Thomas Ritz, Christian Bunse (Hrsg.)
MMS 2011: Mobile und ubiquitäre Informationssysteme Proceedings zur 6. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2011)

P-186 Gerald Eichler, Axel Küpper, Volkmar Schau, Hacène Fouchal, Herwig Unger (Eds.)
11th International Conference on Innovative Internet Community Systems (I^2CS)

P-187 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
4. DFN-Forum Kommunikations-technologien, Beiträge der Fachtagung 20. Juni bis 21. Juni 2011 Bonn

P-188 Holger Rohland, Andrea Kienle, Steffen Friedrich (Hrsg.)
DeLFI 2011 – Die 9. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. 5.–8. September 2011, Dresden

P-189 Thomas, Marco (Hrsg.)
Informatik in Bildung und Beruf
INFOS 2011
14. GI-Fachtagung Informatik und Schule

P-190 Markus Nüttgens, Oliver Thomas, Barbara Weber (Eds.)
Enterprise Modelling and Information Systems Architectures (EMISA 2011)

P-191 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2011
International Conference of the Biometrics Special Interest Group

P-192 Hans-Ulrich Heiß, Peter Pepper, Holger Schlingloff, Jörg Schneider (Hrsg.)
INFORMATIK 2011
Informatik schafft Communities

P-193 Wolfgang Lehner, Gunther Piller (Hrsg.)
IMDM 2011

P-194 M. Clasen, G. Fröhlich, H. Bernhardt, K. Hildebrand, B. Theuvsen (Hrsg.)
Informationstechnologie für eine nachhaltige Landbewirtschaftung
Fokus Forstwirtschaft

P-195 Neeraj Suri, Michael Waidner (Hrsg.)
Sicherheit 2012
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 6. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

P-196 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2012
Proceedings of the 11th International Conference of the Biometrics Special Interest Group

P-197 Jörn von Lucke, Christian P. Geiger, Siegfried Kaiser, Erich Schweighofer, Maria A. Wimmer (Hrsg.)
Auf dem Weg zu einer offenen, smarten und vernetzten Verwaltungskultur
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2012

P-198 Stefan Jähnichen, Axel Küpper, Sahin Albayrak (Hrsg.)
Software Engineering 2012
Fachtagung des GI-Fachbereichs Softwaretechnik

P-199 Stefan Jähnichen, Bernhard Rumpe, Holger Schlingloff (Hrsg.)
Software Engineering 2012
Workshopband

P-200 Gero Mühl, Jan Richling, Andreas Herkersdorf (Hrsg.)
ARCS 2012 Workshops

P-201 Elmar J. Sinz Andy Schürr (Hrsg.)
Modellierung 2012

P-202 Andrea Back, Markus Bick, Martin Breunig, Key Pousttchi, Frédéric Thiesse (Hrsg.)
MMS 2012:Mobile und Ubiquitäre Informationssysteme

P-203 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
5. DFN-Forum Kommunikations-technologien
Beiträge der Fachtagung

P-204 Gerald Eichler, Leendert W. M. Wienhofen, Anders Kofod-Petersen, Herwig Unger (Eds.)
12th International Conference on Innovative Internet Community Systems (I2CS 2012)

P-205 Manuel J. Kripp, Melanie Volkamer, Rüdiger Grimm (Eds.)
5th International Conference on Electronic Voting 2012 (EVOTE2012)
Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-206 Stefanie Rinderle-Ma, Mathias Weske (Hrsg.)
EMISA 2012
Der Mensch im Zentrum der Modellierung

P-207 Jörg Desel, Jörg M. Haake, Christian Spannagel (Hrsg.)
DeLFI 2012: Die 10. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.
24.–26. September 2012

P-208 Ursula Goltz, Marcus Magnor, Hans-Jürgen Appelrath, Herbert Matthies, Wolf-Tilo Balke, Lars Wolf (Hrsg.) INFORMATIK 2012

P-209 Hans Brandt-Pook, André Fleer, Thorsten Spitta, Malte Wattenberg (Hrsg.) Nachhaltiges Software Management

