INFOS 2013

15. GI-Fachtagung „Informatik und Schule“

Praxisband

Norbert Breier, Peer Stechert, Thomas Wilke
(Hrsg.)

26.–28. September 2013 in Kiel

Kiel Computer Science Series (KCSS) 2013/3 v1.1

ISBN 978-1-291-62236-2 (print version)
ISBN 978-1-291-63388-7 (electronic version)

Electronic version, updates, errata available via https://www.informatik.uni-kiel.de/kcss

The author can be contacted via thomas.wilke@email.uni-kiel.de

Published by the Department of Computer Science, Kiel University

Please cite as:

▻ Norbert Breier, Peer Stechert, Thomas Wilke (Hrsg.). *INFOS 2013, 15. GI-Fachtagung „Informatik und Schule", Praxisband.* Number 2013/3 in Kiel Computer Science Series. Department of Computer Science, Kiel University, 2013.

```
@proceedings{
  booktitle = {INFOS 2013, 15. GI-Fachtagung "Informatik und Schule", Praxisband},
  editor = {Norbert Breier, Peer Stechert, Thomas Wilke},
  publisher = {Department of Computer Science, CAU Kiel},
  year = {2013},
  number = {2013-03},
  series = {Kiel Computer Science Series}
}
```

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit `http://www.informatik.uni-kiel.de/kcss` for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

Vorwort

Die Fachtagung „Informatik und Schule“ (INFOS) der Gesellschaft für Informatik e. V. (GI) fördert traditionsgemäß den Erfahrungsaustausch zwischen Lehrenden und Forschenden und bietet Anregungen dafür, wie erfolgreicher Informatikunterricht gestaltet werden kann. Sie wird in diesem Jahr zum 15. Mal ausgerichtet, steht unter dem Motto „Informatik erweitert Horizonte“ und findet in Kiel und damit zum ersten Mal in Schleswig-Holstein statt.

Die INFOS soll eine zeitgemäße qualifizierte informatische Bildung in der Schule auf vielfältige Weise befördern. So besteht das Programm in diesem Jahr aus vier Vorträgen eingeladener Referentinnen und Referenten sowie vierzehn ausgewählten Forschungs-, Entwicklungs- und Erfahrungsbeiträgen, die vom Programmkomitee aus 34 eingereichten Beiträgen ausgewählt wurden. Diese Beiträge beleuchten die folgenden Schwerpunkte der Tagung aus unterschiedlichen Perspektiven: Informatik im Kontext, Schülervorstellungen, Informatikunterricht in der Sekundarstufe I, Schülerlabore, Lernchancen im Informatikunterricht, Roboter im Informatikunterricht sowie Studium und Fortbildung. Eine Abrundung erfährt die Konferenz durch insgesamt 18 Workshops, ein Forum für Doktorandinnen und Doktoranden, ein Forum für Lehrkräfte im Ausbildungsstadium, Berichte aus der schulischen Praxis, Poster und eine Ausstellung.

Der hier vorliegende „Praxisband“ dokumentiert gemeinsam mit dem eigentlichen Tagungsband die Ergebnisse der INFOS 2013. Neben den Berichten aus der Praxis enthält er Aufsätze zu fast allen Workshops, einen Beitrag zu einem vorgestellten Poster sowie die prämierte Einreichung für den Unterrichtspreis der Gesellschaft für Informatik e. V.

Wir danken allen Autorinnen und Autoren für ihre Beiträge zu diesem Praxisband.

Kiel und Hamburg, im September 2013

Norbert Breier
Peer Stechert
Thomas Wilke

Inhaltsverzeichnis

Workshop-Artikel 93

Modellierung als ein wichtiges Werkzeug der Informatik am Beispiel einer Ampelsteuerung mittels Mikrocontroller

Nadine Bergner, Marc Weintz, Ulrik Schroeder

Lehr- und Forschungsgebiet Informatik 9
RWTH Aachen
Ahornstr. 55
52074 Aachen
{bergner, schroeder}@informatik.rwth-aachen.de
marc.weintz@rwth-aachen.de

Abstract: Anhand einer Alltagssituation aus dem Straßenverkehr erfahren Schülerinnen und Schüler (SuS) der Sekundarstufe I im Rahmen dieses Moduls des InfoSphere – Schülerlabor Informatik der RWTH Aachen auf experimentelle Weise die Bedeutung der Modellierung als eines der grundlegenden informatischen Werkzeuge. Dazu bauen und verschalten die Lerner zuerst ein echtes Kreuzungsmodell, bevor sie dieses digital modellieren und anschließend implementieren. Die SuS erfahren die Modellierung als strukturelles Hilfsmittel für den Programmierer und sehen darüber hinaus den Nutzen für die automatisierte Generierung von Quellcode. Insgesamt erlangen die SuS durch entdeckendes Lernen ein Gefühl dafür, wie theoretische und praktische Aspekte in der Informatik auf einmalige Art und Weise zusammen fließen.

1 Einleitung und Motivation

Modellierung ist eines der grundlegenden Themengebiete der Informatik und als solches auch Bestandteil der Bildungsstandards der Gesellschaft für Informatik (GI) (siehe [Ge08]). Daher stellt sich die Frage, wie dieses wichtige Gebiet auch SuS der Sekundarstufe I zielgruppengerecht vermittelt werden kann. Speziell die Situation in Nordrhein-Westfalen, dass viele Informatik-Lehrkräfte des Wahlpflichtbereichs (Klassenstufe 8 und 9) Quereinsteiger sind und somit kein (komplettes) Lehramtsstudium im Fach Informatik absolviert haben, erfordert es hier Unterrichtsangebote zu verschiedenen, neuartigen Themengebieten zur Erweiterung des Schulunterrichts anzubieten. Das im Folgenden vorgestellte Modul „Grün, gelb, rot – Aufbau, Modellierung und Programmierung einer Ampelanlage“ des InfoSphere – Schülerlabor Informatik[1] der RWTH Aachen ermöglicht es den Teilnehmenden in einem alltagsnahen Kontext die Vorteile der Modellierung selbstständig zu entdecken. Als Anwendungsbezug wurde hier die Modellierung einer Kreuzungssituation gewählt, da diese allen SuS aus ihrem alltäglichen Leben bekannt ist und somit möglichst viele

[1] http://schuelerlabor.informatik.rwth-aachen.de

Lerner anspricht. Für dieses Modul wurden elektrische Ampelmodelle entwickelt, um den SuS das Lernen am echten Modell (siehe [Be06]) zu ermöglichen. Die Arbeit mit den Ampelmodellen bietet darüber hinaus eine Verknüpfung zur Elektrotechnik bzw. Physik, da die Situationen immer zuerst aus den Modellen erbaut werden müssen, bevor es in einer zweiten Phase zur digitalen Modellierung und schließlich zur Implementierung kommt.

2 Related work

Der Themenbereich der computergesteuerten Ampelanlagen wurde bisher zweimal in ähnlicher Weise für SuS didaktisch aufbereitet. Da ist zum einen der Ansatz von Gerhard Lechner, einem Hauptschullehrer aus Imst, „Bau und Programmierung einer computergesteuerten Ampelanlage“ (siehe [Le09]) und zum anderen das Projekt „Objektorientierte Modellierung und Programmierung“ für eine 10. Jahrgangsstufe vom Werner-von-Siemens-Gymnasium in Regensburg (siehe [We12]) zu nennen.

Der erste Ansatz wurde für eine siebte Klasse einer Hauptschule im fächerübergreifenden Unterricht in Physik und IKT (Informations- und Kommunikationstechnik) entwickelt. Im Rahmen dieser Unterrichtsreihe wurden die Schaltungen als reale Modelle aufgebaut, jedoch standen nicht die Modellierung, sondern die elektrotechnischen Grundlagen und die textuelle Implementierung im Vordergrund. Die SuS erlernten im Verlauf dieser Reihe ihre erste Programmiersprache, wodurch ein Fokus auf die Programmierung entstand. Insgesamt lag also eine ähnliche Motivation vor, jedoch ein komplett anderer inhaltlicher Schwerpunkt.

Bei dem zweiten Projekt liegt der informatische Schwerpunkt, wie bei dem im Folgenden vorgestellten Modul, auf der Modellierung. Allerdings wurde im Anschluss an die Modellierungsphase immer händisch in Java implementiert. Die Modellierung diente also lediglich als Vorbereitung zur Programmierungsphase, welche nicht automatisiert wurde. Die komplette Modellierung wurde hierbei in BlueJ vermittelt, das heißt die Modelle wurden lediglich virtuell erzeugt und es gab keinerlei Hands-On Materialien oder ähnliche greifbare Objekte.

Somit vereint das hier beschriebene Modul Aspekte beider Ansätze, indem die motivierenden Elemente der realen Modelle mit dem Fokus auf die informatische Modellierung kombiniert werden. Im Gegensatz zum ersten Projekt werden die Kreuzungssituationen bereits von Beginn an von den Lernern selbst aufgebaut und angeschlossen, jedoch nicht mehr selbst verlötet. Dies ist im fächerübergreifenden Unterricht mit der Physik als eigenständiger Inhalt gewünscht, würde im Modul jedoch die Aufmerksamkeit von der Informatik weg lenken. Anfangs wird im Modul die Modellierung analog zum zweiten Ansatz als Vorbereitung zur Implementierung verwendet, aber im Laufe des Moduls lernen die SuS auch die Möglichkeiten des automatischen Generierens von Programmcode aus den Automatenmodellen kennen und erfahren so wie Modelle auch zur Implementierung genutzt werden können.

3 Überblick über das Modul

3.1 Zielgruppe, benötigtes Vorwissen und Einordnung in den Schulunterricht

Dieses Modul hat, wie bereits oben erwähnt, das Ziel SuS der Sekundarstufe I experimentell zu vermitteln, was Modellierung im informatischen Sinne ist und welche Rolle sie in der Fachdisziplin spielt. Die Zielgruppe sind SuS des Wahlpflichtbereichs II, also der Jahrgangsstufen 8 und 9. Diese Zielgruppe wurde gewählt, um SuS vor der Wahl ihrer Kurse für die Sekundarstufe II bzw. einer Ausbildung einen Blick auf die Breite der Informatik zu ermöglichen. Somit soll frühzeitig vermittelt werden, dass die Informatik praktische und auch theoretische Elemente beinhaltet und auf einmalige Weise kombiniert.

Insgesamt umfasst eine Moduldurchführung einen Zeitraum von ca. 4,5 Zeitstunden inkl. Pausen, was jedoch abhängig vom Vorwissen und Können der Teilnehmergruppe etwas variiert. Das nötige Vorwissen beschränkt sich dabei auf erste Erfahrungen in der Programmierung, wobei explizit keine textuelle Programmiersprache vorausgesetzt wird; es sollen lediglich die wichtigsten Programmierkonzepte (Schleifen, Abfragen und Variablen) bereits bekannt sein.

Da es für das Fach Informatik in der Sekundarstufe I in NRW keine gültigen und verbindlichen Vorgaben in Form eines Lehrplans gibt, empfiehlt sich eine Orientierung an den Bildungsstandards der GI (siehe [Ge08]). In der dortigen Auflistung der Inhalts- und Prozessbereiche für die Sekundarstufe I trifft besonders der Bereich „Modellieren und Implementieren“ exakt den Schwerpunkt dieses Moduls. Darüber hinaus kann das Modul durch den Einsatz der Mikrocontroller zumindest anteilig den Inhaltsbereich „Informatiksysteme“, sowie durch die didaktische Ausgestaltung den Prozessbereich „Kommunizieren und Kooperieren“ abdecken.

3.3 Inhalte des Moduls

Eine Besonderheit dieses Moduls ist, dass es fächerübergreifend zur Elektrotechnik/Physik ausgelegt ist und somit elektrotechnische und informatische Inhalte kombiniert. Im Rahmen der Elektrotechnik werden insbesondere elektrische Stromkreise mit Leds und Widerständen genauer beleuchtet. Darüber hinaus finden Sensoren (Tast- und Drucksensoren) in den Aufbauten Verwendung, damit mit dem System interagiert werden kann (z.B. Fußgänger wartet an der Ampel). Im Bereich der Informatik werden sowohl in die praktische, als auch in die theoretische Informatik erste Einblicke ermöglicht. Die Mikrocontroller-Programmierung mit einem grundlegenden Verständnis von Variablen, Schleifen, Verzweigungen und Methoden deckt dabei die praktische Informatik ab. Einen Einblick in die Welt der theoretischen Informatik erhalten die Teilnehmenden über die verwendeten Automaten, welche hier jedoch nicht theoretisch analysiert, sondern hauptsächlich praktisch verwendet werden.

Im folgenden Kapitel wird das gesamte Modul inklusive Ablauf, Arbeitsmaterialien und Software detailliert beschrieben und wichtige didaktische Entscheidungen dargelegt.

4 Überblick über das Modul

4.1 Ablauf

Das gesamte Modul gliedert sich neben der Begrüßung und der Abschlusspräsentationen in drei große Blöcke: Teil I – Einstieg, Teil II – Station 1-5 und Teil III – Freiarbeit.

Nachdem die Betreuer die SuS im InfoSphere begrüßt haben und ein informierender Unterrichtseinstieg die SuS auf das Thema eingestimmt hat, startet das Modul mit einer interaktiven Einstiegspräsentation in der sich die Betreuer gemeinsam mit den SuS schrittweise an das Thema Modellierung einer Alltagssituation heranarbeiten. Im zweiten Teil durchlaufen die SuS in Zweierteams nacheinander 5 aufeinander aufbauende Stationen. Dabei wird die Kreuzungssituation, angefangen von einer einspurigen Straße mit Baustellenampeln bis zu einer kompletten Kreuzung mit Fußgängerampeln und Sensoren, die den Verkehrsfluss registrieren, sukzessive aufgebaut. Im Anschluss erfolgt nach einer Pause der dritte Teil des Moduls, die Freiarbeitsphase, in welcher die SuS in 4er-Teams eigenständig neue Verkehrssituationen ausarbeiten und dabei den aus Teil II bekannten Prozess des Aufbauens, Modellierens und Implementierens durchlaufen. Die individuellen Ergebnisse werden abschließend von den SuS der gesamten Gruppe präsentiert.

Der zeitliche Ablauf inklusive der jeweiligen Lernmethode, der verwendeten Medien bzw. Materialien und der eingesetzten Software (näheres dazu in Abschnitt 4.2 Verwendete Software) kann der folgenden Tabelle entnommen werden.

Zeit	Inhalt	Methode	Medien/Materialien	Software
0.00	Begrüßung, informierender Einstieg	Betreuervor-trag	Smartboard oder Beamer	
0.10	Teil I - Einstieg	Betreuer-SuS-Gespräch	Smartboard	Notebook-Software der Firma SMART
0.40	Teil II - Station 1	Partnerarbeit	Grundplatte, Modellampeln, Sensoren, Arbeits- & Merkblätter, rote Folie, Zustandskärtchen	Arduino-Sketch & -Loader
1.20	Teil II - Station 2	Partnerarbeit		
1.50	Teil II - Station 3	Partnerarbeit		DAVE & UML2Arduino
2.20	Teil II - Station 4	Partnerarbeit		
2.50	Teil II - Station 5	Partnerarbeit		
3.10	Pause			
3.30	Teil III - Abschlussprojekt	Gruppenar-beit (4 SuS)	Arbeitsblatt, Kreuzungsskizze, Modellampeln & Sensoren	Alle bisherige
4.15	Abschluss	SuS-Präsentation	Alle bisherigen	Alle bisherige
4.30	Ende			

4.2 Verwendete Software

Im Rahmen dieses Moduls werden neben der Präsentationssoftware Notebook der Firma SMART für die interaktive Einstiegspräsentation insgesamt vier (teils selbst entwickelte) Software-Produkte verwendet:

- Arduino-Sketch[2]: Einfacher Editor mit integriertem Compiler für Arduino-Programme.
- Arduino-Loader: Programm zum vereinfachten Aufspielen eines Arduino-Programms auf den Mikrocontroller. (teilweise Eigenentwicklung, aufbauend auf AVRDUDE[3])
- DAVE[4]: Ein Werkzeug zur Erstellung von Zustandsdiagrammen.
- UML2Arduino: Ein Quelltextgenerator, um aus den Zustandsdiagrammen in DAVE ausführbaren Quelltext zu generieren. (Eigenentwicklung)

Mit dem Arduino-Loader wurde eine grafische Oberfläche zum einfachen Aufspielen fertiger Arduino-Programme auf den Mikrocontroller erschaffen. Dies ist zur Testung der Aufbauten vor der Programmierphase notwendig, um einen korrekten Aufbau des Modells zu überprüfen. Dazu wurde für jede Situation (Station 1-5) ein kleines Testprogramm erstellt, welches die Anschlüsse aller verwendeten Leds testet.

Der Quelltextgenerator UML2Arduino dient dazu in den späteren Stationen aus dem Zustandsmodell, welches mittels DAVE erstellt wurde, automatisiert den Quellcode für den Arduino zu erzeugen. Mit diesem Hilfsmittel erfahren die SuS die Modellierung nicht nur als Hilfsmittel für den Programmierer, sondern darüber hinaus als Möglichkeit der automatischen Quellcodegenerierung.

5 Didaktische Analyse des Moduls

5.1 Allgemeine didaktische und methodische Entscheidungen

Der Leitidee des InfoSphere (siehe [BHS12]) folgend, lernen die SuS in diesem Modul hauptsächlich selbstentdeckend (siehe [M ALB02]). Lediglich der Einstieg (Teil I) des Moduls erfolgt als Betreuer-SuS-Gespräch, um mittels Gesprächsleitung durch die Betreuer einen möglichst homogenen Vorwissensstand zu erzeugen. Alle weiteren Elemente des Moduls werden von den Lernern in Partner- bzw. Teamarbeit eigenständig gemeistert. Dazu wurden die Arbeitsmaterialien auf das selbstständige Lernen zugeschnitten und so gestaltet, dass der Detailgrad der Anleitung von Station 1 zu Station 5 (in Teil II) sukzessive abnimmt und in der Freiarbeitsphase (Teil III) letztendlich lediglich das Szenario vorgestellt wird. Über das Prinzip der abnehmenden

[2] http://arduino.cc/en/Guide/HomePage
[3] http://savannah.nongnu.org/projects/avrdude
[4] http://musoft.cs.uni-dortmund.de:8080/musoft/auto?self=$81d928e8000000f6190f4387

Hilfestellung und schrittweisen Öffnung der Arbeitsaufträge (vgl. [Pe10]) soll durch eigene Erfolgserlebnisse die Selbstwirksamkeitserwartung und Motivation der SuS (siehe [Hu07]) in Bezug auf das informatische Modellieren gesteigert werden. Zugleich bietet insbesondere die Freiarbeitsphase am Ende des Moduls sehr viel Spielraum zur Entfaltung der eigenen Ideen und fördert und unterstützt somit die Kreativität der SuS.

Weitere wichtige didaktische und methodische Entscheidungen in Bezug auf die einzelnen Phasen werden im Folgenden, dem Modulablauf folgend, detailliert erläutert.

5.2 Teil I - Einstieg

Nach der Begrüßung der Lerngruppe in den Räumlichkeiten des InfoSphere und einem kurzen informierenden Einstieg, beginnt der Tag mit einem Betreuer-SuS-Gespräch zum Einblick in die Thematik des Moduls. Dieses wird mit einer interaktiven Präsentation begleitet, welche ermöglicht, dass die SuS und Betreuer gemeinsam die Inhalte erarbeiten, Bilder beschriften, Lückentexte ausfüllen oder Grafiken und Texte richtig anordnen. Als ein Beispiel sind in Abbildung 1 zwei Folien dieser interaktiven Einstiegspräsentation exemplarisch dargestellt. Diese Art der Präsentation wurde gewählt, um die Lerner von Anfang an aktiv in den Lernprozess einzubeziehen und das bei den SuS vorhandene Vorwissen (inkl. möglicher Fehlvorstellungen) direkt aufzugreifen. Da die meisten Informatikkurse in Bezug auf das Vorwissen der SuS eine sehr inhomogene Gruppe bilden, ist es wichtig, dass die Betreuer die Gesprächsleitung übernehmen, um ein allgemeines Grundwissen sicherzustellen. Der Einstieg über die vertraute Lernform des Lehrer-SuS-Gesprächs dient darüber hinaus dazu die SuS an den außerschulischen Lernort und die damit verbundenen neuartigen Lernformen zu gewöhnen. Untersuchungen (siehe [Gu07]) zeigen, dass besonders SuS der Sekundarstufe I andernfalls überfordert bzw. so stark abgelenkt werden, dass der eigentliche Lernprozess gestört wird.

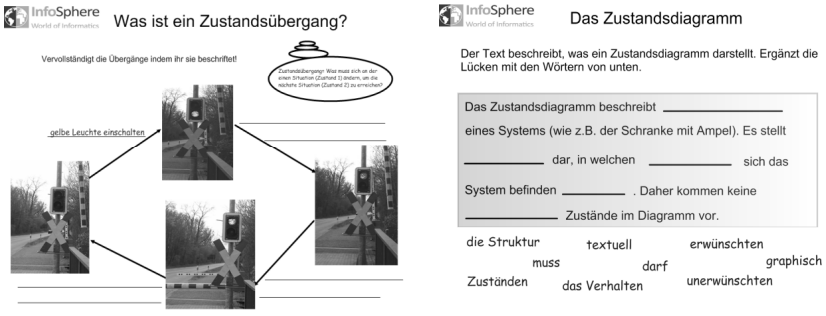
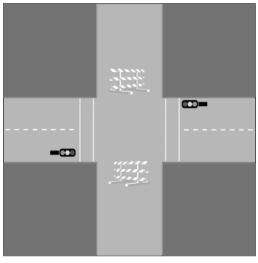


Abbildung 1: Beispiele der interaktiven Einstiegspräsentation

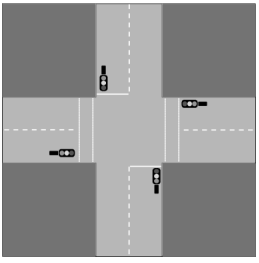
In der Präsentation werden die SuS schrittweise von der ikonischen Darstellung einer Verkehrssituation mittels Fotos zur textuellen Darstellung mittels der entscheidenden Merkmale geführt. Als Verkehrssituation wurde hierbei ein beschrankter Bahnübergang mit seinen verschiedenen Zuständen verwendet, um zum einen ein den SuS bekanntes Szenario aufzugreifen und zum anderen die Übertragung auf die später verwendete Kreuzungssituation zu erleichtern. Darüber hinaus ist dieses Szenario sehr gut geeignet, da die intuitiv wahrgenommenen Zustände der Schranke bzw. Warnleuchte nahezu exakt den Zuständen des Automaten entsprechen. Auch der Begriff des Zustandsübergangs ist für die SuS in diesem Zusammenhang naheliegend. Nach dieser ersten Phase kennen die SuS den Begriff des Zustandes, können Zustände zur Beschreibung eines Systems verwenden, kennen Zustandsübergänge und stellen diese angemessen im Automaten dar.

5.3 Teil II – Station 1-5

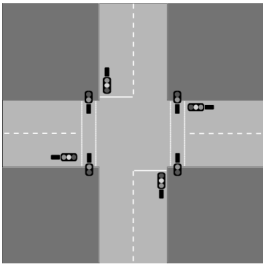
Im zweiten Teil, welcher den Hauptteil des Moduls bildet, bearbeiten die SuS fünf aufeinander aufbauende Stationen (siehe dazu Abbildung 2) die am Beispiel einer Kreuzungssituation die Modellierung mittels Automaten in praktischer Weise vermitteln. Wie die Grafiken zu den einzelnen Stationen schon erkennen lassen, wird über die fünf Stationen hinweg sukzessive eine Kreuzungssituation mit vier Autoampeln, vier Fußgängerampeln mit dazugehörigen Tastern und zwei Drucksensoren in der Straße zur Registrierung der Verkehrslage aufgebaut.



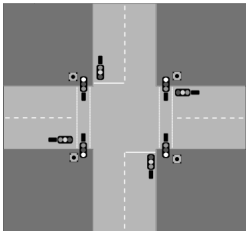
a) Station 1



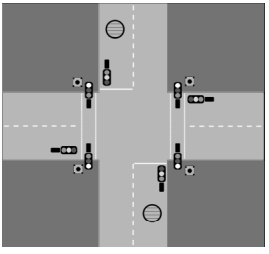
b) Station 2



c) Station 3



d) Station 4



e) Station 5

Abbildung 2: Kreuzungssituationen von Station 1-5

Bei der Bearbeitung jeder einzelnen Station durchlaufen die SuS immer folgende Schritte:

- Aufbau und Verkabelung der Kreuzungssituation aus Modellampeln, Sensoren und Mikrocontroller
- Modellierung der Kreuzungssituation mithilfe von Zustandskärtchen (Station 1 und 2) bzw. mit der Software DAVE (Station 3-5)
- Implementierung entweder über den Arduino-Sketch direkt (Station 1 und 2) oder mittels eines Quelltextgenerators aus DAVE heraus (Station 3-5)

Während des kompletten Teils II arbeiten die SuS in Zweierteams. Dies hat den Vorteil das alle SuS aktiv am Lernprozess beteiligt sind und einzelne Vorwissensunterschiede ausgeglichen werden können (vgl. [Nu03]). Die Gestaltung der Arbeitsmaterialien (siehe Abbildung 3) lässt den SuS viel Freiraum zum Experimentieren und eigenständigen Erforschen, was der gesamten Philosophie des InfoSphere entspricht.

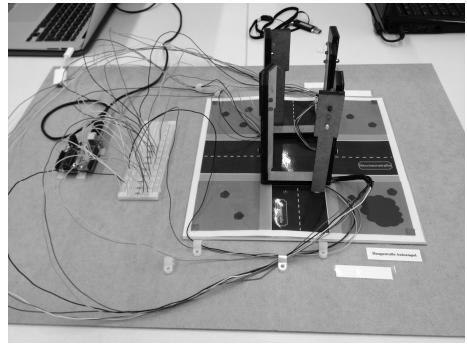


Abbildung 3: Kreuzungsmodell

Die offenen Aufgabenstellungen ermöglichen neben der Kreativitätsentfaltung auch ein individuelles Lerntempo, was speziell bei sehr heterogenen Informatikkursen von großer Bedeutung ist. Alle Aufgabenblätter beinhalten einen Pflichtteil, den alle SuS meistern sollen, damit ein sinnvoller Übergang zur nächsten Station möglich wird. Darüber hinaus enthält jede Station Möglichkeiten die Lösung beliebig weit zu verfeinern und zu verbessern, so dass auch bei schnellen Teams keine Langeweile entsteht. Die große Freiheit in der Arbeitsweise bedarf allerdings an einigen Stellen einer Sicherung, die zur Unterstützung des individuellen Lerntempos nicht in Form eines Unterrichtsgesprächs erfolgen kann. Die größte Überprüfung erfolgt in diesem Modul durch die Ampelschaltungen selbst, denn wenn das reale Modell alle Kriterien der jeweiligen Station erfüllt, kann davon ausgegangen werden, dass alles korrekt modelliert bzw. implementiert wurde. Zusätzlich zu der Selbstkontrolle über die Ampelmodelle wurden an einigen Stellen weitere Elemente zur Selbstkontrolle hinzugefügt. So ergibt beispielsweise die richtige Anordnung der Zustandskärtchen bei Station 1 und 2 ein

Lösungswort und erst mit diesem erhalten die SuS von den Betreuern das nächste Arbeitsmaterial.

5.4 Teil III – Freiarbeit

Dieser Teil dient der Sicherung und Vertiefung des in Teil I und II Gelernten, indem das Wissen auf eine neuartige, jedoch ähnliche Situation übertragen wird. Die 4er-Teams erhalten verschiedene Szenarien, für die sie eigenständig die drei oben beschriebenen Phasen des Aufbaus, der Modellierung und Generierung des Quelltextes durchlaufen. Dabei werden den SuS explizit keinerlei Vorgaben zur Ausgestaltung der Situation gemacht; diese müssen in den Teams selbstständig erarbeitet, diskutiert und festgelegt werden. Hierbei werden unter anderem Kompetenzen im Bereich Kommunikation und Kooperation bei den SuS gefördert.

5.5 Abschluss

Zum Abschluss des Moduls stellen alle Gruppen ihre Szenarien und selbst entwickelten Lösungen im Rahmen eines Museumsganges (vgl. [MK10]) der gesamten Gruppe vor. Dies soll zum einen das fachliche Diskutieren fördern und zum anderen den SuS die Wertschätzung ihrer eigenständigen Arbeit vermitteln.

6 Fazit

Dieses fachübergreifende Modul des InfoSphere – Schülerlabor Informatik der RWTH Aachen ermöglicht es SuS der Sekundarstufe I auf experimentelle Weise im Anwendungskontext Straßenverkehr den Nutzen der Modellierung zur Steuerung einer Ampelanlage selbstständig zu entdecken. Im Rahmen dieses 4,5 stündigen Moduls erlernen die SuS wie zustandsorientierte Modelle sowohl als strukturelle Hilfe für den Programmierer, als auch zur automatischen Generierung von Quellcode genutzt werden können. Der offene Aufgabencharakter ermöglicht neben den inhaltlichen Kompetenzen zusätzlich den Ausbau überfachlicher Kompetenzen im Bereich Kommunizieren und Kooperieren. Insgesamt kann dieses Modul dazu beitragen auch jüngeren SuS ein Gefühl dafür zu vermitteln, wie theoretische und praktische Aspekte in der Informatik auf einmalige Art und Weise zusammen fließen.

Literaturverzeichnis

[Be06] Bell, T.: Schülervorstellungen und Lernen von Physik. Forschendes Lernen, Kiel, 2006.

[BHS12] Bergner, N.; Holz, J.; Schroeder, U.: Concept of an Extracurricular Learning Environment for Computer Science. In (Knobelsdorf, M.; Romeike, R. Hrsg.): PRE-PROCEEDINGS 7th Workshop in Primary and Secondary Computing Education WiPSCE 2012, 2012; S. 26–33.

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

[Gu07] Guderian, P.: Wirksamkeitsanalyse außerschulischer Lernorte. Der Einfluss mehrmaliger Besuche eines Schülerlabors auf die Entwicklung des Interesses an Physik. Dissertation, Berlin, 2007.

[Hu07] Hubwieser, P.: Didaktik der Informatik. Grundlagen, Konzepte, Beispiele. Springer-Verlag, Berlin, Heidelberg, 2007.

[Le09] Lechner, G.: Bau und Programmierung einer computergesteuerten Ampelanlage, Imst, 2009.

[M ALB02] Aepkers, M. et al.: Entdeckendes, forschendes und genetisches Lernen. Schneider-Verl. Hohengehren, Baltmannsweiler, 2002.

[MK10] Müller, F.; Klippert, H.: Selbstständigkeit fördern und fordern. Handlungsorientierte und praxiserprobte Methoden für alle Schularten und Schulstufen. Beltz, Weinheim, 2010.

[Nu03] Nuhn, H.-E.: Die Sozialformen des Unterrichts: Pädagogik.

[Pe10] Peschel, F.: Offener Unterricht. Schneider-Verl. Hohengehren, Baltmannsweiler, 2010.

[We12] Werner-von-Siemens-Gymnasiums Regensburg: Objektorientierte Modellierung und Programmierung. Informatik 10 Startseite. http://www.schulen.regensburg.de/wvsgym/images/Faecher/Informatik/Informatik_10/informatik_10_start.html, 13.02.2013.

Kinder auf dem Weg zur Informatik: Roboter in der Grundschule

Christian Borowski
Universität Oldenburg
Department für Informatik
Didaktik der Informatik
christian.borowski@informatik.uni-oldenburg.de

Abstract: Dieser Praxisbericht beschreibt die Durchführung eines Unterrichtsvorhabens mit Lego-NXT-Robotern in einer dritten Klasse der Grundschule. Nachdem einige Jahre das Programmieren von Robotern im Rahmen einer Arbeitsgemeinschaft angeboten wurde, wurde in diesem Jahr ein Roboterprojekt über zwei Tage realisiert. Es wird über die organisatorischen Rahmenbedingungen, die Unterrichtsplanung, den durchgeführten Unterricht und den daraus gezogenen Konsequenzen berichtet. Das Projekt fand bei allen Beteiligten einen so großen Anklang, dass es nun geplant ist in allen dritten und vierten Klassen eine entsprechende Unterrichtsreihe durchzuführen. Durch die Bereitstellung der entsprechenden Rahmenbedingungen, wie Zeit, Material (Roboter und Unterrichtsmaterial) und Personal, ist es nun somit erstmalig möglich das Thema Programmieren von Roboter im Rahmen des Sachunterrichts für alle Schülerinnen und Schüler der Klassen 3 und 4 zu realisieren.

1 Einleitung

Nachdem einige Jahre das Programmieren von Robotern im Rahmen einer Arbeitsgemeinschaft angeboten wurde, wurde entschieden in diesem Jahr ein Roboterprojekt über zwei Tage (acht Unterrichtsstunden) für eine dritte Klasse zu realisieren. Es sollte herausgefunden werden, ob dieses Thema in einem vertretbaren Aufwand mit einer gesamten Klasse im normalen Unterricht realisiert werden kann. Es wird über die organisatorischen Rahmenbedingungen, die Unterrichtsplanung und die Konsequenzen aus dem durchgeführten Unterricht berichtet. Ein übergeordnetes Ziel ist es das Unterrichtsmaterial und den Unterrichtsprozess so gut zu strukturieren, dass letztendlich diese Unterrichtseinheit von engagierten Grundschullehrkräften durchgeführt werden kann.

2 Rahmenbedingungen

2.1 schulische Bedingungen

Das Projekt fand in einer dreizügigen Oldenburger Grundschule statt. Insgesamt hat die Schule zwölf Klassen und ca. 250 Schülerinnen und Schüler. Die Schülerinnen und Schüler kommen zu einem großen Teil aus bildungsfernen Elternhäusern. Die Schule zeichnet sich dadurch aus, dass sie seit Jahren integrativ und inklusiv arbeitet, dadurch hat sich in der Schule eine vielfältige Lehr- und Lernkultur entwickelt. Bei den Lernprozessen werden die Schülerinnen und Schülern nicht nur durch Lehrerinnen und Lehrer, sondern auch durch Erzieherinnen und Erzieher, ehrenamtlicher Helferinnen und Helfer und Eltern unterstützt. Einer dieser Mathematik- und Lesehelfer bietet schon seit Jahren einen Programmier- und Roboterkurs für eine Gruppe von 4-6 Schülerinnen und Schüler an und nahm auch an diesem Vorhaben teil. Regelmäßig kooperiert die Schule mit der Universität Oldnburg. Studentinnen und Studenten leisten ihre Fachpraktika ab. Zudem finden Praxisteile von Seminarveranstaltungen, z.B. im Fach Musik oder Sachunterricht, in der Schule statt. Insgesamt herrscht also in der Schule ein sehr offenes und kreatives Klima.

2.2 beteiligte Personen

An der Planung und Durchführung des Unterrichts waren beteiligt:

- die Klassenlehrerin,
- ein Bachelorstudent (Berufsziel Gymnasiales Lehramt, Fächer Informatik und Musik),
- ein ehrenamtlicher Mathe-Helfer und AG-Leiter und
- der Autor dieses Artikels.

Die dritte Klasse bestand aus 22 Schülerinnen und Schüler. Es handelte sich um eine Montessori-Klasse , dies bedeutet, dass die Klasse sehr gut auf eigenständiges Arbeiten, das Arbeiten in Gruppen und dem handelnden Umgang mit Materialien vorbereitet war. Vier Schüler aus der Klasse hatten bereit sein halbes Jahr vorher an der ÄG Roboter programmierenteilgenommen.

3 Unterrichtsplanung und -vorbereitung

Die Vorbereitungszeit für ein solches Unterrichtsvorhaben ist nicht unerheblich. Für einen klaren und konsistenten Unterrichtsablauf ist es unerlässlich, das Material angemessen vorzubereiten, die Aufgaben zu erproben und klare Arbeitsblätter für Musterlösungen zu erstellen.

Im Zentrum des Vorhabens standen Aufgaben aus der FIRST LEGO League (FLL) 2012 - Senior Solutions. Eine gute Übersicht über die Aufgaben findet sich auf Youtube im diesem Video[FLL].

Zuerst wurden die Aufgaben, bezüglich der Einfachheit und der Möglichkeit sie mit den Schülerinnen und Schülern zu lösen, analysiert. Am geeignetsten erschien die Aufgabe „Kegeln“, die zudem sehr motivierend erscheint. Diese Aufgabe sollte unsere Musteraufgabe werden, die jede Schülergruppe zu lösen hatte. Als Anschlussaufgabe eignete sich unseres Erachtens die Aufgabe „Stuhl reparieren“. Im Internet wurden einige Lösungsvideos recherchiert [VIDa, VIDb, VIDc, VIDd], sie zeigten wie man Aufgaben bei der FLL lösen kann.

Beim Bau des Roboters wurde auf eine Bauanleitung der Webseite WWW.NXTPROGRAMS.COM zurückgegriffen. Aus vorherigen AGs war bekannt, dass das Modell Express-Bot[EB] mit Schülerinnnen und Schüler innerhalb einer Unterrichtsstunde gebaut werden kann. Das Modell wurde mit einem Wurfarm modifiziert. Es wurde eine entsprechende Anleitung mit dem LEGO DIGITAL DESIGNER[LDD] erstellt. (Abbildung 1).

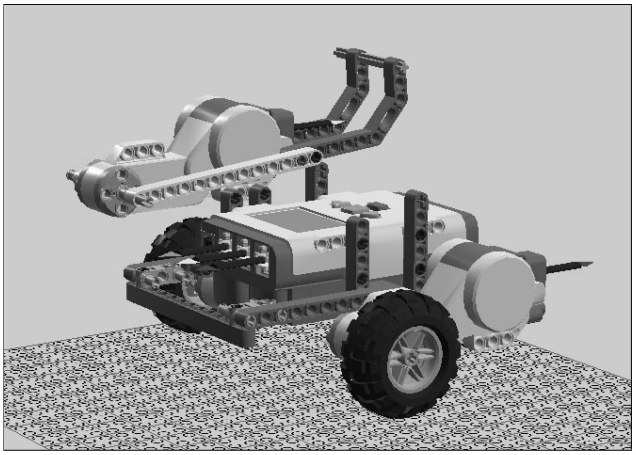


Abbildung 1: Ballwerfer im LEGO DIGITAL DESIGNER

Das Modell wurde von den Lehrkräften erprobt und ein entsprechendes Arbeitsblatt zum Programmieren des Roboters (Abbildung 2) wurde erstellt. Der Algorithmus wurde in zwei Formen formuliert, einmal so wie der Schüler ihn besser verstehen sollte:

1. Fahre von der „Base “bis zur „Holzfläche“.
2. Drehe dich in Richtung Kegel.
3. Werfe den Ball.

Zum anderen haben so formuliert, dass man besser versteht was zu programmieren ist:

1. Drehe die Räder an Motor A und C 9,7 mal nach vorne.
2. Drehe das Rad am Motor C 1,4 mal nach vorne.
3. Drehe den Motor B (Wurfarm) um 0,3 Umdrehung nach vorne.

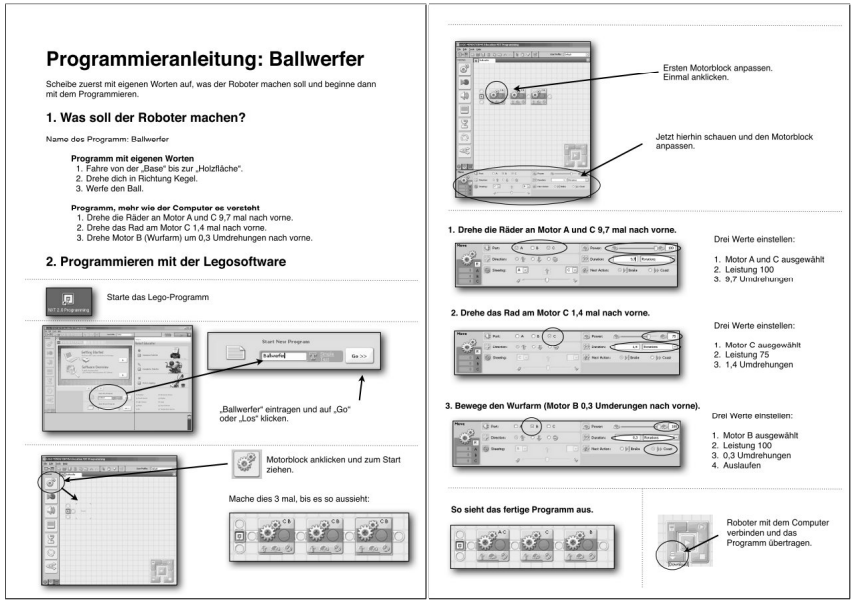


Abbildung 2: Programmieranleitung des Ballwerfers

Anschließend wurde eine Anleitung erstellt, die den Schülerinnen und Schüler zeigte, wo sie die Bauanleitung für den Roboter im Laptop finden und wie der LEGO DIGITAL DESIGNER zu verwenden ist (Abbildung 3).

Die Schülerinnen und Schüler sollten möglichst viel Freiraum und Entfaltungsmöglichkeiten haben, dennoch benötigten sie einen klaren Rahmen indem sie sich bewegen und ihre Erfahrungen machen können. Allerdings ist das Geben von Freiraum in der Grundschule nicht ohne Risiko, denn die Schülerinnen und Schüler verfügen noch nicht über hinreichend viel Erfahrungen, um Probleme effizient zu lösen. Zudem haben sie anders als Schülerinnen und Schüler in der Sekundarstufe I noch keine Strategien entwickelt, um sich selbstständig neue Lösungen, zum Beispiel aus dem Internet anzueignen, und so alternativen Sichtweisen zu bekommen. Alles muss selbst ausprobiert und erfahren werden, oft wird an einem Lösungsweg festgehalten, ohne neue Möglichkeiten zu betrachten. Dieses Vorgehen kostet sehr viel Zeit. Für die Lehrerinnen oder den Lehrer besteht ein hoher Aufwand sich jeweils in neue Denkwege und Ideen der Kinder einzudenken. Aus diesen Gründen wurde entschieden zu Beginn des Projektes möglichst viel Informationen zu geben, und dann die Gruppen möglichst frei und selbstständig arbeiten zu lassen. Durch die

hohe Anzahl der Erwachsenen waren genug Ressourcen vorhanden, um Kinder auch in individuellen Lernwegen und Ideen zu unterstützen. Jeder Erwachsene sollte, je nach seiner Erfahrung mit den LEGO-Robotern, ein bis zwei Gruppen unterstützen. Insgesamt sollten fünf Gruppen nach Zufallsauswahl (Ziehen im Kreis) gebildet werden, da diese Gruppenbildung von den Schülerinnen und Schülern als fair erachtet und akzeptiert wird. Bei dieser Gruppenbildung kommt es in der Regel zu weniger Diskussionen, im Gegensatz zu einer Einteilung durch die Lehrkraft. Es kann so schneller mit dem Arbeiten und den Lösen der Aufgaben begonnen werden.

Für den Unterricht standen je 4 Unterrichtsstunden an zwei Schulvormittagen zur Verfügung. Es konnte der Klassenraum und zwei Gruppenräume genutzt werden. Am zweiten Tag stand ebenfalls das Forum der Schule zur Verfügung.

Aus diesen Überlegungen heraus ergab sich für die zwei Tage folgende Unterrichtsplanung:

Tag 1:

	Geplanter Unterrichtsschritt
1. Stunde	- Begrüßung und Vorstellung aller Beteiligten
	- Besprechung des Vorhabens
	- Vorstellung der Aufgaben am Spielfeld
	- Einteilung der Gruppen
2. Stunde	- Verteilen des Materials
	- Lösen der Aufgabe „Kegeln“
	- Bau des Roboters nach Anleitung
	- Programmieren des Roboters nach Anleitung
	- Testen des Roboters und der Programmierung
3.-4. Stunde	- Auswahl weiterer Aufgaben
	- Entwicklung eigener Lösungen

Tag 2:

	Geplanter Unterrichtsschritt
1.-2. Stunde	- Arbeit an eigenen Lösungen
3.-4. Stunde	- Präsentation der eigenen Lösungen im Klassenverband
	- Aufräumen

Nach der abgeschlossenen Planung wurde das Material sortiert. Jede Gruppe sollte eine Kiste mit allen benötigten Teilen zum Bau des Ballwerfers, einen Laptop, eine Anleitung zum Finden der Bauanleitung und eine Programmieranleitung für den Ballwerfer bekommen. Alle weiteren und zusätzlichen Teile wurden in handelsüblichen Sortierkästen bereitgestellt.