P-210 Erhard Plödereder, Peter Dencker, Herbert Klenk, Hubert B. Keller, Silke Spitzer (Hrsg.) Automotive – Safety & Security 2012 Sicherheit und Zuverlässigkeit für automobile Informationstechnik

P-211 M. Clasen, K. C. Kersebaum, A. Meyer-Aurich, B. Theuvsen (Hrsg.) Massendatenmanagement in der Agrar- und Ernährungswirtschaft Erhebung - Verarbeitung - Nutzung Referate der 33. GIL-Jahrestagung 20. – 21. Februar 2013, Potsdam

P-212 Arslan Brömme, Christoph Busch (Eds.) BIOSIG 2013 Proceedings of the 12th International Conference of the Biometrics Special Interest Group 04.–06. September 2013 Darmstadt, Germany

P-213 Stefan Kowalewski, Bernhard Rumpe (Hrsg.) Software Engineering 2013 Fachtagung des GI-Fachbereichs Softwaretechnik

P-214 Volker Markl, Gunter Saake, Kai-Uwe Sattler, Gregor Hackenbroich, Bernhard Mit schang, Theo Härder, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 13. – 15. März 2013, Magdeburg

P-215 Stefan Wagner, Horst Lichter (Hrsg.) Software Engineering 2013 Workshopband (inkl. Doktorandensymposium) 26. Februar – 1. März 2013, Aachen

P-216 Gunter Saake, Andreas Henrich, Wolfgang Lehner, Thomas Neumann, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 – Workshopband 11. – 12. März 2013, Magdeburg

P-217 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.) 6. DFN-Forum Kommunikations- technologien Beiträge der Fachtagung 03.–04. Juni 2013, Erlangen

P-218 Andreas Breiter, Christoph Rensing (Hrsg.) DeLFI 2013: Die 11 e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI) 8. – 11. September 2013, Bremen

P-219 Norbert Breier, Peer Stechert, Thomas Wilke (Hrsg.) Informatik erweitert Horizonte INFOS 2013 15. GI-Fachtagung Informatik und Schule 26. – 28. September 2013

P-220 Matthias Horbach (Hrsg.) INFORMATIK 2013 Informatik angepasst an Mensch, Organisation und Umwelt 16. – 20. September 2013, Koblenz

P-221 Maria A. Wimmer, Marijn Janssen, Ann Macintosh, Hans Jochen Scholl, Efthimios Tambouris (Eds.) Electronic Government and Electronic Participation Joint Proceedings of Ongoing Research of IFIP EGOV and IFIP ePart 2013 16. – 19. September 2013, Koblenz

P-222 Reinhard Jung, Manfred Reichert (Eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2013) St. Gallen, Switzerland September 5. – 6. 2013

P-223 Detlef Hühnlein, Heiko Roßnagel (Hrsg.) Open Identity Summit 2013 10. – 11. September 2013 Kloster Banz, Germany

P-224 Eckhart Hanser, Martin Mikusz, Masud Fazal-Baqaie (Hrsg.) Vorgehensmodelle 2013 Vorgehensmodelle – Anspruch und Wirklichkeit 20. Tagung der Fachgruppe Vorgehensmodelle im Fachgebiet Wirtschaftsinformatik (WI-VM) der Gesellschaft für Informatik e.V. Lörrach, 2013

P-225 Hans-Georg Fill, Dimitris Karagiannis, Ulrich Reimer (Hrsg.) Modellierung 2014 19. – 21. März 2014, Wien

P-226 M. Clasen, M. Hamer, S. Lehnert, B. Petersen, B. Theuvsen (Hrsg.) IT-Standards in der Agrar- und Ernährungswirtschaft Fokus: Risiko- und Krisenmanagement Referate der 34. GIL-Jahrestagung 24. – 25. Februar 2014, Bonn

P-227 Wilhelm Hasselbring,
Nils Christian Ehmke (Hrsg.)
Software Engineering 2014
Fachtagung des GI-Fachbereichs Softwaretechnik
25. – 28. Februar 2014
Kiel, Deutschland

P-228 Stefan Katzenbeisser, Volkmar Lotz, Edgar Weippl (Hrsg.)
Sicherheit 2014
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
19. – 21. März 2014, Wien