Zusammenfassend vollzog sich die Unterrichtsplanung in folgenden Schritten und Zeiteinheiten:

- Analyse und Auswahl der Aufgaben, die von den Schülern eventuell gelöst werden können (1 Stunde),

- Suche auf Youtube auf nach Lösungen(1-2 Stunden),
- Bau eines Modells, welches die Aufgabe Kegeln löst (1-2 Stunden),
- Umsetzen dieses Modells in den LEGO DIGITAL DESIGNERS (2-3 Stunden),
- Programmiern und Testen des Modells (1 Stunde),
- Erstellen der Anleitungen „Wo finde ich die Bauanleitung“und „Programmierung des Ballwerfers “(1-2 Stunden) und
- Sortieren und bereitlegen des Materials (1 Stunde).

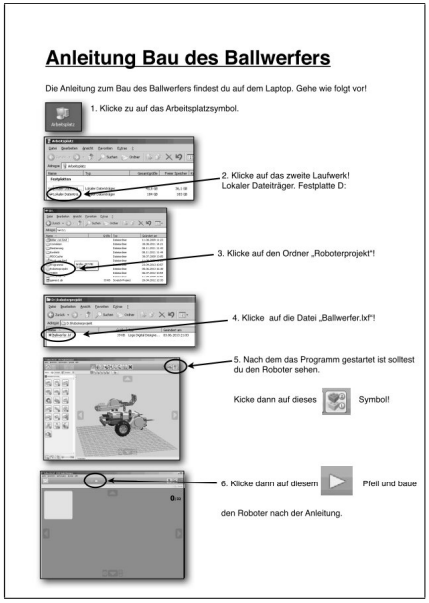


Abbildung 3: Anleitung: Bau des Ballwerfers

4 Unterrichtsverlauf

Die Schülerinnen und Schüler waren sehr motiviert. Das Erklären der Aufgaben, die Gruppenbildung, das Verteilen des Materials, das Bauen und Programmieren des Roboters verliefen insgesamt gemäß der Planung. Der Bau des Roboters, die Programmierung und das Austesten der Musterlösung dauerten bei den Gruppen unterschiedlich lange. Während eine Gruppe bereits nach 35 Minuten fertig war, benötigte eine andere fast 2 Unterrichtsstunden und die Unterstützung einer Lehrperson.

Anschließend begannen die Gruppen sich unterschiedlichen Aufgaben zuzuwenden. Eine Gruppe entschloss sich die Aufgabe „Stuhl reparieren“zu lösen. Mit wenig Unterstützung löste diese Gruppe die Aufgabe bis zum Ende des ersten Tages. Eine weitere Gruppe entschloss sich sich stärker mit dem Ballwerfer zu beschäftigen. Die Schülerinnen und Schüler entschieden sich ein Tor aufzubauen in das der Roboter den Ball werfen sollte. Dieses Tor wurde außerhalb des FFL-Spielfeldes aufgebaut. Eine Gruppe wollte unbedingt die Aufgabe „Flaschen holen“lösen. Bei dieser Aufgabe muss der Roboter grüne und orange LEGO-Steine unterscheiden. Da es nötig ist mit dem Lichtsensor oder Farbsensor zu arbeiten, es aber keine erstellten Anleitung dafür gab, wandte sich der Mathehelfer dieser Gruppe zu und erarbeitete mit ihnen kleinschrittig eine Teillösung. Der Roboter war zwar nicht in der Lage die „Flaschen“zu holen, aber wenigstens zu erkennen und zu unterscheiden, ob es eine orange oder grüne Flasche war. Eine Gruppe beschäftigte sich mehr mit dem Bauen und Umbauen des Roboters als mit der zielgerichteten Lösungsfindung einer Aufgabe. Auch Interventionen oder Hilfestellungen durch Lehrkräfte führten nicht dazu, dass die Schülerinnen und Schüler sich ausdauernd mit dem Bauen und dem Programmieren beschäftigten. Diese Gruppe war stärker mit dem konstruieren und Zusammenstecken der LEGO-Teile beschäftigt. Eine Gruppe hatte starke Probleme zusammenzuarbeiten, da ein Schüler nicht in der Lage war, sich mit seine Mitschülerinnen und Mitschülern zu arrangieren.

Da der Eindruck entstanden war, dass die Gruppeneinteilung für zwei Gruppen etwas unglücklich war, wurde am Beginn des zweiten Tages die Gruppenzusammenstellung thematisiert. Nach einer ca. 20-minütigen Diskussion wurde eine Gruppenaufteilung gefunden, bei der alle Schülerinnen und Schüler bekundeten, dass sie so gut zusammenarbeiten könnten. Erfreulicherweise haben nur ein Schüler und zwei Schülerinnen die Gruppen getauscht, sodass die Gruppen relativ stabil blieben und mit den gebauten Robotern weitergearbeitet werden konnte. Die ersten zwei Unterrichtsstunden des zweiten Tages arbeiteten die Schülerinnen und Schüler weiter an ihren selbstgewählten Aufgaben. Zum Ende der zweiten Unterrichtsstunde des zweiten Tages breitete sich Unruhe in den Gruppen aus. Das Interesse richtete sich u.a. in vielen Gruppen auf die Sensoren, die mit beigelegt waren. Deshalb wurde vom Plan abgewichen und der Ultraschallsensor wurde einzuführt. Die gesamte Klasse wurde im Forum der Schule versammelt und es wurde mit Hilfe eines Beamers die Funktionsweise und Programmierung des Ultraschallsensors erklärt. Es wurde ein Fahrroboter gebaut, der vor der Wand stoppt. Alle Gruppen realisierten in kurze Zeit den Umbau des Roboters und die Programmierung. In den leistungsstarken Gruppen wurde das Programm schnell so modifiziert, dass der Roboter sich umdrehte und wieder zurück fährt. Jede Gruppe präsentierte vor der Gesamtgruppe ihr Programm. Zufällig kamen Schülerinnen und Schüler einer zweiten Klasse vorbei, diesen wurde stolz die Roboter und die Programme präsentiert.

5 Ergebnis des Unterrichtsvorhabens und weitere Schritte

Die Schülerinnen und Schüler zeigten ein starkes Interesse an diesem Projekt und hätten es gerne noch weiter fortgesetzt. Die Lernleistungen der Schülerinnen und Schüler sind sehr

differenziert zu betrachten. Für manche Schülerinnen und Schüler stand das konstruieren und bauen mit Lego im Vordergrund, andere vertieften sich in Aufgaben der FLL und versuchten eine nach der anderen zu lösen. Im Rahmen dieses Projektes wurden Audio- und Videoaufnahmen gemacht, diese wurden jedoch nicht systematisch ausgewertet. Für die nächste Durchführung würde sich anbieten in Absprache mit der Klassenlehrerin einige Schülerinnen und Schüler speziell in den Fokus zu nehmen (Fokusschüler) und gezielt zu beobachten.

Innerhalb der Schule erregte dies Projekt so viel aufsehen, dass es nun einfach ist, mit der Unterstützung der Schulleitung und den Grundschullehrkräften dieses Unterrichtsvorhaben auch in den Parallelklassen und dem Jahrgang darunter durchzuführen. Sollte dies gelingen haben am Ende des nächsten Schuljahres alle Schülerinnen und Schüler der 3. und 4. Klassen erfahren, was es heißt einen kleinen Roboter zu programmieren.

Der Aufwand an Personal war bei diesem Vorhaben sehr hoch, dies scheint aber gerechtfertigt, da sich auf unbekannten Gebiet bewegt wurde. Im den folgenden Durchführungen werden vermutlich 2-3 Lehrpersonen genügen. Viel Zeit beanspruchte auch die Erstellung des Modells, sowie das Entwickeln der Arbeitsblätter und der Musterlösung. Auf Sensoren wurde bewusst verzichtet, da die Lösung der einfachen Aufgaben auch ohne Sensoren möglich wäre. Das Modell und Material sind jetzt da und es werden weitere Aufgaben und Anleitungen auch zum Thema Sensoren entwickelt.

Literatur

[EB] Express-Bot. http://www.nxtprograms.com/9797/express-bot/index.html. Letzter Zugriff: 30.6.2012.

[FLL] FLL 2012 Senior Solutions Robot Game Aufgaben. http://www.youtube.com/watch?v=tS6QUsf9XYQ. Letzter Zugriff: 30.6.2013.

[LDD] LEGO Digital Designer. http://ldd.lego.com/de-de/. Letzter Zugriff: 30.6.2013.

[VIDa] FLL 2012 Senior Solutions 400 point tournament run. http://www.youtube.com/watch?v=e-JgcFt7zj0. Letzter Zugriff: 30.6.2012.

[VIDb] FLL Senior Solutions - Easy 300 points. http://www.youtube.com/watch?v=4XOsCz0wVEE. Letzter Zugriff: 30.6.2013.

[VIDc] 570 Points! Mindfactory at the South-West European Semi-Final, http://www.youtube.com/watch?v=cUkFDwq92eQ. Letzter Zugriff: 30.6.2013.

[VIDd] 460 Points in 2.5 min, FLL 2012 Senior Solutions. http://www.youtube.com/watch?v=TsRm_6mNp40. Letzter Zugriff 30.6.2013.

Erst nehmen wir Greenfoot. Und dann BlueJ?

Etwas mehr als ein Bericht aus der Praxis

Dieter Engbring

FG Didaktik der Informatik
Universität Paderborn
Fürstenallee 11
33102 Paderborn
didier@upb.de

Abstract: Mit diesem Bericht aus der Praxis wird der Versuch unternommen, die Diskussion um Entwicklungsumgebungen aus der Perspektive der softwaretechnischen Unterstützung von Lernprozessen zu betrachten. Denn durch die Umgebungen scheint sich ein heimlicher Lehrplan zu konstituieren, der eben nicht das Modellieren sondern das Implementieren in das Zentrum des Unterrichts rückt. Dies ist ein Mangel, der durch eine andere, bessere Gestaltung solcher Umgebungen als Einbettung in die Lernprozesse geheilt werden kann.

1 Einleitung

Auch mir als Informatiker fällt es inzwischen schwer den Überblick zu behalten, mit welchen Umgebungen man in den Informatikunterricht einführen kann bzw. sollte. Deren Bewertung fällt schwer, da sie alle spezifische Vor- und Nachteile haben aber ein beständiger Wechsel nicht möglich ist.[1] Kernbestandteil dieses Praxisberichtes ist es, die Erfahrungen, die ich selbst (aber durchaus in Gesprächen mit Kollegen reflektiert) und über die Beobachtungen im Rahmen einer Examensarbeit mit Greenfoot und BlueJ gemacht habe, nicht einfach nur nachzuerzählen, sondern derart zu strukturieren und dann zu verallgemeinern, dass daraus Bewertungskriterien entstehen könn(t)en.[2] Bevor in den Bericht eingestiegen wird, muss zunächst geklärt werden, unter welchen Rahmenbedingungen die dann zu schildernden Beobachtungen gemacht wurden, bevor dann der eigentliche Bericht erfolgt. Abschließend wird neben einem Fazit auch ein Ausblick auf die anstehenden Forschungsaufgaben gegeben.

2 Prämissen und Annahmen

Dieser Beitrag ist unter einigen Prämissen geschrieben, die aus Platzgründen nicht diskutiert werden können, die aber zugegebenermaßen diskussionswürdig sind. Die erste Prä-

[1] Dies muss an dieser Stelle so allgemein bleiben, wird aber weiter unten noch konkretisiert.

[2] Was dann aber einen Forschungsansatz erfordert, der zum einen neu ist und zum anderen noch nicht systematisiert wurde.

misse besteht in der Annahme (bzw. Behauptung), dass Informatikunterricht tatsächlich kein Programmierkurs sein kann bzw. soll. Prozesse der Modellierung sollen im Vordergrund stehen. Die Kompetenzen, die die Schülerinnen und Schüler (im folgenden Schüler) erwerben sollen, müssten sich auf die informatischen Modelle beziehen. Eine Beobachtung, die in diesem Bericht dargestellt wird, ist jedoch, dass auch mit der besten (m.a.W. entgegengesetzten) Absicht, tatsächlich doch ein Programmierkurs – als heimlichen Lehrplan – unterrichtet wird, und dies auch an den eingesetzten Umgebungen liegt.

Die zweite Voraussetzung besteht in der Behauptung, dass sich *over all* keine vergleichende Bewertung der Lernwirksamkeit der Umgebungen möglich sein wird. Die Lernwirksamkeit solcher dann letztlich Lernumgebungen ist von zu vielen Faktoren abhängig, als dass sich diese in der täglichen Praxis kontrollieren ließen. Mit Laborexperimenten würde man sich hingegen zu weit von der Einsatzwirklichkeit entfernen, wodurch dann die darin gewonnenen Erkenntnisse nicht auf die tägliche Praxis übertragen werden könnten. Aus der täglichen Praxis, die im folgenden *Alltagspraxis* genannt wird, ergibt sich eine Beobachtungsperspektive, die im folgenden darzustellen sein wird. Es ist eine Perspektive der Nutzung von Software, (*Usability)*, die bislang m.E. im Kontext didaktischer Forschung nicht untersucht wurde. Bei der Gestaltung der entsprechenden Hilfsmittel (dies sind in der Tat keine Werkzeuge) haben Kriterien der Brauchbarkeit im Unterricht – so die Annahme dieses Beitrages – die auch zum Teil belegt werden kann, bislang eine eher untergeordnete Rolle gespielt. Sie sind eher für den Prozess der Programmierung (bzw. Implementierung) geschaffen.

Die vierte Prämisse geht davon aus, dass der in Abbildung 1 dargestellte Zyklus nicht nur die Zusammenhänge von Modellierung, Implementierung und Evaluation darstellt. Er zeigt auch auf, dass man im Informatikunterricht verscheidene Denktraditionen und -strukturen miteinander verbinden kann. Dies ist wohl ein wesentlicher Beitrag, den die Informatik zur Allgemeinbildung leisten kann, aber wohl nur dann, wenn man diesen Zyklus vollständig durchläuft, was dann die Realisierung des Modells und dessen Evaluation mit beinhaltet. Leider kommt es in diesem Zyklus zu Brüchen und zusätzlichen Aufwänden.

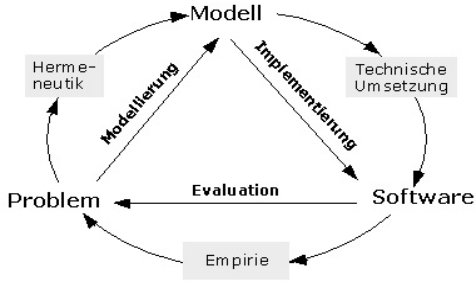


Abbildung 1: Zyklisches Vorgehen

Die Idee zu dieser Untersuchungsperspektive geht auf Beobachtungen zurück, die sich bei der Nutzung von BlueJ mit verschiedenen didaktischen Konzepten als Invariante gezeigt haben: Die Schüler scheitern oftmals nicht bei oder an der Modellierung; sie scheitern (bzw. tun sich schwer) bei der Implementation der Modelle. Dieses Scheitern – und das ist die Hypothese – kann darauf zurückgeführt werden, dass BlueJ in der Tat das Umsetzen von in der Regel außerhalb (z.B. mit Bleistift und Papier) entstandenen Modellen (hierzu zählen nicht nur Klassendiagramme sondern auch Pseudo-Code, Ablaufpläne, ggf. Aktivitätendiagramme oder state-charts) nicht oder nur sehr rudimentär unterstützt. Wohingegen andere Umgebungen, wie z.B. der Java-Editor oder auch Scratch,

bzw. Alice hierfür (partiell) Lösungen anbieten.[3] Deren Nutzung ist jedoch aus anderen – hier nicht ausführlich darstellbaren – Gründen nicht möglich oder nicht ratsam.

3 Eine Sichtweise auf Informatikunterricht

Nach diesen Vorüberlegungen, die die Motivation für die Beobachtungsperspektive auf den Unterricht definiert, wird als erstes gezeigt, unter welcher Perspektive man erfahrungsgestützt sich dem Phänomen der Mensch-Maschine-Interaktion im Informatikunterricht, der in der Regel in Computerräumen stattfindet, widmen kann. Es ist die Perspektive aus dem Bereich des E-Learning, die wiederum in der Software-Ergonomie verwurzelt ist. Schlüsselbegriffe sind Medienfunktionen bzw. Medienbrüche.

3.1 Medienfunktionen

Im Informatikunterricht ist die Nutzung von Computern und Software zwar üblich, aber nicht unbedingt lernförderlich. Die Autoren des CS unplugged Ansatzes weisen darauf hin: „We have found that many important concepts can be taught without using a computer—in fact, sometimes the computer is just a distraction from learning“ [BW06, S. i] Daraus kann man nicht – und es ist auch wohl nicht die Absicht der Autoren – ableiten, dass man im Informatikunterricht grundsätzlich den Stecker ziehen sollte. Es gibt Themenbereiche, die den Computereinsatz geradezu erfordern.[4] Die von den Autoren des CS unplugged Ansatzes beschriebene Ablenkung durch die Technik ist kein grundsätzliches Problem sondern auch eines einer unzureichenden Gestaltung der technischen Hilfsmittel, die die Spezifika der unterrichtlichen Prozesse zu wenig in den Blick nehmen.Sie sind zum einen eher vom Software-Entwicklungsprozess inspiriert. Zum anderen fokussieren sie eher das Endergebnis (ein Stück Software aufgeteilt in Klassen) und eben nicht die (unterrichtlichen) Prozesse dahin zu kommen.

Bei der Beobachtung des Unterrichts sollen daher im Wesentlichen die Prozesse und nicht die Ergebnisse der Prozesse in den Blick genommen. Beispiele für solche Prozesse sind: Die Schüler erstellen Medienobjekte in Form von Klassen- bzw. Ablaufdiagrammen oder Programmcode. Sie benennen Eigenschaften bzw. Aufgaben von Objekten (die dann z.B. auf CRC-Karten oder in Klassendiagrammen notiert werden) oder sie verbalisieren eine Problemlösung in Umgangssprache, die dann vielleicht entsprechend notiert oder in Pseudo-Code transformiert wird. Dies alles geschieht oftmals in nicht digitaler Form, da z.B. BlueJ es – wie viele andere Umgebungen auch – nicht vorsieht, entsprechende digitale Medienobjekte zu erzeugen.[5] Nutzt man nun solche Hilfsmittel, die die das Erstellen solcher Medienobjekte unterstützt, muss man in deren spezifische Nutzung einführen, ohne dass man dieses digitale Objekt in der Programmierumgebung nutzen könnte. Man muss es manuell übertragen. Es kommt zu *Medienbrüchen,*.Zeit kosten. Die hierfür nötigen Prozesse werden zum Lerninhalt. Es entsteht ein *heimlicher Lehrplan*.

---

[3] Vielleicht klärt diese Aussage auch den ersten Allgemeinplatz mit den spezifischen Vor- und Nachteilen.

[4] Die Grundzüge des informatischen Modellierens können nicht ohne Rechnerunterstützung erlernt werden.

[5] Natürlich könnte man die umgangssprachliche Lösung als Kommentar festhalten, aber das macht den Prozess auch nicht einfacher zu durchschauen, zumal Kommentare eigentlich anders aussehen sollen.

Bei solchen *Medienbrüchen* setzt der Ansatz von Keil an, der ursprünglich für den Bereich der Software-Ergonomie entwickelt wurde und heute im wesentlichen im Bereich des E-Learning genutzt wird [Ke90]. Anknüpfend daran wurde von Keil und seinen Mitarbeitern das Konzept der *Medienfunktionen* entwickelt. Von Interesse im Zusammenhang mit diesem Bericht sind vor allem die *primären Medienfunktionen*. Als primäre Medienfunktionen werden folgende Prozesse genannt: „erzeugen, löschen, arrangieren, verknüpfen, übertragen, zugreifen und synchronisieren" [Se08]. Diese Prozesse werden in verschiedenen Phasen des Informatikunterrichts von Schülern und Lehrern durchgeführt. Dies betrifft das Erstellen eines UML-Diagramms wie auch die Programmierung einer Klasse (Medienfunktionen: *erzeugen*, *löschen* und *arrangieren*). Darüber hinaus spielen aber auch das Veröffentlichen oder der Austausch von Lösungen, z.B. zur Bewertung durch den Lehrer oder im Rahmen einer Projektarbeit (Medienfunktionen: verknüpfen, übertragen, zugreifen und synchronisieren) eine Rolle. Viele andere Beispiele könnten an dieser Stelle gegeben werden. Daraus ergeben sich zwei Perspektiven für die Unterrichtsbeobachtung. 1. Wo entstehen Medienbrüche? und 2. Wie (bzw. zu welchen Grad) werden die primären Medienfunktionen unterstützt?

Bevor von den auf diese Sichtweise bezogenen Beobachtungen und Befunden berichtet wird und diese dann interpretiert werden, soll aber zunächst erläutert werden, warum sich die Beobachtungen auf den Einsatz von *Greenfoot* und *BlueJ* beziehen, die ganz offensichtlich viele der Medienfunktionen nicht sehr gut unterstützen. In dieser Aufzählung ist bereits erkennbar, dass (der Editor von) BlueJ gerade auch die Medienfunktionen erzeugen, löschen und arrangieren nur auf der Basis textueller Eingabe bzw. entsprechender Manipulationen unterstützt. Es gibt dennoch gute Gründe auf diese beiden Umgebungen aufzusetzen.

3.2 Exkurs: Warum Greenfoot und dann BlueJ?

In den Jahren zuvor wurde an den betreffenden Schulen nach dem Umstieg auf die Sprache Java, zunächst *BlueJ* nach dem Konzept von Barnes und Kölling [BK03] eingesetzt. Danach wurde versucht, über den Stifte-und Mäuse Ansatz von Schriek [Sc05] einzusteigen. Jeweils konnte beobachtet werden, dass sehr früh die Motivation sich mit den Inhalten des Unterrichts zu befassen bei einer ganzen Reihe von Schülern nachließ und es anderen nicht schnell genug voranging. Eine Mini-Welt vorzuschalten, schien bevor *Greenfoot* vorgestellt wurde [HK04], aber auch keine Alternative zu sein, da man diese – gleichgültig, welche man genommen hätte – irgendwann verlassen muss und dadurch ebenfalls Motivation dämpfende Brüche entstehen, da sich die Schüler dann auf eine neue Software einstellen müssen und den Umgang mit dieser erlernen müssen, was wiederum nicht nur Zeit kostet sondern auch von den eigentlichen Lernzielen ablenkt.[6]

Die auf den ersten und zweiten Blick durchaus motivierende Umgebung Alice kann aufgrund anderweitig gesammelter Erfahrungen kaum genutzt werden, da sich aus der drei dimensionals erscheinenden Darstellung weitere Probleme ergeben und insgesamt sogar

[6] Auch das ist hier nur sehr grob darstellbar. Es fehlt der Platz dieses genauer zu erläutern. Was damit z.B. gemeint ist, wird in Abschnitt 4.3. deutlich, in dem von den Schwierigkeiten beim Übergang Greenfoot zu BlueJ die Rede ist.

dazu verleitet Fehlvorstellungen in Bezug auf die grundlegenden Begriffe Objekt und Klasse (Vgl. hierzu: [DM09], [En11] und [UC10]) zu erhalten.[7]

So wird seit drei Jahren der Einstieg mit Greenfoot realisiert, das als eine Art Mini-Welt auch nur begrenzt weit trägt, aber den Vorzug bietet, dass Greenfoot und BlueJ den selben Programm-Editor nutzen. So sollte – das war die Hoffnung – der Umstieg von Mini-Welt zu tatsächlicher Programmierumgebung eigentlich keinen Bruch darstellen. Für die Nutzung liegt außerdem ein Schulbuch vor [KT10], in dem ein weitestgehend überzeugendes Konzept verfolgt wird. Das Schulbuch arbeitet mit Roboter-Szenario, das *Karol, the robot* [Pa81] nachempfunden ist.

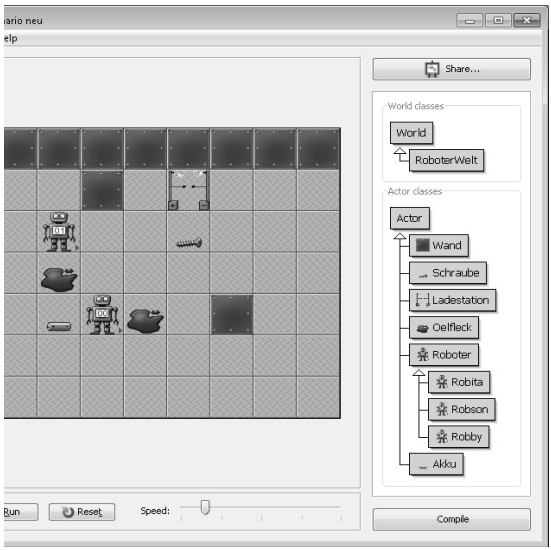


Abbildung2: Greenfoot mit Klassenhierarchie

Die Abbildung 2 liefert einen Eindruck davon, was diese spezielle Welt beinhaltet. *Greenfoot* besitzt zwei Klassen (Actor und World) von denen alles andere abgeleitet wird. Konstituierend für das jeweilige Szenario ist die von *World* erbende Klasse in diesem Fall die *RoboterWelt* als Hauptprogramm. Gefüllt wird das Szenario mit Objekten, die aus den jeweiligen Unterklassen von *Actor* instanziiert werden. Die Klassen Actor und World können via *Greenfoot* nicht manipuliert werden.

[7] Die Nutzung des Java-Editors scheiterte jedoch daran, dass dieser auf der Plattform Solaris nicht verwendet werden kann und auf dem Windows-Server-System nicht so installiert werden konnte, so dass er genutzt werden könnte.

4 Beobachtungen und Befunde

Die nun darzustellenden Beobachtungen und Befunde beziehen sich auf drei Phasen im Verlauf des ersten Halbjahrs (oder ein wenig darüber hinaus), die über drei Jahre gesammelt wurden. Es geht erstens um die Einstiegsphase, bei der es um die Einführung und die vertiefte Benutzung der Kontrollstrukturen geht. Zweitens wird berichtet wie die Schüler, das im Schulbuch vorgegebene Roboter-Szenario durch eigene Klassen bereichern und drittens geht es um den Wechsel von Greenfoot zu BlueJ. Viele der hier dargestellten Befunde stammen aus eigenen unterrichtspraktischen, den Beobachtungen anderen Unterrichts mit anschließenden reflektierenden Gesprächen und aus Beobachtungen, die im Rahmen einer Staatsexamensarbeit durchgeführt wurden [Sc11]. Die nun folgenden Unterabschnitte enthalten auch immer Bemerkungen zum didaktischen Konzept, dessen Gestaltung möglicherweise auch ursächlich für die Beobachtungen und Befunde sein könnte (siehe auch die einleitende Bemerkung zur Untersuchung der Lernwirksamkeit und Seiteneffekten).

4.1 Einstiegsphase

Nach ersten Gehversuchen im Umgang mit dem Szenario, bei denen zunächst nicht nur die Objekte durch Mausclick in die Welt gesetzt werden, sondern auch die bereits bestehenden, vorgefertigten Methoden auf die instanziierten Objekte angewandt werden, ist eine große Motivation großer Teile der Schüler feststellbar. Es entsteht hier der Wunsch den Robotern komplexere (dies ist relativ zu sehen) Aufgaben „beizubringen“. Dieses „Beibringen“ kann einerseits durch Programmieren der zunächst leeren Methode *act* erfolgen oder durch das Schreiben neuer Methoden. Der Fokus liegt darauf, die algorithmischen Grundlagen des Modellierens zu vermitteln: Sequenzen, Fallunterscheidungen und Wiederholungsanweisungen. Es ist ein Leichtes sich Aufgaben auszudenken, bei denen die Roboter verschiedene Fälle unterscheiden müssen oder Anweisungsblöcke wiederholen sollen. Das Buch beinhaltet entsprechende Aufgaben; die Schüler entwickeln aber auch eigene Ideen.

Die Programmierung eigener Methoden oder auch der Methode *act* wird allerdings nicht besonders gut unterstützt. Eine Aufzeichnungsfunktion der interaktiven Manipulation der Objekte wäre für den weiteren unterrichtlichen Verlauf sehr hilfreich, da das Ergebnis der Aufzeichnung reflektiert und weiterverarbeitet werden könnte.

Die Probleme, die in dieser Phase auftreten, scheinen relativ banal zu sein,[8] beeinflussen aber den weiteren Unterricht ganz entscheidend. Sie hängen in zweifacher Hinsicht im Wesentlichen mit den Medienfunktionen des Erzeugens (sowie des Löschens und des Arrangierens) zusammen. Zum einen werden die Schüler durch das Schulbuch angehalten bei den Kontrollstrukturen auch Programmablaufpläne zu zeichnen, was aber auf jeden Fall außerhalb von Greenfoot erfolgt. Es bietet weder die Möglichkeit solche Diagramme zu erstellen, noch diese importieren. Hier liegt ein Medienbruch vor. Zum anderen betrifft dies die Codeerstellung. Bei dieser muss penibel auf Groß- und Kleinschreibung geachtet werden. Dies ist z.B. bei Pascal bzw. Delphi nicht nötig und stellt eine zu-

[8] Dies ist aber die abgeklärte Sicht derer, die Programmieren können.

sätzliche Barriere dar, da der Code zeichenweise (Buchstabe für Buchstabe) eingegeben werden und nicht etwa wie in Alice, Scratch oder auch dem Java-Editor per *Drag and Drop* oder auch durch *Click* auf eine Schaltfläche.

Im Zusammenspiel mit der exakten Beachtung von Groß- und Kleinschreibung sowie der umständlichen Unterstützung auf bereits definierte Methoden zugreifen zu können, ergeben sich Hürden, die die anfängliche Motivation bei einigen Schülern rasch absinken lässt.[9] Warum – so fragt man sich – gibt es in BlueJ keine entsprechenden Schaltflächen zu Erzeugung von Programm-Code wie z.B. im Java-Editor.

Es ist insbesondere auch beim ersten Durchgang zu beobachten gewesen, dass auch das relativ kleinschrittige Vorgehen des Schulbuches Motivation geraubt hat. Viele Schüler haben vergeblich nach einem Sinn in den vielen kleinen Aufgaben gesucht. Deswegen ist im zweiten Durchlauf diese Phase des Erlernens und des Erkennens algorithmischer Strukturen durch das Ziel motiviert worden, einen Roboter am Ende der Phase „autonom" durch ein Labyrinth laufen lassen zu können, aus dem es sicher einen Ausweg gibt. Es hat sich aber gezeigt, dass es hier Probleme mit der Erkennung von während des Programms abgelegten Objekten (z.B. Schrauben zur Wegmarkierung) gibt, die weder zu antizipieren noch zu lösen waren.[10] So konnte eine höhere Motivation beim Erlernen der algorithmischen Strukturen beobachtet werden, die dann aber durch die Probleme der Realisierung nachdrücklich gedämpft wurden.

4.2 Entwicklung weiterer Klassen

In einer zweiten Phase geht es darum, weitere Objekte über neue Klassen der Roboterwelt hinzufügen. Auch hier generieren die Schüler von sich aus interessante, mehr oder weniger leicht zu realisierende Ideen aber auch das Lehrbuch bietet wieder solche Aufgaben. Neben den schon geschilderten Problemen, kommen nun weitere hinzu. Auch die (UML-)Klassendiagramme müssen außerhalb des Programms erzeugt werden; die Umsetzung, der wiederum oftmals mit Papier und Stift erzeugten Diagramme kostet viel Zeit, dessen größter Anteil aus Routinetätigkeiten besteht, wie z.B. dem Setzen von Methodenköpfen, von geschweiften Klammern und dem Schreiben einer *return* Anweisung Etwas später kommen dann auch beim Erzeugen von *get*- und *set*-Methoden weitere Routinehandlungen dazu, die die kreativen Prozesse regelmäßig ausbremsen.

An dieser Stelle geht wertvolle Unterrichtszeit verloren. Die immer gleiche korrekte Übertragung eines Klassendiagramms in Programm-Code wird zum eigentlichen Lerninhalt.[11] Auch hier wird wieder die Medienfunktionen des Erzeugens (inklusive Löschens und Arrangierens) nur sehr schwach unterstützt.

---

[9] Natürlich lässt sich hier bereits beobachten, dass Schüler ebenso mit dem abstrahierenden und logischen Denken Probleme haben. Dieses Problem lässt sich allerdings kaum softwaretechnisch lösen.

[10] Das Problem scheint in der Implementierung der Klassen *World* und *Actor* zu liegen.

[11] Diese als Hausaufgabe zu stellen, ist nicht keine Alternative, aber das kann hier nicht dargelegt werden.

Der Dialog zur Erzeugung einer Klasse (s. Abbildung 3) setzt diesbezüglich wohl falsche Prioritäten, da er sich im wesentlichen auf das (Erscheinungs-)Bild der abgeleiteten Objekte bezieht, nicht aber auf das Erstellen anderer Attribute.[12]

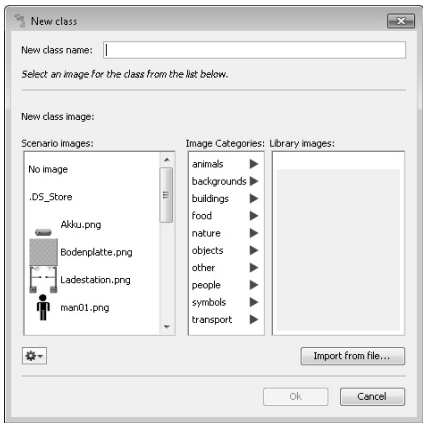


Abbildung 3: Klassenerzeugung mit Greenfoot

Ein Resultat dieser bis hierhin geschilderten Schwierigkeiten ist das Anwachsen der Heterogenität der Lerngruppen, die auch (nicht nur s. Fußnote 9) in handwerklich technischen Problemen begründet liegt. Möglicherweise kann hier durch eine bessere technische Unterstützung der Medienfunktion „erzeugen" und mithin auch „arrangieren und löschen", die es in anderen Umgebungen gibt – so zumindest die unbewiesene Vermutung zu diesen Beobachtungen – bei der Modellierung und im Übergang zu Codierung hier Abhilfe schaffen.

4.3 Der Wechsel zu BlueJ

Das Lehrbuch sieht vor, dass man, wenn Greenfoot ausgereizt ist, auf BlueJ umsteigt. Dies ist nötig, da man über die bis hierhin erworbenen Kompetenzen im Bereich der Objektorientierung und der Algorithmik hinaus gehen will. Die Lehrpläne und Vorgaben sehen vor, dass man darüber hinaus geht und z.B. Einblick in die Herstellung von Software bekommt. An dieser Stelle sind viele weitere Effekte zu beobachten.

Für den Umstieg modellieren die Schüler ein relativ einfaches Würfelspiel mit dem Namen „Verflixte Sieben". Nach der Modellierung, die ohne größere Probleme abläuft, bei der – für einen ersten Entwurf – im sog. „Objektspiel" ein brauchbares Klassenmodell (in Form eines Klassendiagramms) entsteht, zeigen sich erwartete und nicht erwartete Probleme. Zum einen treten die oben geschilderten Probleme bei der Erzeugung von Klassen noch einmal und deutlicher hervor. Viele Schüler haben zusätzlich große Probleme die Funktionsweise auch einfacher Methoden zu beschreiben, so bald mehr als eine

[12] Dies ist im Java-Editor bspw. besser, wenn auch noch relativ abstrakt, über einen Dialog gelöst.

Kontrollstruktur benutzt werden muss. Obschon zuvor in Greenfoot solche oder auch schon größere Komplexität scheinbar kein Problem war, treten diesbezüglich Schwierigkeiten auf. Es scheint zu einem Rückschritt in der Entwicklung von Informatik-Kompetenzen zu kommen.[13]

Denn die Repräsentation der Objekte in BlueJ eine andere, da wesentlich abstrakter. Gut daran ist, dass sie einzeln für sich untersucht bzw. getestet werden können. Nachteilig ist, dass man sie nicht einfach in eine Welt (entsprechend dem Hauptprogramm) integrieren kann und dass verschiedene Objekte im Unterschied zu Greenfoot nicht mehr einfach miteinander interagieren können. Das Zusammenspiel ist lediglich auf der Ebene der Klassen erkennbar, ohne dass hier aber auch nur annähernd UML-Klassendiagramme erkennbar wären. Alles in allem ist die Benutzungsschnittstelle von BlueJ verglichen mit Greenfoot sehr abstrakt.

Die notwendige Abstraktionsleistung wird zudem erschwert durch den Versuch, an dieser Stelle bereits das MVC-Konzept/Pattern umzusetzen. So wichtig dieses Konzept für eine adäquate Software-Entwicklung ist, so wenig intuitiv ist das für die Schüler. Seine Notwendigkeit ist darüber hinaus auf der Grundlage der geringen Erfahrung der Schüler erst nach Fertigstellung einer ersten Version der Software für die Schüler erfahrbar zu machen. Denn im und durch das Objektspiel werden die Schüler lediglich aufgefordert die am Spiel beteiligten Objekte zu benennen (bzw. dann durch Attribuierung zu modellieren). Es besteht kein Anlass zwischen Daten- und GUI-Objekten zu unterscheiden. Auf diese Schwierigkeit, hat Moll hat schon 2002 verwiesen [Mo02]. Hier zwischen Daten (Model) und graphischer Repräsentation (View) zu unterscheiden, muss den Schülern künstlich vorkommen., denn die handelnden und damit die Daten haltenden Objekte sind zugleich auch sichtbar.

Ein weiteres Probleme entsteht, da das Objektspiel nicht alle Methoden und Eigenschaften, die für eine Implementierung relevant sind, erfassen hilft. Man bleibt damit auf der Ebene eines Entwurfsdiagramm und erreicht noch nicht die Ebene der Implementationsdiagramme. Durch die Implementierung kommen ggf. weitere Methoden hinzu, die nicht antizipierbar sind, wenn man nicht schon einmal Software hergestellt hat, was in der Regel bei den Schüler in dieser Phase der Fall ist.

Insgesamt kommen hier weitere Schwierigkeiten zum Tragen, in denen sich eine andere, schlechtere Repräsentation der Objekte und Klassen mit Problemen des Transferlernens und der Abstraktionsfähigkeit überlagern, ohne dass man genau benennen kann, woran es liegt. Zumindest für den Übergang wäre es wünschenswert, wenn man deutlicher die Medienfunktionen des Erzeugens, des Arrangieren und Löschens unterstützen könnte, obschon ebenso deutlich ist, wie schwierig dieses softwaretechnisch zu realisieren ist.

[13] Dies müsste über entsprechende Kompetenztests genauer untersucht werden.

5 Fazit und Ausblick

Es ist auffällig, wie wenig die Prozesse der Modellierung unterrichtlich durch die vorhandenen Umgebungen – und BlueJ ist hier nur ein Beispiel – unterstützt werden. Vor allem der notwendige Übergang zur Implementierung könnte, wie aufgezeigt wurde, besser unterstützt (z.B. automatisiert) werden. So rücken allein schon von zeitlichen Umfang zusätzliche Lerninhalte (fast schon als heimlicher Lehrplan) zur Entwicklung von Kompetenzen im Umgang mit den Werkzeugen und den Besonderheiten der Programmiersprache in Zentrum des Unterrichts. Es stellt sich die Frage, ob das gewollt ist.

Nur zu modellieren, ist kein Ausweg, da dies nicht auf Dauer trägt. Der Charakter informatische Modelle ist ein anderer. Es sind Modelle (als Mittel) zum Zweck einer Implementierung. Dies unterscheidet sie von Modellen der Mathematik, die auch an und für sich existieren können. Ein Modell der Informatik kann eigentlich erst durch eine Implementierung bewertet werden. In Modelle der Informatik fließt zudem (mit größerer als in der Schule zu erwerbender Erfahrung) Wissen um die Implementierung mit ein.

Eine bessere technische Unterstützung der Modellierungsprozesse und der nachfolgenden Implementierungen scheint wesentlich zu sein, der Informatik den Ruf zu nehmen, sie sei nur ein Fach für Freaks, gar Nerds oder etwas neutraler die *besonders Interessierten*. Die Heterogenität wäre möglicherweise weniger groß.

In der Tat ist das bisher nur eine begründete Vermutung und kein empirischer Befund. Einen solchen empirischen Befund zu erhalten ist eine Untersuchung erforderlich, die die Beobachtung von Handlungsprozessen im Unterricht nötig macht. Mithilfe der Medienfunktionen lassen sich diese Beobachtungen systematischer und zugleich differenzierte aufarbeiten, um daraus Anforderungen an Lernumgebungen für den Informatikunterricht zu gewinnen. Solche Anforderungen sollten dann nicht nur umgesetzt sondern auch wieder in ihrer Umsetzung evaluiert werden. Es ergäbe sich ein Zyklus aus Unterrichtsbeobachtung, Anforderungsdefinition und -umsetzung, der dann wieder bei Unterrichtsbeobachtung startet. Dieses Zyklus müsste mehrfach durchlaufen werden bis man entweder Konvergenz feststellt oder auch feststellt, dass all dies in Bezug auf die Heterogenität der Lerngruppen ohne Wirkung bleibt.

Parallel müsste man nämlich die Kompetenzentwicklung der Schüler verfolgen, ob die Schweirigkeiten tatsächlich Folge der Lernhindernisse sind, die durch mangelhafte technische Unterstützung (mit-)verursacht wird oder ob es doch die Probleme sind, die sich aus den notwendigen Abstraktions- oder Transferleistungen bzw. der Fähigkeit zur Algorithmisierung herrühren. Aber das ist von einem allein und auch einer Fachgruppe allein kaum zu leisten. Dieser Bericht bittet schlicht um Unterstützung bei diesen Arbeiten.

Literaturverzeichnis

[BK03] Barnes, D.J., Kölling, M.: Objektorientierte Programmierung mit Java : eine praxisnahe Einführung mit BlueJ. Objects first with Java dt. Pearson. München. 2003

[BW06] Bell, T., Witten, I. H., Fellows, M. 2006: Computer Science Unplugged: enrichment and extension programme for primary-aged children. http://csunplugged.org/sites/default/files/activity_pdfs_full/CS_Unplugged-en-10.2006.pdf

[DM09] Dohmen, M., Magenheim, J., Engbring, D.: Kreativer Einstieg in die Programmierung - Alice im Informatik-Anfangsunterricht. In: Peters, I. (Hrsg.): Informatische Bildung in Theorie und Praxis, Beiträge zur INFOS 2009, 13. GI-Fachtagung - Informatik und Schule, S.69-80, Berlin (LOG IN Verlag) 2009

[En11] Engbring, D.: Untersuchungen und Bewertungen zum Einsatz von Alice im Informatikunterricht. In: M. Weigend, M. Thomas, F. Otte (Hrsg.): Informatik mit Kopf, Herz und Hand. Praxisbeiträge zur INFOS 2011. ZfL-Verlag. Münster, S. 81 - 90

[HK04] Henriksen, P., and Kölling, M. 2004. greenfoot: combining object visualisation with interaction. Proceeding OOPSLA '04 Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications. Pages 73-82

[Ke90] Keil-Salwik, R.: Gestaltung interaktiver Systeme. Habilitationsschrift TU Berlin.