P-230 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2014
Proceedings of the 13th International Conference of the Biometrics Special Interest Group
10. – 12. September 2014 in Darmstadt, Germany

P-231 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
7. DFN-Forum Kommunikationstechnologien
16. – 17. Juni 2014
Fulda

P-232 E. Plödereder, L. Grunske, E. Schneider, D. Ull (Hrsg.)
INFORMATIK 2014
Big Data – Komplexität meistern
22. – 26. September 2014
Stuttgart

P-233 Stephan Trahasch, Rolf Plötzner, Gerhard Schneider, Claudia Gayer, Daniel Sassiat, Nicole Wöhrle (Hrsg.)
DeLFI 2014 – Die 12. e-Learning Fachtagung Informatik
der Gesellschaft für Informatik e.V.
15. – 17. September 2014
Freiburg

P-234 Fernand Feltz, Bela Mutschler, Benoît Otjacques (Eds.)
Enterprise Modelling and Information Systems Architectures
(EMISA 2014)
Luxembourg, September 25-26, 2014

P-235 Robert Giegerich,
Ralf Hofestädt,
Tim W. Nattkemper (Eds.)
German Conference on Bioinformatics 2014
September 28 – October 1
Bielefeld, Germany

P-236 Martin Engstler, Eckhart Hanser, Martin Mikusz, Georg Herzwurm (Hrsg.)
Projektmanagement und Vorgehensmodelle 2014
Soziale Aspekte und Standardisierung
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V., Stuttgart 2014

P-237 Detlef Hühnlein, Heiko Roßnagel (Hrsg.)
Open Identity Summit 2014
4.–6. November 2014
Stuttgart, Germany

P-238 Arno Ruckelshausen, Hans-Peter Schwarz, Brigitte Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Referate der 35. GIL-Jahrestagung
23. – 24. Februar 2015, Geisenheim

P-239 Uwe Aßmann, Birgit Demuth, Thorsten Spitta, Georg Püschel, Ronny Kaiser (Hrsg.)
Software Engineering & Management 2015
17.-20. März 2015, Dresden

P-240 Herbert Klenk, Hubert B. Keller, Erhard Plödereder, Peter Dencker (Hrsg.)
Automotive – Safety & Security 2015
Sicherheit und Zuverlässigkeit für automobile Informationstechnik
21.–22. April 2015, Stuttgart

P-241 Thomas Seidl, Norbert Ritter, Harald Schöning, Kai-Uwe Sattler, Theo Härder, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015)
04. – 06. März 2015, Hamburg

P-242 Norbert Ritter, Andreas Henrich, Wolfgang Lehner, Andreas Thor, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015) – Workshopband
02. – 03. März 2015, Hamburg

P-243 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
8. DFN-Forum Kommunikationstechnologien
06.–09. Juni 2015, Lübeck

P-244 Alfred Zimmermann,
Alexander Rossmann (Eds.)
Digital Enterprise Computing
(DEC 2015)
Böblingen, Germany June 25-26, 2015

P-246 Douglas W. Cunningham, Petra Hofstedt,
Klaus Meer, Ingo Schmitt (Hrsg.)
INFORMATIK 2015
28.9.-2.10. 2015, Cottbus

P-247 Hans Pongratz, Reinhard Keil (Hrsg.)
DeLFI 2015 – Die 13. E-Learning
Fachtagung Informatik der
Gesellschaft für Informatik e.V. (GI)
1.-4. September 2015
München

P-249 Jens Gallenbacher (Hrsg.)
Informatik
allgemeinbildend begreifen
INFOS 2015 16. GI-Fachtagung
Informatik und Schule
20.–23. September 2015

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de

Gesellschaft für Informatik e.V. (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into
• seminars
• proceedings
• dissertations
• thematics
current topics are dealt with from the vantage point of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure high quality contributions.

The volumes are published in German or English.

Information: http://www.gi.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-643-5

"INFOS 2015" is the 16th event in a conference series organized by the GI special interest group IBS, which focuses on education in informatics (computer science) in schools. The carefully reviewed contributions reflect the state of the art in didactics of informatics (computer science education research), including examples of best practice, research methodology and results, educational models, and learning environments.