[KT10] Kempe, T., and Tepaße, D. 2010. Informatik 1: Softwareentwicklung mit Greenfoot und BlueJ. Bildungshaus Schulverlage Westermann Schroedel Diesterweg Schöningh Winklers GmbH Braunschweig Paderborn Darmstadt

[Mo02] Moll, S.: Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Informatikunterricht der Jahrgangsstufe 11. In: Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur »Didaktik der Informatik«. Theorie, Praxis, Evaluation 1. Workshop der GI-Fachgruppe DDI 10.-11. Oktober 2002 in Witten-Bommerholz. GI-Edition - Lecture Notes in Informatics (LNI), P-22. S 43 – 52

[Pa81] Pattis, R.: Karol the robot. A gentle Introduction to the Art of Programming with Pascal. John Wiley & Sons. New York. 1981

[Sc05] Schriek, B. 2005. Informatik mit Java: Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse. Band 1. Nili-Verlag. Werl

[Sc11] Schell, A: Gestaltung eines Fragebogens zur Anforderungsanalyse einer Arbeitsumgebung für das Modellieren und Implementieren im Informatikunterricht auf der Grundlage unterrichtspraktischer Beobachtungen. Hausarbeit im Rahmen der 1. Staatsprüfung

[Se08] Selke, H.: Sekundäre Medienfunktionen für die Konzeption von Lernplattformen für die Präsenzlehre. Universität Paderborn, Heinz Nixdorf Institut, Informatik und Gesellschaft, 2008. http://digitool.hbznrw.de:1801/view/action/singleViewer.do?dvs=1355127105464~548&locale=de_DE&preferred_extension=pdf&preferred_usage_type=VIEW_MAIN&DELIVERY_RULE_ID=10100&frameId=1&usePid1=true&usePid2=true

[UC10] Utting, I., Cooper, S., Kölling, M., Maloney, J., and Resnick, M. 2010: Alice, Greenfoot, and Scratch: A Discussion. Trans. Comput. Educ. 10, 4, Article 17 (November 2010), 11 pages.

Einsatz von Speicherprogrammierbaren Steuerungen im Lernlabor Abenteuer Technik

Dr.-Ing. Jens Gallenbacher, Dominik Heun, Kristin Rammelt

Technische Universität Darmstadt
Didaktik der Informatik
Hochschulstraße 10
64289 Darmstadt
jg@di.tu-darmstadt.de
dh@di.tu-darmstadt.de
kr@di.tu-darmstadt.de

Abstract: Im Lernlabor Abenteuer Technik der Technischen Universität Darmstadt wurde ein neuer Workshop entwickelt. Mit Speicherprogrammierbaren Steuerungen wird eine Modellampel einer Kreuzung programmiert und durch spielerisches Durchlaufen verschiedener Situationen erprobt. Dieser Praxisbericht stellt das Konzept der Speicherprogrammierbaren Steuerungen dar und begründet den Mehrwert für den Einsatz in Schülerworkshops. Danach wird das Ampelmodell und das zugehörige Workshopkonzept vorgestellt. Das Konzept wurde bisher viermal erprobt. Die Erkenntnisse aus den Erprobungen werden zum Abschluss dargestellt.

1 Speicherprogrammierbare Steuerungen

Speicherprogrammierbare Steuerungen (SPS) sind nach der europäischen Norm EN 61131 digital arbeitende elektronische Systeme für den Einsatz in industrieller Umgebung. Sie besitzen einen programmierbaren Speicher, der Steuerungsanweisungen enthält, um durch digitale oder analoge Eingangs- und Ausgangssignale verschiedene Arten von Maschinen zu steuern.[1]

Eine SPS kann in vielfältigen Bereichen ihre Anwendung finden. So können industrielle Fertigungsstraßen von einer SPS überwacht und gesteuert oder Kleinanlagen für die Automatisierung im häuslichen Bereich genutzt werden. Werden die Ausgangs-leitungen einer SPS zur Schaltung von Relais genutzt, können jegliche Geräte gesteuert werden.

[1] vgl. [vA 05]

Die Arbeitsweise der Steuerung ist zyklenbasiert und besteht aus den folgenden Arbeitsschritten:

- Einlesen der Eingänge
- Anwenderprogramm abarbeiten
- Ausgänge entsprechend der Berechnung schalten

In den Arbeitsschritten wird das EVA-Prinzip deutlich. Die zeitliche Länge eines Zyklus ist variabel und von der Anzahl der nötigen Berechnungsschritte abhängig. Werden bspw. 1000 Rechenanweisungen ausgeführt, dauert der Zyklus einer Kleinsteuerung ca. 1,5 ms. Industrielle Steuerungen weisen eine schnellere Abarbeitung auf. Durch Interrupts kann ein Zyklus unterbrochen werden, um zeitkritische Anwendungen auszuführen.

Mögliche Sprachen zur Programmierung von SPS werden im dritten Teil der Norm zusammengefasst. Sie definiert fünf Sprachen:

- Anweisungsliste (AWL)
- Kontaktplan (KOP)
- Funktionsbausteinsprache (FBS)
- Ablaufsprache (AS)
- Strukturierter Text (ST)

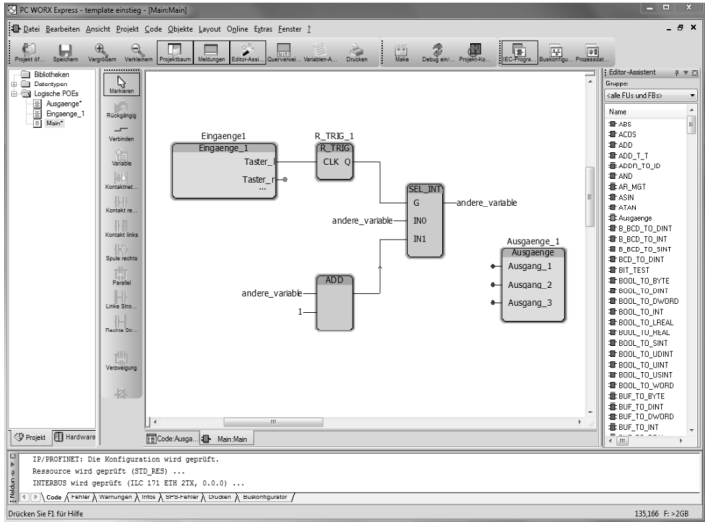


Abbildung 1: Funktionsbausteinsprache in PC WORX-Express

Im weiteren Verlauf wird die Funktionsbausteinsprache zur Programmierung verwendet. Sie zeichnet sich durch grafisch orientierte Programmierung aus. Aus den Eingangssignalen wird durch logische Schaltungen, Speicherung und Berechnung das Ausgangssignal bestimmt. Die Berechnungen und Schaltungen werden während der Programmierung durch logikgatterähnliche Blöcke dargestellt. Die Ein- und Ausgänge der Blöcke können mit virtuellen Kabeln verbunden werden, um die gewünschte Abarbeitung zu erreichen (vgl. Abbildung 1). Des Weiteren wird Modularisierung unterstützt. Komplexe Berechnungen werden als neuer Block abstrahiert und können im Programm als Black-Box genutzt werden.

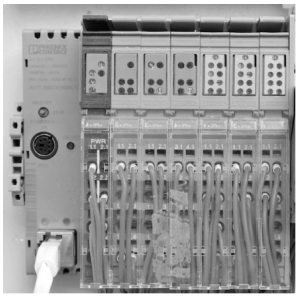


Abbildung 2: Modularisierte Speicherprogrammierbare Steuerung

In diesem Praxisbericht werden Steuerungen von Phoenix Contact verwendet, die für den Kleinanlagenbereich genutzt werden. Sie zeichnen sich durch einen modularen Aufbau aus. In Abbildung 2 ist beispielhaft eine Steuerung dargestellt. Die orangefarbenen Module stellen die Basiseinheiten der SPS dar, die fest verbaut sind. Durch Zusatzmodule wurden weitere Eingänge (blau) und Ausgänge (rosa) angesteckt und können für die Programmierung genutzt werden.

Die Steuerung hat eine eingebaute Echtzeituhr, die zeitlich abhängige Aktionen anstoßen kann. Die SPS kommuniziert über ein Netzwerk-Kabel mit dem Rechner.

Zur Programmierung wird die Software PC WORX-Express genutzt, die kostenlos über die Internetseite von Phoenix Contact[2] bezogen werden kann. Sie ist ebenfalls in Abbildung 1 zu sehen und unterstützt alle genannten Programmiersprachen für eine SPS. Des Weiteren bietet PC WORX-Express einen Debug-Modus, bei dem durch farbliche Hervorhebung die aktuellen Berechnungen und geschalteten Eingänge visualisiert werden. Kabel, die *virtuell* Strom führen, werden rot dargestellt. Dieser Debug-Modus erlaubt einen LiveView auf die Ausführung des Programms.

Zur Ausführung des Programms ist ein Kompilierungsvorgang nötig, der mit PC WORX-Express durchgeführt werden kann. Über die Verbindung mittels Netzwerkkabel kann das Kompilat ebenfalls über die Software auf die Steuerung geladen werden. Die Software bietet weiterhin Steuerungsmöglichkeiten, um die Ausführung des Programms starten und beenden zu können.

[2] https://www.phoenixcontact.com

2 Gründe für den Einsatz in Schülerworkshops

Eine SPS bietet einen guten Ansatzpunkt um Schülerinnen und Schüler der beruflichen Schulen zu fördern. Da die Möglichkeiten der Ansteuerung von Geräten durch die Verwendung von Relais sehr vielfältig sind, können motivierende Beispiele erstellt werden.

Im Rahmen einer Fachdidaktik-Veranstaltung an der Technischen Universität Darmstadt wurde ein Aufzug-Modell aus Fischertechnik mit einer SPS programmiert. Aus den gewonnen Erkenntnissen sollte ein motivierender Workshop für Lernende der Haupt- und Realschulen erstellt werden, damit diese mit Anlagensteuerungen vertraut gemacht werden und somit einen Einblick in die Berufe der Mechatroniker und Elektroniker erhalten.

Die Verwendung von echten Steuerungen hat gegenüber Nachbildungen wie bspw. Crossroads von der knobloch-gmbh[3] den Vorteil, dass die Schülerinnen und Schüler mit der real verwendeten Programmiersprache vertraut werden und mögliche Modellvorstellungen, die sie über diese Sprache bilden, in einer evtl. Ausbildung tatsächlich anwenden können. Trotzdem bieten die unterschiedlichen Programmiersprachen zur Programmierung von SPS verschiedene Ansätze zum Programmierung. Für den Einstieg in die Programmierung eignet sich die Funktionsbausteinsprache am besten, da sie eine intuitive Möglichkeit bietet. Da die Programmierung auf textuelle Aspekte verzichtet, treten keine Syntaxfehler auf sobald die Blöcke miteinander verbunden werden.

Die Programmierung wird durch Drag-and-Drop der Blöcke in die Arbeitsfläche und Verbinden der Kabel realisiert. Dafür benötigen die Schülerinnen und Schüler nur wenige Computerkenntnisse und können vergleichbar zu einstiegsfreundlichen Sprachen wie Scratch oder Squeak Etoys arbeiten. Dies ist ein weiterer positiver Aspekt, der für die Verwendung von SPS in Schülerworkshops spricht. Obwohl es sich um eine Sprache zur Steuerung von realen industriellen Anlagen handelt, weist sie eine hohe Einsteigerfreundlichkeit auf.

Die Schülerinnen und Schüler können ohne weitere Computerkenntnisse eine Steuerung programmieren. Lediglich eine Vorstellung von Variablen sollten die Lernenden für den Workshop mitbringen. Die Vermittlung der nötigen Variablenkenntnisse kann aber auch im Workshop selbst erfolgen.

Insgesamt bieten SPS große Vorteile für den Einsatz in Workshops, wie in diesem Abschnitt dargelegt wurde. Mittels motivierender Anlagen lernen die Schülerinnen und Schüler anhand authentischer Programmierumgebungen wie sie in ihrer Ausbildung Anlagen ansteuern können. Des Weiteren schult die Verwendung von SPS allgemeinbildende Kompetenzen der GI-Bildungsstandards[4] wie Modellieren und Implementieren.

[3] http://www.knobloch-gmbh.de/

[4] [P+08]

3 Ampelmodell

Das verwendete Modell trägt erheblich zum Motivationsaspekt der SPS bei. Des Weiteren sollen die Schülerinnen und Schüler konkrete Vorstellungen aus ihrem Alltag mitbringen, die ihnen bei der Programmierung des Modells behilflich sind. Für einen Workshop im Lernlabor Abenteuer Technik wurde eine Kreuzung entworfen, die durch die SPS angesteuert wird.

Eine einzelne Ampel bzw. mehrere Ampeln an einer Kreuzung kennen die Lernenden aus ihrem Alltag. Sie können direkt Annahmen zur Programmierung treffen und kennen Regeln, die bei einer Ampelprogrammierung beachtet werden müssen.

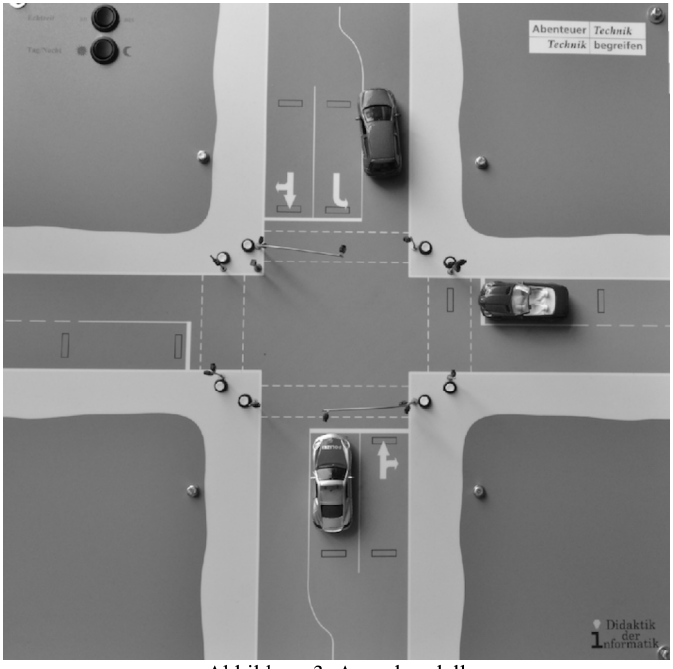


Abbildung 3: Ampelmodell

In Abbildung 3 ist das erstellte Ampelmodell zu sehen. Der generelle Aufbau der Kreuzung wird deutlich. Eine Hauptstraße hat separate Abbiegespuren während die Nebenstraße nur eine Ampel für alle Richtungen besitzt.

Es gibt in der linken oberen Ecke zwei Schalter, die zur zeitlichen Steuerung genutzt werden können. Ein Schalter setzt die Echtzeituhr der Steuerung außer Kraft und lässt den Tag- bzw. Nachtmodus über den zweiten Schalter einschalten. An jeder Ampel befindet sich ein Fußgängertaster, der bedarfsgerecht die Fußgängerampel auf grün schaltet.

Auf der Fahrbahn sind rechteckige Markierungen zu erkennen, die Induktionsschleifen zur bedarfsgesteuerten Ampelsteuerung darstellen. Im Modell wurden diese Induktionsschleifen durch Reed-Kontakte realisiert, die die mit Magneten ausgestatteten Modellautos erkennen.

Neben den visuellen Fußgängerampeln sind im Modell Summer verbaut, die Blindenampeln simulieren.

Die Ampeln wurden durch Modellbauampeln realisiert, bei denen jede einzelne LED angesteuert werden kann. Die Ampel an der Hauptstraße ist eine Bogenampel, die zwei separate Ampeln trägt. Damit kann die Abbiegespur unabhängig von der Geradeausspur geschaltet werden.

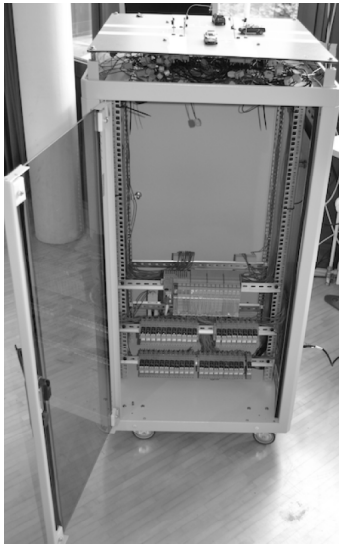


Abbildung 4: Mobiler Schrank mit SPS und Ampelmodell

Das Modell befindet sich in einem mit Rollen ausgestatteten Serverschrank, der alles Notwendige für die Programmierung mit der SPS enthält (vgl. Abbildung 4). Auf dem Schrank befindet sich das dargestellte Ampelmodell (vgl. Abbildung 3), mit dem die Lernenden interagieren können. Im Schrank ist die SPS mitsamt der nötigen Relais verbaut, um die Ausgänge im Modell schalten zu können. Über der SPS kann ein Rechner mitsamt Monitor verstaut werden. Es ist nur eine einzige Steckdose nötig, um alle Bestandteile in Betrieb zu nehmen. Ein Hauptschalter im Schrank erlaubt das komfortable Ein- und Ausschalten aller Komponenten.

4 Workshopkonzept

Die Lernenden für den betreffenden Workshop sind in der 7. Klasse und haben keine Vorerfahrungen mit Programmierung. Die Klasse ist Teil einer Haupt- und Realschule. Sie bringen die üblichen heterogen verteilten Computerkenntnisse mit, die in Lerngruppen dieser Altersstufe üblich sind.

Das Modell weist mit der Vielzahl von Ein- und Ausgängen eine sehr hohe Komplexität auf. Um die Programmierung für die Lernenden zu ermöglichen, wurde eine Vorlage entwickelt, die die Komplexität verringert. So schaltet die Vorlage die beiden sonst einzeln ansteuerbaren Fußgängerampeln eines Überwegs zusammen. Ferner werden Abbiegeampel und Hauptampel gleichgeschaltet sowie die Reed-Kontakte der Straßen versteckt. Die Vorlage erlaubt jedoch jederzeit eine Erhöhung der Komplexität des Modells und somit eine individuelle Anpassung des Schwierigkeitsgrades auf die Lerngruppe.

Das Vorgehen des Workshops gliedert sich in drei Bereiche:

- Modellierung einer Ampelschaltung
- Einführung in die Programmierung
- Implementierung der modellierten Ampelschaltung

Diese werden jeweils in 90-minütigen Einheiten bearbeitet.

Zunächst werden die Teilnehmenden aufgefordert eine Ampelschaltung zu entwerfen, wobei ihnen die Darstellungsform freigestellt wird. Da die Schülerinnen und Schüler aus ihrem Alltag Ampelschaltungen kennen, können sie einige Regeln formulieren, die für ihre spätere Ampelschaltung wichtig sind. Es wird erwartet, dass die Darstellungsformen sehr unterschiedlich sein werden, was in einem Museumsrundgang mit den Lernenden erörtert wird. Als Lösung werden Automaten vorgestellt, die anhand eines Beispiels mit den Schülerinnen und Schüler erarbeitet werden. In einer anschließenden Phase übersetzen die Lernenden ihre Modellierung in Automaten.

Der zweite Abschnitt des Workshops führt die Lernenden in die Programmierung mit PC WORX-Express ein. Da sie keinerlei Vorwissen zu logischen Operatoren und Programmierung haben, wird ihnen am Beispiel eines AND-Gatters vorgeführt wie die Programmierung in PC WORX-Express funktioniert. Danach erhalten sie einen Forschungsauftrag, um die Funktionsweise von anderen Gattern herauszufinden. Durch Programmierung und Analyse des Verhaltens werden die Lernenden nach und nach in weitere Konzepte der Programmierung eingeführt. Variablen werden am Beispiel einer Schatzkiste veranschaulicht. Eine Schatzkiste besitzt einen Namen sowie einen Wert als Inhalt. In einem kurzen Rollenspiel erfahren die Lernenden anhand dieses Beispiels wie Zuweisungen funktionieren. Die Schülerinnen und Schüler lernen eine Variable nach Tasterdruck hochzuzählen. Diese Variable wird im letzten Abschnitt den aktuellen Zustand repräsentieren. Insgesamt benötigen die Schülerinnen und Schüler zur erfolgreichen Implementierung der Ampelschaltung nur fünf verschiedene Blöcke.

Im letzten Abschnitt tragen die Schülerinnen und Schüler die Erkenntnisse aus den ersten beiden Einheiten zusammen. Aus dem Automaten der ersten Phase und den Variablen sowie den logischen Operatoren der zweiten Phase setzen sie nun ihre fertige Schaltung zusammen. Sie können ihren aktuellen Fortschritt jederzeit am Ampelmodell und im Debug-Modus überprüfen.

Als Minimalziel sollen die Lernenden eine einzelne Ampel programmieren, die die verschiedenen Ampelphasen durchläuft. Danach können durch die Flexibilität der Vorlage weitere Aufgaben hinzugefügt werden, wie bspw. die Einbindung der Abbiegeampel, die Reed-Kontakte zur Bedarfsschaltung und der Schalter zur Tag- und Nachtschaltung.

Zum Abschluss des Workshops präsentieren die Teilnehmenden in Gruppen die Ergebnisse der einzelnen Phasen und machen die unterschiedlichen Arbeitsschritte (Modellierung, Formalisierung, Implementierung) deutlich.

5 Durchführung und Erfahrung

Der dargestellte Workshop wurde bisher mit vier Schülergruppen von je 12-14 Personen durchgeführt. Das Feedback, aber auch viele Einfälle der Lernenden während des Workshops haben zur Weiterentwicklung des Workshops zu seiner aktuellen Form beigetragen.

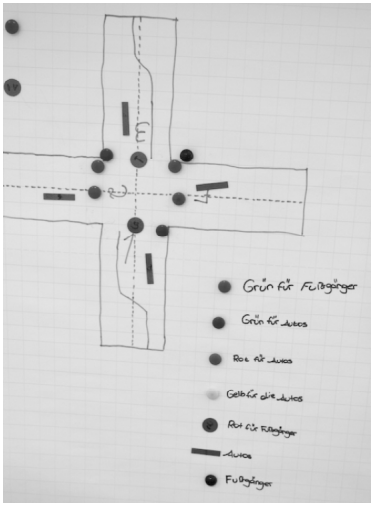


Abbildung 5: Darstellung einer Ampelschaltung mit Magneten

In der ersten Phase konnten die Lernenden sehr schnell erste Annahmen zu ihrer Ampelschaltung treffen. Die Ergebnisse der Gruppen zeigten unterschiedliche Ideen auf, die auf verschiedene Arten visualisiert wurden. Um die verschiedenen Ampelphasen

darstellen zu können, entschied sich eine Gruppe verschiedenfarbige Magnete zu nutzen, um durch Änderung des Magneten die Ampelphase zu ändern (Abbildung 5). Die Übertragung von einzelnen Ampelphasen zu Automaten konnten die Schülergruppen ebenfalls sehr schnell umsetzen und auf ihren Flipcharts geeignet visualisieren.

Die Inhalte der zweiten Phase haben die Lernenden schnell aufgenommen und konnten diese in kleinen Beispielen eigenständig anwenden. Die Arbeit am Rechner mit PC WORX-Express stellte für sie keine schwerwiegenden Probleme dar. Sie konnten die Bedienung schnell eigenständig umsetzen und einfache Schaltungen lösen. Besonders die Forschungsaufträge zur Bestimmung der Funktion bestimmter Blöcke wurden begeistert bearbeitet.

Bei der Implementierung der Ampelphasen gab es in den einzelnen Gruppen große Unterschiede. Einige Schülerinnen und Schüler konnten die gesamte Kreuzung mit der vereinfachenden Vorlage programmieren. Einige Gruppen haben nur das Minimalziel erreicht.

Insgesamt hat sich das Konzept als sehr motivierend für die Schülerinnen und Schüler herausgestellt. Durch ihr Vorwissen zu Ampeln konnten sie schon zu Beginn mitreden und ihre Kenntnisse einbringen. Das mobile Ampelmodell in einem Serverschrank erwies sich ebenfalls als positiv, da der Durchführungsort unkompliziert verlegt werden kann. Es reicht aus, die Schränke zu verschieben und die Stromverbindung herzustellen. Die Lernenden probierten gerne ihre Programme mit dem Ampelmodell aus und spielten verschiedene Szenarien durch. Dabei waren sie motiviert evtl. Probleme ihrer Programmierung zu lösen, um die Ampel zum korrekten Schalten zu bringen.

Die stufenartige Änderung der Komplexität war für die bisherigen Schülerinnen und Schüler nicht notwendig, da sie Programmieranfänger waren und somit die Programmierung mittels der einfachen Vorlage schon einen Erfolg für sie darstellt.

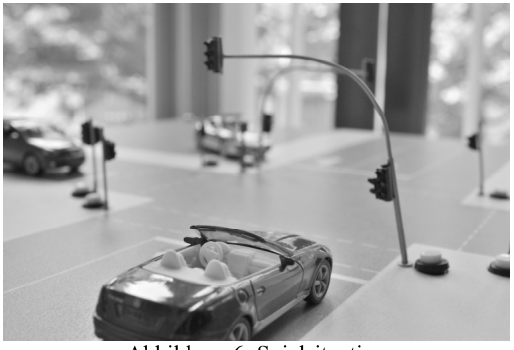


Abbildung 6: Spielsituation

6 Fazit

Speicherprogrammierbare Steuerungen können durch einen motivierenden Kontext schon für Schülerinnen und Schüler der Jahrgangsstufe 7 auf Haupt- und Realschulen eingesetzt werden. Doch auch für höhere Klassenstufen bietet das Modell durch geeignete Vorlagen Herausforderungen. Bis hin zur kompletten Ansteuerung aller Ampeln und Nutzung aller Sensoren können die Vorlagen angepasst werden. Damit ist ebenfalls ein mehrtägiger Workshop denkbar, der die Problemstellung Schritt für Schritt erweitert.

Das in sich geschlossene Modell ist mobil und kann in unterschiedlichen Räumen schnell genutzt werden. Durch sein motivierendes Ampelmodell werden die Schülerinnen und Schüler an Speicherprogrammierbare Steuerungen herangeführt.

Zukünftig wird der Workshop für weitere Klassenstufen geplant und durchgeführt und auch in anderen Schulformen erprobt.

Literaturverzeichnis

[P+08] Hermann Puhlmann et al. Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. Beilage zu LOG IN, (150/151), 2008.

[vA05] J. von Aspern. *SPS-Grundlagen: Aufbau, Programmierung (IEC 61131, S7), Simulation, Internet, Sicherheit*. Hüthig, 2005.

[WZ98] Günter Wellenreuther und Dieter Zastrow. *Steuerungstechnik mit SPS*. Vieweg, 1998.

Aufgabenqualität im Informatik-Biber

Wolfgang Pohl, Hans-Werner Hein

BWINF / Bundeswettbewerb Informatik
Wachsbleiche 7
53111 Bonn
{pohl,hein}@bwinf.de

Abstract: Die Diskussion der Qualität von Aufgaben steht im Fach Informatik im Allgemeinen noch am Anfang. Beim Aufgabenwettbewerb Informatik-Biber wurde diese Diskussion auf internationaler Ebene schon früh geführt. Die Entwicklung von Aufgaben für den Informatik-Biber wird kurz beschrieben und eine sich an der Durchführung des Wettbewerbs orientierende Qualitätsmaxime formuliert. An einem Beispiel wird die Entwicklung einer Biber-Aufgabe bis hin zur Schlussfassung demonstriert und gezeigt, dass die Änderungen der hier aufgestellten Maxime folgen.

1 Einleitung

Die Qualität von Aufgaben wird im Zusammenhang mit der Qualität von schulischem Unterricht intensiv diskutiert. Für das Fach Mathematik etwa gibt es zahlreiche Arbeiten zur Aufgabenqualität. „Qualität“ ist ein relativer Begriff und muss sich auf einen Maßstab beziehen. Das gilt auch für Aufgaben: „Aufgaben an sich sind nicht in einem absoluten Sinn gut; um von guten Aufgaben reden zu können, bedarf es eines Qualitätsmaßstabes“ [Wal].

Für das Fach Informatik ist die Diskussion um Aufgabenqualität weniger ausgeprägt. Der Bildungsserver Berlin-Brandenburg dokumentiert die Ergebnisse eines Workshops zum Thema „Aufgabenkultur Informatik“[1] und gesteht unter der gleichen Überschrift ein: „Im Fach Informatik steht die Diskussion einer Aufgabenkultur erst am Anfang.“[2]

Aufgabenqualität ist für Schülerwettbewerbe des Typs „Aufgabenwettbewerb“ [Poh06a] ein wichtiges Thema, wird aber zumindest in Deutschland nur selten formal diskutiert. Für den ältesten deutschen Aufgabenwettbewerb in Informatik, den Bundeswettbewerb Informatik [Poh06b], gibt es nur interne Papiere, die die Ergebnisse von Diskussionsprozessen insbesondere des Aufgabenausschusses dokumentieren.

Anders sieht es beim Informatik-Biber [PSH09] aus, dem Einstiegsangebot zum Bundeswettbewerb Informatik. Im Rahmen des internationalen Abstimmungsprozesses bei der Entwicklung von Aufgaben (s. Abschnitt 2) wurden Aufgaben und ihre Qualität nicht nur intern früh diskutiert, sondern auch in Publikationen thematisiert [ODT06, DF08].

[1] http://bildungsserver.berlin-brandenburg.de/gute_aufgaben_informatik.html
[2] http://bildungsserver.berlin-brandenburg.de/aufgabenkultur_informatik.html

In dieser Arbeit wollen wir für den Informatik-Biber exemplarisch Aspekte von Aufgabenqualität aufzeigen, die in diesem Schülerwettbewerb eine Rolle spielen. Der folgende Abschnitt beschreibt kurz den Prozess der Aufgabenentwicklung im Informatik-Biber und stellt erste, allgemeine Überlegungen zur Aufgabenqualität an, die in eine Qualitätsmaxime münden. In Abschnitt 3 werden die Entwicklung einer Aufgabe von einer frühen bis zur endgültigen Fassung demonstriert, die Veränderungen erläutert und mit der Qualitätsmaxime in Beziehung gesetzt. Der letzte Abschnitt rundet diese Arbeit mit einem Fazit ab.

2 Aufgabenentwicklung im Informatik-Biber

Der Informatik-Biber ist deutscher Partner der internationalen Bebras-Initiative [Dag08, Beb]. Alle Bebras-Partner in mittlerweile über 21 Ländern weltweit nutzen einen Pool von Aufgaben, der in der ersten Jahreshälfte beim so genannten Internationalen Aufgaben-Workshop erstellt wird. Dabei steuert jedes Land bzw. jeder Partner eine Reihe von Aufgabenvorschlägen in englischer Sprache bei; zum Workshop 2013 z. B. sollten jeweils sieben bis zehn Aufgaben eingereicht werden. Alle eingereichten Aufgabenvorschläge werden in Arbeitsgruppen begutachtet, teilweise bearbeitet und letztlich entweder in den Aufgabenpool des aktuellen Biber-Jahres aufgenommen oder abgelehnt. Einige Aufgaben werden zu Pflichtaufgaben („mandatory tasks“) erklärt, die in allen Bebras-Ländern gleichermaßen verwendet werden müssen.

Nach dem Workshop wählen die Partner unabhängig voneinander die über die Pflichtaufgaben hinaus benötigten Aufgaben aus dem Pool und sorgen dann für die Übertragung in die Landessprache. Dabei können Modifikationen gegenüber dem Original nötig werden: Zum einen mag die im Pool vorliegende Fassung noch Schwächen haben, zum anderen können durch die Übertragung in die Landessprache sprachliche (semantisch, idiomatisch, metaphorisch) oder auch kulturelle Anpassungen (Beachtung landesspezifischer „political correctness“, abweichenden „common sense“ und eigenkultureller Tabus) erforderlich werden.

Auch beim Informatik-Biber muss es für eine qualitätsorientierte Entwicklung und Bearbeitung von Aufgaben einen Qualitätsmaßstab geben. Dagiene und Futschek [DF08] gehen auf Qualitätskriterien für Bebras-Aufgaben ein, die aber, wenn sie sich auf von der Wettbewerbsdurchführung unabhängige Fragen der Aufgabenpräsentation beziehen, oberflächlich bleiben und kaum messbar sind. Nicht alle diese Kriterien sind derzeit Konsens in der Bebras-Gemeinschaft, etwa die Lösbarkeit einer Aufgabe ohne Hilfsmittel oder die Darstellbarkeit einer Aufgabe auf einer einzigen Bildschirmseite. Allein durch die Randbedingungen eines Bebras-Wettbewerbs, in dem ein Teilnehmer innerhalb kurzer Zeit recht viele Aufgaben bearbeiten soll (beim Informatik-Biber 18 Aufgaben in 40 Minuten), ist aber erstens das folgende Kriterium bedeutsam: Eine Aufgabe muss in möglichst kurzer Zeit verstanden werden können. Ein zweites Kriterium ist für die faire Durchführung eines Wettbewerbs unerlässlich: Eine Aufgabe muss korrekt verstanden werden können. Dabei ist das Verständnis der Aufgabe dann korrekt, wenn es dem Verständnis der Aufgabensteller entspricht. Drittens gilt im Informatik-Biber grundsätzlich: Die Aufgabe muss

ohne fachliche Vorkenntnisse verständlich sein. Zusammengefasst lässt sich als technische Maxime für die Autoren einer Biber-Aufgabe formulieren: *Die Aufgabe muss in möglichst kurzer Zeit und ohne fachliche Vorkenntnisse korrekt verstanden werden können.*

3 Qualitätsentwicklung im Informatik-Biber am Beispiel

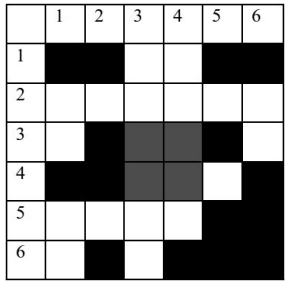
Als Beispiel wählen wir die Aufgabe „Verlorene _nf_rmat_on“, die in 2011 und 2012 im Informatik-Biber verwendet wurde [Bib13, S. 40]. Sie besteht, wie viele Aufgaben im Informatik-Biber, aus einem einleitenden Text, einer zentralen Abbildung, einer Frage (die hier zunächst noch durch einen Satz eingeleitet wird) und den Antwortalternativen, die in diesem Fall wiederum durch Abbildungen dargestellt werden. Alle Textbestandteile einer Aufgabe (einleitender Text und Frage) bezeichnen wir im Folgenden als Aufgabentext.

Darüber hinaus gehören zu einer Aufgabe im Informatik-Biber zwei weitere Bestandteile, die aber erst nach dem Wettbewerb veröffentlich werden: (1) ein *Lösungstext*, der die Lösung der Aufgabe behandelt und sowohl auf die korrekte Lösung als auch auf falsche Lösungsalternativen eingeht, und (2) ein *Hintergrundtext*, der unter der Überschrift „Das ist Informatik!“ den fachlichen Hintergrund der Aufgabe erläutert.

Missing Piece

Beaver John has received a secret message. Unfortunately a part of the message has been destroyed by a spill of red colour.

This case was foreseen and there are additional squares in the message. Each square in the rightmost column (column 6) or the lowest row (row 6) is coloured such that the number of black squares in a row, respectively in a column is even.



John considers there are sixteen different possible messages. Only four of them make sense to him.**What is the pattern of the red piece?**

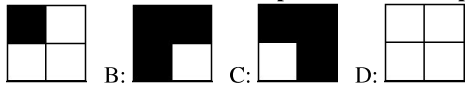


Abbildung 1: Stand der Aufgabenstellung nach dem Internationalen Aufgaben-Workshop

3.1 Fassung nach dem Internationalen Aufgaben-Workshop

Abbildung 1 zeigt den Stand der Aufgabenstellung nach dem Internationalen Aufgaben-Workshop. Fachlich geht es um selbstkorrigierende Codes: ein in Form einer Matrix aus 5×5 schwarzen und weißen Feldern gegebener Code („secret message“) wird um je eine weitere, fehlerkorrigierende Spalte und Zeile ergänzt.

3.2 Erste Übertragung ins Deutsche

Verlorene Information
Die Informatik-Biber kennzeichnen gefällte Bäume mit einer Art Strichcode in einem 6×6 Quadrat mit der Besonderheit, dass die Anzahl schwarzer Felder in jeder Zeile und jeder Spalte gerade sein muss.
Beim folgenden Strichcode wurden die vier roten Felder beschädigt.
Bild wie in Abbildung 1
Wie sahen die vier roten Felder aus?
Antwortalternativen wie in Abbildung 1

Abbildung 2: Erste Übertragung der Aufgabenstellung ins Deutsche

Abbildung 2 zeigt die erste Übertragung der Aufgabenstellung ins Deutsche. Während die Abbildungen unverändert geblieben sind, unterscheidet sich der Text deutlich vom englischen Original, und zwar in folgenden Punkten:

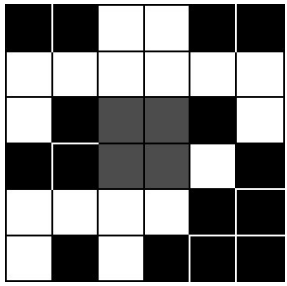
1. Die Geschichte der Aufgabe wurde verändert, aus der „Nachricht“ wurde eine „Kennzeichnung“ von Bäumen. Dies passt besser zur zweidimensionalen Darstellung des Codes.
2. Die recht umständliche Beschreibung des Paritätseffektes der zusätzlichen Zeile und Spalte wurde vermieden, indem der Code gleich als 6×6 Matrix mit Paritätseigenschaft in jeder Zeile und Spalte beschrieben wird.
3. Der in der englischen Fassung unternommene Versuch, die vier Antwortalternativen zu motivieren („. . . sixteen different possible messages. Only four of them make sense . . .“) ist unnötig und wird weggelassen.
4. In der englischen Fassung wird der fehlerhafte Teil des Codes uneinheitlich bezeichnet, nämlich einmal als „part of the message“ und einmal als „piece“. In der deutschen Übertragung ist einheitlich von „die vier roten Felder“ die Rede. Interessant ist, dass in der Frage die Bezeichnung „die vier roten Felder“ vollständig wiederholt wird, anstatt sie vermeintlich eleganter mit „die vier Felder“ abzukürzen.

Alle Änderungen sind im Sinne unserer Maxime: Die erste und die letzte Änderung erleichtern das korrekte Verständnis. Die Kürzungen (Punkte 2 und 3) dienen einer schnel-

leren Auffassung der Aufgabe. Spezifisch informatische Vorkenntnisse sind weder in der englischen Fassung noch in der deutschen Übertragung zum Verständnis der Aufgabe erforderlich. Allerdings kann nicht davon ausgegangen werden, dass der Begriff „Strichcode“ jeder/m potenziellen Teilnehmenden bekannt ist.

3.3 Redaktionelle Änderungen

Verlorene _nf_rmat_on
Die Informatik-Biber kennzeichnen ihre gefällten Bäume. Ein Kennzeichen besteht aus einer Matrix von 6 mal 6 Feldern, die schwarz oder weiß sein können. Bei jedem Kennzeichen ist in jeder Reihe und in jeder Spalte die Anzahl der schwarzen Felder immer gerade. So ist das Kennzeichen in der rauen Umgebung etwas robuster.
Dieses Kennzeichen wurde beim Baumtransport verschmutzt:



Wie sahen die vier roten Felder vorher aus?
Antwortalternativen wie in Abbildung 1

Abbildung 3: Schlussfassung

Abbildung 3 zeigt die Aufgabenstellung nach redaktioneller Bearbeitung. Zunächst stellen wir fest, dass sich die zentrale Abbildung verändert hat. Zum einen sind die obere Zeile und die linke Spalte entfallen. Sie enthielten Zeilen- und Spaltennummern, die zwar noch in der englischen Fassung, aber in der deutschen Übertragung schon nicht mehr benötigt wurden. Außerdem waren die Felder mit den Nummern nicht ausreichend von den Feldern des Codes zu unterscheiden. Zum anderen sind zwischen schwarzen Feldern nun weiße Trennlinien zu sehen, entsprechend den schwarzen Trennlinien zwischen weißen Feldern. Ursprünglich hatten nebeneinander liegende schwarze Felder einen durchgängigen schwarzen Block gebildet, wodurch die Struktur des Codes verschleiert wurde. Leider wurde versäumt, dieses Prinzip auf die Bilder mit den Antwortalternativen zu übertragen.

Darüber hinaus gibt es einige Änderungen im Aufgabentext:

1. Die vergleichende Beschreibung der Kennzeichen als „eine Art Strichcode“ wurde aufgegeben. Dieser Vergleich war dazu gedacht, den Matrixcode der Kennzeichnen mit etwas Bekanntem in Beziehung zu setzen. Das kann sinnvoll sein, aber auch schädlich: Ist ein solcher Vergleich nämlich unpassend, kann er zu Assoziationen führen, die für das Verständnis der Aufgabenstellung nicht nötig sind und die korrekte Beantwortung möglicherweise störend beeinflussen können. Im Fall unserer Aufgabe passt der Vergleich nicht gut, da ein Strichcode (auch: Barcode) andere Eigenschaften als der gezeigte Matrixcode hat.

2. An die Stelle des Strichcode-Vergleichs tritt das Substantiv „Kennzeichen“, passend zum unmittelbar vorher benutzten Verb „kennzeichnen“. Damit erhält der Code einen eigenen Bezeichner.

3. Der lange erste Satz der ersten deutschen Fassung wird in insgesamt drei kürzere Sätze aufgeteilt. Die semantische Klammer wird durch die wiederholte Verwendung des Worts „Kennzeichen“ bzw. des passenden Verbs „kennzeichnen“ gesetzt.

4. Der unklare Begriff „6x6 Quadrat“ wird durch „Matrix mit 6 mal 6 Feldern, die schwarz oder weiß sein können“ ersetzt. Damit wird die formale Beschaffenheit des Codes klar definiert. Indirekt wird die Rotfärbung einiger Felder in der Abbildung als Fehlersituation vorbereitet.

5. Als explizite Bezeichnung der Fehlersituation genügt nun das Verb „verschmutzt“. Die missverständliche Formulierung „wurden die vier roten Felder beschädigt“ (unklar: Ist die rote Färbung die Beschädigung, oder sind die Felder rot und wurden dann beschädigt?) kann entfallen.

Im Sinne der Maxime soll mit diesen Änderungen das schnelle und korrekte Verständnis erleichtert werden. Aus den konkreten Maßnahmen lässt sich ableiten, was auch allgemein für die Formulierung einer Aufgabe wichtig ist (mit Motivation):

- kurze Sätze (sind leichter lesbar, erleichtern das schnelle Verstehen);
- Wortwiederholungen (ermöglichen die Aufteilung langer Sätze und festigen das Verständnis);
- klare Definitionen (erleichtern das korrekte Verstehen);
- passende Assoziationen (unpassende Assoziationen behindern das schnelle, aber auch das korrekte Verstehen);
- unmissverständliche Formulierungen (missverständliche Formulierungen behindern das korrekte Verständnis).

Zwei Besonderheiten gibt es zur Schlussfassung noch anzumerken: Zum einen wurden im Titel einige Buchstaben durch Unterstriche ersetzt, was zum Inhalt der Aufgabe passt. Wir sind der Auffassung, dass der Titel einer Aufgabe für das Verständnis nicht bedeutsam ist, und halten eine solche „Spielerei“ für unschädlich. Zum anderen wurde der Satz „So

ist das Kennzeichen in der rauen Umgebung etwas robuster.“ eingefügt. Dieser Satz ist zum Verständnis der Aufgabe nicht nötig, sondern bereitet den nach der Durchführung des Wettbewerbs mit der Aufgabe veröffentlichten Hintergrundtext vor. Dieser „Das ist Informatik!“ Text ist für die Wirkung einer Biber-Aufgabe über den eigentlichen Wettbewerb hinaus sehr wichtig. Deshalb scheint es uns legitim, den Bezug zwischen Aufgabentext und Hintergrundtext durch einen Satz herzustellen, auch wenn dieser für das Verstehen des Aufgabentextes nicht zwingend erforderlich ist.

In der Schlussfassung sehen wir zumindest zwei weitere kritische Punkte:

1. Die Aufgabe verwendet den mathematischen Begriff „Matrix“. Dieser Terminus dürfte den Schülerinnen und Schülern der Klassenstufen 7 und 8, denen die Aufgabe auch gestellt wurde, in der Regel unbekannt sein. Er kann beim Verstehen ignoriert werden, da die Phrase „6 mal 6 Felder“ im Zusammenspiel mit der zentralen Abbildung ausreichend klar ist. Folglich könnte er aber auch weggelassen („Ein Kennzeichen besteht aus 6 mal 6 Feldern“) oder durch einen nicht-fachlichen Begriff ersetzt werden („Ein Kennzeichen ist eine Anordnung von 6 mal 6 Feldern“).

2. Der in der Frage verwendete Begriff „die vier roten Felder“ ist zumindest bedenklich, da ein Kennzeichen laut Definition nur aus schwarzen oder weißen Feldern besteht. Präziser wäre wohl die Formulierung „die vier rot gefärbten Felder“. Für die kürzere Formulierung spricht aber eben ihre Kürze (schnelles Verstehen); außerdem ist die Referenz der Formulierung (nämlich auf die in der Abbildung rot dargestellten inneren Felder des Kennzeichens) ausreichend klar.

4 Fazit

Durch analytische Beobachtung der Arbeit an einer Aufgabe des Informatik-Biber wurde gezeigt, wie die verschiedenen Änderungen der Aufgabe einer Qualitätsmaxime genügen, die sich aus den Randbedingungen der Wettbewerbsdurchführung ergibt. Die Maßnahmen zur Verbesserung der Aufgabenqualität lassen sich zum Teil so verallgemeinern, dass sie auf andere Biber-Aufgaben übertragbar sind. Es bleibt zu untersuchen, inwieweit Techniken, deren Anwendung zu guten Aufgaben für den Informatik-Biber führt, auch allgemein für die Qualität von Aufgaben in der Informatik nützlich sein können. Für eine hohe Qualität der Aufgaben des Informatik-Biber selbst spielen vermutlich noch weitere Aspekte eine wichtige Rolle, die in zukünftigen Arbeiten näher untersucht werden sollen.

Literatur

[Beb] Bebras: International Contest on Informatics and Computer Fluency, http://www.bebras.org/. Letzter Zugriff: 2013-09-04.

[Bib13] Wolfgang Pohl, Hans-Werner Hein und Aimée Eisele, Hrsg. Informatik-Biber: Aufgaben und Lösungen 2012. BWINF Selbstverlag, Februar 2013. http://informatik-

biber.de/assets/files/Informatik-Biber_2012_Web_01032013_mitLoesungen.pdf. Letzter Zugriff: 2013-09-04.

[Dag08] Valentina Dagiene. The BEBRAS Contest on Informatics and Computer Literacy – Students' Drive to Science Education. In *Joint Open and Working IFIP Conference. ICT and Learning for the Net Generation*, Seiten 214–223, Kuala Lumpur, 2008.

[DF08] Valentina Dagiene und Gerald Futschek. Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. In Roland Mittermeir und Maciej Syslo, Hrsg., *Informatics Education – Supporting Computational Thinking*, LNCS 5090, Seiten 19–30, Berlin Heidelberg, 2008. Springer-Verlag.

[ODT06] Mārtiņš Opmanis, Valentina Dagiene und Ahto Truu. Task Types at "Beaver"Contests. In Valentina Dagiene und Roland Mittermeir, Hrsg., *Information Technologies at School: Proceedings of the 2nd International Conference "Informatics in Secondary Schools: Evolution and Perspectives"*, Seiten 509–519, Vilnius, 2006. Institute of Mathematics and Informatics.

[Poh06a] Wolfgang Pohl. Computer Science Contests for Secondary School Students: Approaches to Classification. *Informatics in Education*, 5(1):125–132, 2006.

[Poh06b] Wolfgang Pohl. Wettbewerb im Silberglanz. *LOG IN*, 26(141/142):10–13, 2006.

[PSH09] Wolfgang Pohl, Kirsten Schlüter und Hans-Werner Hein. Informatik-Biber: Informatik-Einstieg und mehr. In Bernhard Koerber, Hrsg., *Zukunft braucht Herkunft — 25 Jahre „INFOS – Informatik und Schule"*, Seiten 38–49, Bonn, 2009. Gesellschaft für Informatik.

[Wal] Gerd Walther. Modul 1: Gute und andere Aufgaben (Arbeitsversion), Mathematikmodul 1 des Programms „Sinus-Transfer Grundschule". http://sinus-transfer-grundschule.de/fileadmin/Materialien/Modu1.pdf. Letzter Zugriff: 2013-09-04.

Lehrerinnen und Lehrer für verständnisintensiven Unterricht im Schulfach Informatik

Otto Thiele

Entwicklungsprogramm für
Unterricht und Lernqualität
Löbstedter Straße 67
D-07743 Jena
otto.thiele@verstehenlernen.de

Abstract: Der pädagogische Paradigmenwechsel von der „Schule des Lehrens zu einer des Lernens“[1] bedeutet für Lehrerinnen und Lehrer, ihr berufliches Handeln so zu professionalisieren, dass sie individuelles Lernen im Unterricht verstehen, begleiten und fördern können. Sich dieser Herausforderung stellend, arbeiten und lernen Thüringer Informatiklehrerinnen und -lehrer theoriegeleitet, fach- und praxisbezogen kollegial zusammen.

1 Lehrerinnen und Lehrer arbeiten und lernen kollegial zusammen

Seit Februar 2012 nehmen Informatiklehrerinnen und -lehrer aus Thüringer Gymnasien, Gesamt- und Berufsschulen an der Fortbildung zum Erwerb des Zertifikats „Lehrer/in für verständnisintensiven Informatikunterricht“ teil. Die Fortbildung ist eine Veranstaltungsreihe des Entwicklungsprogramms für Unterricht und Lernqualität (E.U.LE.)[2] und des Thüringer Instituts für Lehrerfortbildung, Lehrplanentwicklung und Medien (ThILLM), die vom Autor des Praxisberichts, der Trainer für „Verständnisintensives Lernen“ ist und das Konzept der Fortbildung entwickelte, gemeinsam mit einem Berater für „Verständnisintensives Lernen“ durchgeführt wird. Das Anliegen der Fortbildung besteht in einer Professionalisierung der beruflichen Kompetenz der teilnehmenden Lehrerinnen und Lehrer, individuelles Lernen der Schülerinnen und Schüler im Informatikunterricht verstehen, begleiten und fördern zu können. Dementsprechend erhalten die Lehrerinnen und Lehrer Einblicke in die konstruktivistisch orientierte pädagogische Lerntheorie des „Verständnisintensiven Lernens“ und die handlungsorientierte pädagogische Professionstheorie des „Verstehens zweiter Ordnung“. Da die Theorien[3] erst über eine Verankerung in den subjektiven Theorien[4] der Lehrenden handlungsleitend für den Unterricht werden, bekommen die Lehrerinnen und Lehrer in der Fortbildung insbesondere im Praxisfeld Schule und Unterricht vielfältige Trainingsmöglichkeiten geboten, sich ihrer subjektiven Theorien be-

[1] [FRW12, S. 191]
[2] E.U.LE. siehe www.verstehenlernen.de.
[3] Die Theorien gehen auf den Jenaer Schulpädagogen und -entwickler Peter Fauser zurück.
[4] Subjektive Theorien der Lehrerinnen und Lehrer leiten deren Handeln im Unterricht (vgl. [Grz02, S. 178 f.]).

wusst zu werden, diese zu hinterfragen und im Kontext der Lern- und Professionstheorie weiterzuentwickeln oder zu verändern.

Schwerpunkte der Lehrerfortbildung

- Einblicke in die Lerntheorie des „Verständnisintensiven Lernens“ und die Professionstheorie des „Verstehens zweiter Ordnung“
  - Strukturelle und prozessuale Qualitäten von Lernen
  - Professionell kontrollierter Perspektivwechsel
- Subjektive Theorien der Lehrerinnen und Lehrer von Unterricht und Lernen
- Wahrnehmen der Perspektivenvielfalt beim Lernen im Unterricht und Perspektivwechsel
  - Lernen beobachten
  - Lernen verstehen
  - Verständnisintensives Lernen begleiten und fördern

Die Fortbildung umfasst acht Ein- und zwei Zweitagesveranstaltungen, die jeweils an Schulen teilnehmender Lehrerinnen und Lehrer stattfinden. Dadurch kann in jeder Fortbildungsveranstaltung Unterricht besucht oder gehalten werden. Die Fortbildung endet im Dezember 2013 mit der Übergabe der Zertifikate an die Lehrerinnen und Lehrer.

2 Einblicke in die pädagogischen Theorien des „Verständnisintensiven Lernens“ und „Verstehens zweiter Ordnung“

Der Beobachter, der vor Grönland vom Schiff aus Eisberge erblickte und im Foto (Abbildung 1) festhielt, sah deren unterschiedlich geformte Spitzen. Die Kiele (Hauptteile) der Eisberge blieben ihm verborgen, da sie sich unterhalb der Wasseroberfläche befinden. Das Lernen der Schülerinnen und Schüler ist im übertragenen Sinne vergleichbar mit Eisbergen. Im Unterricht haben es Lehrerinnen und Lehrer mit den unterschiedlich geformten „sichtbaren Spitzen“ und „unsichtbaren Kielen“ des Lernens zu tun. Unter Zuhilfenahme geeigneter pädagogischer Lerntheorien können sie sich insbesondere die „unsichtbaren Kiele“ vorstellen und dadurch mitverfolgen, was die Schülerinnen und Schüler beim Lernen im Unterricht verstehen und wie sie es verstehen. Entsprechend können sie das Lernen fördern.
Eine pädagogische Lerntheorie ist die des „Verständnisintensiven Lernens“. Die Lerntheorie geht zum einen davon aus, dass verständnisintensives Lernen durch seine Strukturqua-

Abbildung 1: „Spitzen“ von Eisbergen (Foto: Dr. Joachim Wiesner)

lität, das Zusammenwirken von Erfahrung, Vorstellung, Begreifen[5] und Metakognition bestimmt wird. Erfahrung ist hierbei der Bezug auf die „äußere Wirklichkeit“ und bildet den praktischen, emotionalen Kontext des Erlebens und Handelns im Zusammenhang mit Problemen, Aufgaben sowie Situationen (vgl. [Fau09b, S. 19]). Vorstellung gehört zur „inneren Wirklichkeit“ und ist eine sinnesnahe, erfahrungsanaloge Form des Denkens, durch die Erfahrungen verarbeitet werden und Handeln geplant wird (vgl. ebenda). Begreifen ist logisch-begriffliches Denken, das Bearbeiten von Problemen und Aufgaben mithilfe von abstrakten Kategorien und symbolischen Mitteln (vgl. ebenda). Metakognition ist kritisches, selbstreflexives Begleiten und Optimieren des Lernens (vgl. ebenda), die Fähigkeit, sich mit den eigenen kognitiven Prozessen auseinanderzusetzen (vgl. [Ges08, S. 6]).

Zum anderen geht die Lerntheorie davon aus, dass verständnisintensives Lernen durch die Prozessqualitäten Autonomie- und Kompetenzerfahrung sowie die Erfahrung sozialer Eingebundenheit bestimmt wird (vgl. [Fau03, Fau09b, S. 263 ff., S. 19]). Autonomie lässt sich u. a. mit den Worten Johann Gottlieb Fichtes beschreiben, dass als gebildet gilt, wer „ganzheitlich aus sich selbst heraus will, was [...] erfordert ist“ [Fic62, S. 392 ff.]. Kompetenz bezeichnet die „Fähigkeit, intelligentes Wissen mit intelligentem Handeln zu verbinden“ [Fau09a, Folie 17]. Soziale Eingebundenheit bedeutet, „dass der Mensch die angeborene motivationale Tendenz hat, sich mit anderen Personen in einem sozialen Milieu verbunden zu fühlen, in diesem Milieu effektiv zu wirken (zu funktionieren) und sich dabei persönlich autonom und initiativ zu erfahren“ [DR93, S. 229].

Verständnisintensives Lernen der Schülerinnen und Schüler bedarf eines verständnisintensiven Unterrichts. Diesbezüglich verweist die pädagogische Professionstheorie des „Verstehens zweiter Ordnung“ auf die von Lehrerinnen und Lehrern benötigte Handlungskompetenz, im Unterricht einen professionell kontrollierten Perspektivwechsel zu vollziehen, um die Differenzen zwischen dem eigenen Verstehen und dem der Lernenden wahrnehmen

[5] Verstehen und Begreifen werden oftmals gleichbedeutend verwendet. In der pädagogischen Lerntheorie des „Verständnisintensiven Lernens“ sind sie wissenschaftliche Begriffe, wobei Begreifen als eine Strukturdimension des Verstehens aufgefasst wird.

und dadurch deren Lernenprozesse ko-konstruktiv begleiten und fördern zu können (vgl. [FRW12, S. 185]). Ko-Konstruktion bringt die Lehrerinnen und Lehrer in eine neue Beziehung mit den Lernenden, in der die Einen nicht nur Wissen an die Anderen vermitteln, sondern in der es im Sinne eines systemischen Ansatzes zu einem produktiven Zusammenwirken von „Menschen mit unterschiedlichen Kompetenzen“ kommt (vgl. [NH09, S. 16]). Lernen wird somit insbesondere durch Interaktion und Zusammenarbeit bestimmt, in der die Lehrerinnen und Lehrer gemeinsam mit den Lernenden das Interesse am Unterrichtsstoff teilen, Vorhaben und Tätigkeiten ausführen und darüber reflektieren, was beim Lernen erlebt, gefühlt und verstanden wurde (vgl. [Fth08, S. 1]).

Ein Beispiel verständnisintensiven Lernens

Das Thema der hier beschriebenen Unterrichtsstunde im Kurs Informatik der Oberstufe eines Gymnasiums war „Formale Sprachen“. Der Arbeitsauftrag an die Schülerinnen und Schüler lautete: Wählt aus einer Reihe vorgegebener Gegenstände einen Gegenstand aus und findet heraus, was der Gegenstand mit dem Thema zu tun hat!
Zwei Schülerinnen, die sich als Lerntandem zusammenfanden, wählten Taschenrechner, ein älteres und ein neueres Modell, aus. Die Taschenrechner sind in der Abbildung 2

Abbildung 2: UPN-Taschenrechner (Foto: Otto Thiele)

zu sehen. Was die Schülerinnen nicht wussten, es handelte sich um UPN-Taschenrechner[6]. Nachdem die Schülerinnen sich die UPN-Taschenrechner eingehender angesehen hatten, legten sie das ältere Modell beiseite und beschäftigten sich ausschließlich mit dem neueren. Wie im Umgang mit ihren Schultaschenrechnern gewohnt, gaben sie dem UPN-Taschenrechner Ausdrücke in der Infixnotation, beispielsweise $2 + 3$ und $5 - 2$, ein. Ir-

[6] UPN ist die Abkürzung für Umgekehrte Polnische Notation.

ritiert waren die beiden zum einen darüber, dass der UPN-Taschenrechner zum Bilden von Ergebnissen keine „=“-Taste besaß. Deshalb schlossen sie die Eingaben der Ausdrücke mit der ENTER-Taste ab. Zum anderen waren sie irritiert, weil auf dem Display des UPN-Taschenrechners keine korrekten Ergebnisse, etwa für die Summe $2 + 3$ und die Differenz $5 - 2$, ausgegeben wurden. Aufgrund der Irritationen nahmen die Schülerinnen einen ihrer Schultaschenrechner zu Hilfe. Sie gaben sowohl dem Schul- als auch

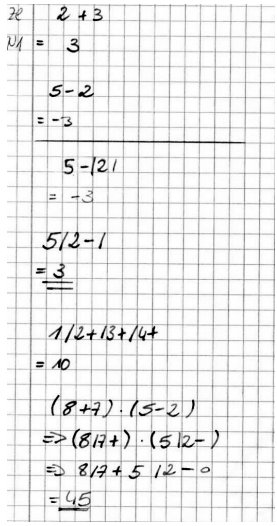


Abbildung 3: Notizen der Schülerinnen (Foto: Otto Thiele)

dem UPN-Taschenrechner Ausdrücke in der Infixnotation ein, verglichen die Ergebnisse miteinander und stellten fest, dass diese nicht gleich waren. Durch weiteres Probieren, Vergleichen und Nachdenken fanden die beiden heraus, dass der von ihnen ausgewählte UPN-Taschenrechner Ausdrücke in einer ihnen bislang unbekannten Notation berechnete. Nach und nach erschlossen sie sich anhand selbst gewählter Beispiele die syntaktische Struktur der unbekannten Ausdrücke. In ihren Notizen, die in der Abbildung 3 zu sehen sind, gaben die Schülerinnen die Tastenfolgen der zu berechnenden Ausdrücke in der für sie „neuartigen“ Notation, beispielsweise $8 \mid 7 + 5 \mid 2 - \cdot$, an. Der senkrechte Strich symbolisiert das Betätigen der ENTER-Taste des UPN-Taschenrechners. Des Weiteren fanden beide heraus, dass der UPN-Taschenrechner, im Unterschied zum Schultaschenrechner, deshalb keine Klammertasten besitzt, weil Ausdrücke in der „neuartigen“ Notation keine Klammerung benötigen.

Beschreibung des Lernens der Schülerinnen mithilfe der pädagogischen Lerntheorie

Bei der Entscheidung, die UPN-Taschenrechner zu wählen, so die Schülerinnen, ließen sie sich von Vorstellungen leiten, die auf ihren Erfahrungen im Umgang mit dem Schulrechner beruhten. Da das Design des UPN-Taschenrechners aus den 1970er Jahren, der in der Abbildung 2 rechts zu sehen ist, am wenigsten dem des Schultaschenrechners glich, beschäftigten sich die beiden mit diesem Taschenrechner nicht weiter und entschieden sich für das neuere UPN-Taschenrechner-Modell. Aufgrund der ähnlichen äußeren Form des Schul- und UPN-Taschenrechners gingen die Schülerinnen davon aus, die Taschenrechner berechnen Ausdrücke in der ihnen bekannten Infixnotation. Deshalb empfanden sie das, womit sie sich beschäftigen wollten, als einfach. „Ich wusste nicht", so eine der Schülerinnen, „was ich mit einem recht einfachen Thema [...] machen sollte". Diese Meinung beruhte auf der fehlenden Erfahrung in der Handhabung andersgearteter Taschenrechner. Erst durch ihr Handeln, das heißt, im praktischen Umgang mit dem UPN-Taschenrechner machten die Schülerinnen die Erfahrung, dass dieser in der Infixnotation eingegebene Ausdrücke nicht wie von ihnen erwartet berechnete. Das veranlasste beide zum Nachdenken und zur Zuhilfenahme des Schultaschenrechners. Versuch und Irrtum durch wiederholtes Eingeben von Ausdrücken in der Infixnotation sowie das Wahrnehmen der unterschiedlichen Ergebnisse beider Taschenrechner waren mit der Fragestellung verbunden, weshalb der UPN-Taschenrechner für Infixausdrücke nicht die erwarteten Ergebnisse ausgab. Da die Schülerinnen ihre Vorgehensweise, den Schul- und den UPN-Tascherrechner in gleicher Weise zu bedienen, infrage stellten (Metakognition), erschlossen sie sich durch weiteres Überlegen (Begreifen) und Probieren (Erfahrung) nach und nach die ihnen unbekannte Postfixnotation von Ausdrücken.

Nach dem Unterricht reflektierte der Informatiklehrer, der die beiden Schülerinnen beim Lernen beobachtete, dass er der Meinung gewesen sei, die bis dato „leistungsschwachen" Schülerinnen würden mit den UPN-Taschenrechnern nicht zurechtkommen. Seine Meinung schien sich nach seinen anfänglichen Beobachtungen auch zu bestätigen. Erstaunt sei er allerdings gewesen, wie sich die beiden zunehmend selbstbestimmter und erfolgreicher mit einem der UPN-Taschenrechner beschäftigten. Als die Schülerinnen selbstständig herausfanden, dass der UPN-Taschenrechner, anders als der Schultaschenrechner, die Postfixnotation verwendet, sagte der Lehrer, hätte er seine anfängliche Meinung ändern müssen. Eine Erfahrung, die ihn zum Nachdenken veranlasste.
Im Gespräch mit den Schülerinnen kam zum Ausdruck, dass sie beim Lernen die Erfahrungen machten, autonom, kompetent und eingebunden gewesen zu sein. Das bedeutet, beide lernten zunehmend selbstbestimmter, fanden eigenständig die Notation der Ausdrücke, die der UPN-Taschenrechner berechnete und erbrachten eine von den Mitschülerinnen, -schülern sowie dem Lehrer anerkannte Leistung.

Ein Beispiel „Verstehens zweiter Ordnung"

Ein Schüler hatte im Informatikunterricht zum Themenbereich „Algorithmen und Datenstrukturen" beim Entwickeln von Computerprogrammen ein Problem mit der Datenstruk-

tur Reihung (eindimensionales Array). Er nahm an, dass in einer Reihung die Indizes der Elemente deren Werten entsprechen. Ein Problem, das Lernende häufiger im Umgang mit Reihungen haben. Der Lehrer wurde auf das Problem des Schülers aufmerksam und versuchte, ihm anhand einer Skizze, die in der Abbildung 4 dargestellt ist, den Aufbau einer Reihung zu erklären. Der Schüler intervenierte jedoch, da er sich eine Reihung nicht so

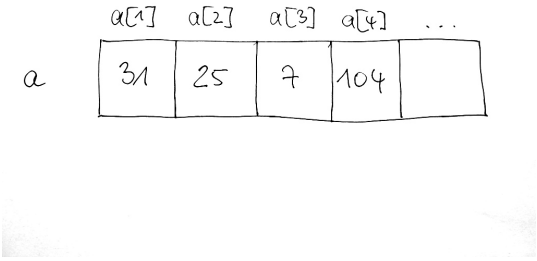


Abbildung 4: Skizze des Lehrers (Foto: Otto Thiele)

vorstellte, wie der Lehrer sie skizzierte und zu erklären versuchte. Daraufhin fragte der Lehrer, wie sich der Schüler denn eine Reihung vorstelle. Dieser antwortete, eine Reihung sei für ihn so etwas wie ein ICE-Zug. Um dem Schüler beim Verstehen zu helfen, ging der Lehrer auf dessen Vorstellung ein und regte ihn an, Analogien zwischen einem ICE-Zug und einer Reihung zu bilden. Der Vorstellung des Schülers entsprach, dass die Elemente einer Reihung wie die Wagen eines ICE-Zuges aneinandergereiht sind. Durch weitere Analogiebildung fand der Schüler sukzessive heraus, die Zugnummer entspricht dem Namen, die Wagennummern entsprechen den Indizes der Elemente und die Inhalte der Wagen (Fahrgäste) den Werten der Elemente einer Reihung. Aufgrund der Analogiebildung kam der Schüler zu der Erkenntnis, dass bei einer Reihung die Indizes der Elemente nicht deren Werte sind, so wie beim ICE-Zug die Nummern der Wagen nicht deren Inhalte sind. Dadurch gelang es dem Schüler im weiteren Verlauf des Informatikunterrichts, Computerprogramme zu entwickeln, in denen korrekt auf die Elemente einer Reihung zugegriffen wurde.

„Verstehen zweiter Ordnung“ als berufliche Kompetenz des Lehrers

Ausgelöst durch die Intervention des Schülers nahm der Lehrer die Differenz zwischen seinem Verstehen und dem, das der Schüler entwickelt hatte, wahr. Die Differenz bestand darin, dass er als Experte über ein konzeptuelles Modell verfügte, das eine fachliche Repräsentation einer Reihung darstellte. Dieses Modell übernahm der Schüler trotz des Erklärungsversuchs des Lehrers nicht. Der Schüler hatte ein eigenes, auf seiner Alltagserfahrung beruhendes mentales Modell entwickelt, das eine vereinfachte Repräsentation einer

Reihung durch einen ICE-Zug darstellte. Die Differenzerfahrung veranlasste den Lehrer, auf die Vorstellung, das mentale Modell, des Schülers einzugehen und dessen Lernprozess ko-konstruktiv zu begleiten. So konnte der Schüler mittels seines eigenen Modells, das anfänglich mehr einer verketteten Liste glich, durch die Anregung des Lehrers zur Analogiebildung ein fachliches Verstehen der Datenstruktur Reihung entwickeln. Die Handlungskompetenz des Lehrers, im Unterricht professionell kontrolliert die Perspektive zu wechseln, wodurch er das Lernen des Schülers verstand und entsprechend fördern konnte, wird, wie bereits im Abschnitt 2 erwähnt, als „Verstehen zweiter Ordnung“ bezeichnet.

3 Wahrnehmung der Perspektivenvielfalt beim Lernen und Perspektivwechsel

Den Kern der Fortbildung zum Erwerb des Zertifikats „Lehrer/in für verständnisintensiven Informatikunterricht“ bildet ein Training, das den Lehrerinnen und Lehrern Wege zum Erwerb der pädagogischen Handlungskompetenz des „Verstehens zweiter Ordnung“ (siehe Abschnitt 2) aufzeigt. Das Training umfasst alle Fortbildungsveranstaltungen und wird stets in Verbindung mit Unterricht durchgeführt. Es ist auf auf das pädagogische Handeln (Lernen beobachten, verstehen und fördern) der Lehrerinnen und Lehrer ausgerichtet und damit verbunden, sie zu befähigen, theoriegeleitet über ihr Handeln zu reflektieren.

Drei aufeinander aufbauende Bausteine des Trainings

Lernen beobachten

Das bewusste Wahrnehmen der unterschiedlichen Perspektiven Lernender erfolgt dadurch, dass die Lehrerinnen und Lehrer anhand von Unterrichtsaufzeichnungen auf Videofilmen das Lernen von Schülerinnen und Schülern beobachten und sich über das Beobachtete austauschen. Dies sind vorbereitende Übungen für sich anschließende Unterrichtsbesuche in Schulfächern, welche die Lehrerinnen und Lehrer selbst nicht unterrichten, wie beispielsweise Englisch, Geschichte, Ethik oder Kunsterziehung. Der Besuch fachfremden Unterrichts fokussiert das Beobachten stärker auf die Lernenden und den Beobachtern fällt es leichter, die Unterrichtenden auszublenden, denn nicht das Lehren, sondern das Lernen soll beobachtet werden. Behilflich für das Beobachten von Lernen sind Aufträge, die von den Fortbildnern und Unterrichtenden vergeben werden, wie zum Beispiel darauf zu achten, ob die Schülerinnen und Schüler selbstbestimmt im Unterricht lernen. Im Anschluss an die Unterrichtsbesuche finden Reflexionsgespräche über das beobachtete Lernen mit den Unterrichtenden und innerhalb der Lehrergruppe statt.

Lernen verstehen

Die Unterrichtsbesuche werden im weiteren Verlauf des Trainings mit dem Ziel durchgeführt, tiefgründiger in die Perspektive der Lernenden zu wechseln, um deren Lernen immer besser verstehen zu können. Dazu begleiten die Lehrerinnen und Lehrer jeweils

einzelne Lernende im Unterricht. Das ermöglicht den Lehrerinnen und Lehrern, im Verlauf des Unterrichts mit den Schülerinnen und Schülern über deren Lernen zu sprechen und ihnen diesbezüglich Fragen zu stellen. Durch diese enge Begleitung wird es den Lehrerinnen und Lehrern möglich, die Lernenden auf beobachtete Erscheinungen beim Lernen anzusprechen und deren Ursachen zu ergründen. Was könnte beispielsweise die Ursache dafür sein, dass sich eine Schülerin oder ein Schüler nicht in eine Lerngruppe integrierte? Nach den Unterrichtsbesuchen (Lernbegleitungen) finden Reflexionsgespräche der Lehrerinnen und Lehrer mit den Lernenden, Unterrichtenden und untereinander statt.

Verständnisintensives Lernen fördern

Mit dem Fokus auf verständnisintensives Lernen wird kollegial und theoriegeleitet Informatikunterricht in der Lehrergruppe geplant, durchgeführt und reflektiert. Dabei sammeln die Lehrerinnen und Lehrer Erfahrungen, ob und wie sich die pädagogische Lerntheorie des „Verständnisintensiven Lernens“ und Professionstheorie des „Verstehens zweiter Ordnung“ über ihre subjektiven Theorien handlungsleitend im Unterricht auf das Fördern des Lernens auswirken.
Die Informatikstunden stellt, in Absprache mit der Schulleitung und der Fachkonferenz[7], die jeweils gastgebende Schule zur Verfügung. Die Unterrichtsthemen werden mit den Informatiklehrerinnen und -lehrern der Schule abgestimmt. Besonders wichtig ist Transparenz gegenüber den Schülerinnen und Schülern, dass der Unterricht deshalb in veränderter Form stattfindet und von Gastlehrerinnen und -lehrern durchgeführt wird, weil diese gemeinsam mit ihnen ausprobieren wollen, wie verständnisintensives Lernen im Informatikunterricht verstanden, begleitet und gefördert werden kann. Im Übrigen ist das eine interessante Erfahrung für die Schülerinnen und Schüler, dass auch Lehrerinnen und Lehrer in ihrem Beruf Lernende sind. Der Unterricht selbst findet in Lerngruppen (Gruppenunterricht) statt, wobei möglichst nur eine Lehrerin oder ein Lehrer mit einer Schülergruppe zusammenarbeitet. Sie bzw. er gibt der Lerngruppe das Thema und eine Problem- oder Aufgabenstellung vor und begleitet die Schülerinnen und Schüler beim Lernen. Beim Begleiten der Lernenden kommt es für die Lehrerinnen und Lehrer vor allem darauf an, durch einen professionell kontrollierten Perspektivwechsel herauszufinden, wann, weshalb und wie im Unterricht interveniert werden sollte, so dass sich Intervenieren in Lernprozessen als förderlich erweist. Die gastgebenden Informatiklehrerinnen und -lehrer beobachten während des Unterrichts gemeinsam mit dem Trainer und Berater das Lernen der Schülerinnen und Schüler. Nach dem Unterricht erfolgt die Reflexion zuerst mit den Lernenden, was sich für die Lehrerinnen und Lehrer als besonders aufschluss- und hilfreich erweist. Danach findet in der Lehrergruppe gemeinsam mit den Lernbeobachtern die Reflexion über den Unterricht statt.

4 Resümee

Eine Informatiklehrerin hat auf die Frage, was sie aus der Fortbildung mit in ihren Unterricht nimmt, geantwortet, dass sie sich noch mehr auf die Schülerinnen und Schüler

[7] Fachkonferenz siehe Thüringer Schulgesetz (ThürSchlG).

einstellen möchte. „Öfter mal Perspektivwechsel betreiben, um Schwierigkeiten der Lernenden beim Verstehen der Unterrichtsinhalte nachvollziehen zu können. Dann kann ich durch die Gestaltung verschiedener Lernsituationen den einzelnen Schülern die Möglichkeit schaffen, Lernerfolge zu erleben. So können die Motivation und das Verstehen der Schüler verbessert werden." Und sie, so die Lehrerin, versucht, in allen Unterrichtssituationen die Ideen und Lösungen der Lernenden zu berücksichtigen.

Die im Praxisbericht vorgestellte Fortbildung „Lehrer/in für verständnisintensiven Informatikunterricht" leistet durch eine enge Verknüpfung von Theorie und Praxis einen Beitrag, aktuelle und moderne Erkenntnisse pädagogischer Lern- und Professionsforschung in den Unterricht der Lehrerinnen und Lehrer im Schulfach Informatik einfließen zu lassen.

Literatur

[DR93] Deci, E. und Ryan R. Die Selbstbestimmungstheorie der Motivation und ihre Bedeutung für die Pädagogik. *Zeitschrift für Pädagogik*, (39): Seiten 223–237, 1993.

[Fau03] Fauser, P. Lernen als innere Wirklichkeit. Über Imagination, Lernen und Verstehen. In Rentsch, I., Madelung, E. und Fauser, P., Hrsg., *Bilder im Kopf. Texte zum Imaginativen Lernen*, Seiten 242–286. Kallmeyer, 2003.

[Fau09a] Fauser, P. Gelungener Umgang mit Heterogenität in der Schule. Grundlagen, Dimensionen und Beispiele, 2009.

[Fau09b] Fauser, P. Lernen in der Schule. In Fauser, P., Prenzel, M. und Schratz, M., Hrsg., *Was für Schulen! Wie gute Schule gemacht wird – Werkzeuge exzellenter Praxis*. Klett, Kallmeyer, 2009.

[Fic62] Fichte, J. G. *Ausgewählte Werke in sechs Bänden. Band V*. Wissenschaftliche Buchgesellschaft, 1962.

[FRW12] Fauser, P., Rißmann, J. und Weyrauch, A. Das Entwicklungsprogramm für Unterricht und Lernqualität — Theoriegeleitete Intervention als Professionalisierungsansatz. In Kraler, Ch., Schnabel-Schüle, H., Schratz, M. und Weyland, B., Hrsg., *Kultur(en) der Lehrerbildung. Professionalisierung eines Berufsstands im Wandel*, Seiten 177–194. Waxmann, 2012.

[Fth08] Fthenakis, W. In hundert Sprachen ko-konstruieren. Sechs Fragen an Prof. Dr. Dr. Dr. Wassilios Fthenakis. *Betrifft KINDER*, (06-07), 2008.

[Ges08] Gesellschaft für Informatik (GI) e. V. Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. *Beilage zu LOG IN*, 28. Jg.(150/151), 2008.

[Grz02] Grzesik, J. *Effektiv lernen durch guten Unterricht. Optimierung des Lernens im Unterricht durch systemgerechte Formen der Zusammenarbeit zwischen Lehrern und Schülern*. Klinkhardt, 2002.

[NH09] Niederegger, P. P. und Hofer, U. Über das Lernen sprechen. Lerngespräche fördern die Beziehung und das Lernen, 2009.

Schülerprojekte - zwischen Theoriekurs und Firmenpraxis

Steffi Heinicke, Dr. Michael Unger

Schülerrechenzentrum
TU Dresden
Gret-Palucca-Straße 1
01069 Dresden
steffi.heinicke@srz-dresden.de
michael.unger@srz-dresden.de

Abstract: Das Schülerrechenzentrum der Technischen Universität Dresden (SRZ) ist ein Zentrum der Begabtenförderung für Schüler in den Bereichen Informatik und Elektronik. Neben der Arbeit in Theoriekursen und der praktischen Beschäftigung mit Computer und Lötkolben fertigt jeder Schüler im Laufe eines Jahres eine Projektarbeit an. Ausgewählte Schülerinnen und Schüler des SRZ erhalten die Möglichkeit, ein Projekt eines Kooperationspartners des SRZ im engen Kontakt mit den Mitarbeitern zu erstellen. Bei einem Firmentag im Oktober stellen Dresdner IT-Unternehmen ihre Ideen für solche Schülerprojekte vor und führen bereits erste Gespräche mit den dabei entstandenen Projektteams. Im Verlauf des Schuljahres stellen die Schüler in Zusammenarbeit mit den Firmen die Projekte fertig. Höhepunkt der Arbeit ist die Verteidigung des Projekts vor einem Gremium aus Firmenvertretern, Mitarbeitern der TU Dresden und Gästen.

1 Das Schülerrechenzentrum der TU Dresden

1.1 Organisation

Das Schülerrechenzentrum der Technischen Universität Dresden (SRZ) ist ein Zentrum für Begabtenförderung auf den Gebieten Informatik und Elektronik. Es bietet Schülern und Schülerinnen eine Möglichkeit zur Vorbereitung auf Studium bzw. Berufsausbildung und dient gleichzeitig als Konsultationszentrum für Schüler und Lehrer.

Partner des SRZ sind neben der Technischen Universität Dresden als Träger das Sächsische Staatsministerium für Kultus, die Stadt Dresden und verschiedene Unternehmen aus der Region. Diese unterstützen das SRZ nicht nur finanziell und materiell, sondern bringen auch Ideen zur inhaltlichen Gestaltung der Kurse und für Projektarbeiten der Schüler ein.

1.2 Kursangebot, Kursaufbau

Für die beiden Gebiete Informatik und Elektronik existiert je ein Angebot an aufeinander aufbauenden Jahreskursen. Um einen Einblick in ein spezielles Thema zu erhalten oder

eine Thematik unter einer anderen Sicht zu behandeln, werden Sonderkurse angeboten. Für jüngere Schüler ohne Vorkenntnisse wurden Vorbereitungskurse konzipiert, die den Schülern einen Einblick in die Informatik oder Elektronik ermöglichen sollen. Ausgewählte Schüler erhalten im Laufe des Schuljahres die Möglichkeit, an Spezialkursen mit anspruchsvollen Themen teilzunehmen. Jedes Jahr bewerben sich etwa 120 Schüler/innen aus 40 Schulen aus Dresden und Umgebung. Sie arbeiten in 15 Arbeitsgemeinschaften, davon 9 mit Theoriekurs.

Am Ende des Schuljahres erhalten die Schüler ein Zeugnis, in welchem ihre belegten Kurse, das Thema der Jahresarbeit und die dabei erreichten Leistungen dokumentiert werden. Ausgewählte Jahreskurse können als Informatik-Grundkurs im Abitur angerechnet werden. Einen Überblick über die Kurse im Schuljahr 2012/13 zeigt Abbildung 1.

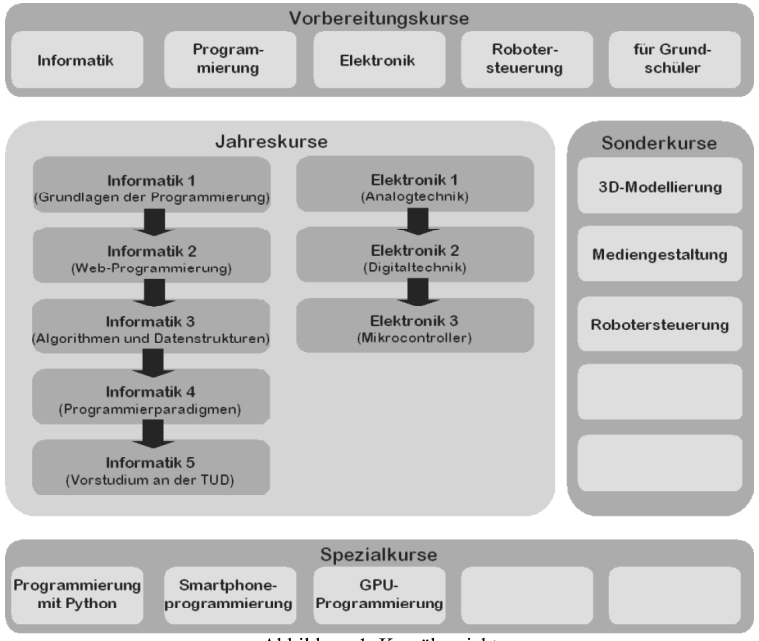


Abbildung 1: Kursübersicht

1.3 Schülerprojekte

Alle Schüler erarbeiten im Laufe eines Schuljahres eine Projektarbeit. Im Rahmen dieser Arbeit werden die Schüler angeregt, im Team zu arbeiten, planmäßig an komplexe Aufgaben heranzugehen und ihre Produkte zu präsentieren.

Bei der Arbeit am Projekt ist ein erster wichtiger Schritt die Themenwahl, die selbständig durch den Schüler erfolgt. Ebenso gehört zur Jahresarbeit die Erstellung eines Pflichtenheftes am Anfang der Projektphase. Das Pflichtenheft soll den Schülern helfen, die Arbeit am Projekt besser zu planen. Dazu soll der Schüler das Thema spezifizieren, die Arbeitsschritte, Teilziele bei der Lösung des Problems festlegen und einen Zeitplan erstellen. Die Erstellung einer projektbegleitenden Dokumentation und eine Zwischenverteidigung gegen Ende der Projektphase soll den Schülern ermöglichen, während der Projektarbeit Zwischenziele zu evaluieren und eventuell Anpassungen an den Projektzielen vorzunehmen. Die öffentliche Verteidigung der Arbeit vor Schülern, Mitarbeitern des SRZ und Gästen ist der Höhepunkt der Arbeit. Bewertet werden am Ende die Qualität der Arbeit und der Dokumentation, die Kreativität beim Finden des Themas und die projektangepasste Verteidigung.

2 Firmenprojekte

2.1 Anliegen

Bereits in früheren Jahren hatten einige Schüler in Eigeninitiative Partner für ihre Jahresarbeiten gesucht und gefunden. Diese Projekte hatten meist anspruchsvolle Inhalte und wurden sehr gut bewertet. Sie brachten nicht nur dem jeweiligen Partner einen praktischen Nutzen, sondern auch dem Schüler die Befriedigung, mit seiner Arbeit einen bleibenden Wert geschaffen zu haben.

Um die Vorteile solcher Projetthemen auch anderen Schülern bieten zu können wurden bestehende Kontakte zu Dresdner Softwarefirmen genutzt, um gezielt nach Themen für Firmenprojekten zu suchen. Mit dieser Verbindung von Unterricht und Wirtschaft reiht sich das Schülerrechenzentrum in eine Reihe von Initiativen ein, die ein ähnliches Anliegen verfolgen, zum Beispiel

- Projekt „Universum Wirtschaft!“ der Stiftung der Deutschen Wirtschaft (sdw) gGmbH
- Projekt „Junior“ des Instituts der deutschen Wirtschaft Köln
- Schülerprojekt „Zukunft MINT“ der Initiative „Campus of Excellence“

Die Teamarbeit von Schülern, Lehrern und Vertretern der Firmen stellt alle Beteiligten vor neue Herausforderungen, insbesondere werden von den Schülern weit mehr als im üblichen schulischen Rahmen soziale Kompetenzen gefordert. Sie müssen gemeinsam mit allen Beteiligten einen Terminplan aufstellen und einhalten, arbeiten häufig direkt in der Firma mit den dort Beschäftigten zusammen und nutzen für sie neue firmeninterne Kommunikationsformen.

2.2 Ablauf

Bereits seit 2009 werden regelmäßig Firmenprojekte am SRZ realisiert. Im Verlauf der Entwicklung ergaben sich durch alle Beteiligten Hinweise zur besseren Umsetzung der Ziele, aus denen sich der im folgenden skizzierte Jahresablauf entwickelt hat.

Zur gezielten Vorbereitung der Schüler findet seit dem Schuljahr 2010/11 jeweils zu Schuljahresbeginn ein Spezialkurs „Einführung in das Projektmanagement“ statt.

Nach den Herbstferien veranstaltet das SRZ einen „Firmentag“, an dem Vertreter der Unternehmen ihre Projektideen vorstellen. Im Anschluss können sich die Schüler für diese Projekte bewerben (siehe Abbildung 2). Dabei stellen sie sich vor, begründen ihr Interesse an dem jeweiligen Projekt und lassen sich die speziellen Anforderungen genauer erklären. Auf diesem Weg wurden im aktuellen Schuljahr 9 Projektgruppen gebildet, an denen insgesamt 17 Schüler beteiligt waren.

Abbildung 2: Firmentag

Der nächste Schritt ist die gemeinsame Erstellung eines Pflichtenhefts. Dieses beinhaltet die konkrete Spezifikation der Aufgabe, Verfahren zur Abstimmung der Projektpartner untereinander und einen detaillierten Terminplan. Die Zeit von November bis Mai ist für die konkrete Arbeit am Projekt vorgesehen. In dieser Phase arbeiten die Schüler individuell an Teilprojekten, halten Kontakt untereinander und zu den Betreuern und treffen sich regelmäßig zu Konsultationen und Absprachen. Parallel dazu entsteht die Dokumentation des Projektes.

Abbildung 3: Verteidigung bei queo

Wie bei wissenschaftlichen Arbeiten üblich wird die Projektarbeit verteidigt. Für die Schüler ist es eine besondere Anerkennung, wenn diese Veranstaltung in der jeweiligen Partnerfirma stattfindet. Dazu sind jeweils die anderen Schüler, Vertreter des Fördervereins des SRZ, weitere Partner und Mitarbeiter der gastgebenden Firma eingeladen. Nach der Präsentation der Ergebnisse haben die Gäste die Möglichkeit, Fragen an die Vortragenden zu stellen. Vertreter des SRZ und der Partnerfirmen bewerten die Ergebnisse gemeinsam.

Zum Tag der offenen Tür des SRZ erhalten die besten Jahresarbeiten einen von Firmen gestitfteten „SRZ-Preis“. Dazu werden an diesem Tag ausgewählte Jahresarbeiten öffentlich vorgestellt und begutachtet. Die Fachjury wird von Mitarbeitern der Firmen und Dresdner Hochschulen gebildet.

2.3 Beispiele 1: T-System – Widgets für das firmeninterne Dashboard

Abbildung 4: Uhr-Widget

Seit 2010 arbeitetet das SRZ sehr eng mit der Firma T-Systems Multimedia Solutions GmbH in Dresden zusammen. In jedem Schuljahr konnten für die Projekte zum Thema „Widgets für das Dashboard der T-Systems Multimedia Solutions GmbH“ Schüler gewonnen werden. An dieser Aufgabenstellung wird sehr gut deutlich, welche Art von Projektaufgaben für Schülerprojekte geeignet sind. Die Aufgabe ist so konkret und interessant, dass ein Schüler sie ohne große Erläuterung verstehen kann, birgt aber so viele Potenzen, dass die Ausarbeitung der Projekte auf ganz unterschiedlichen Niveaustufen erfolgen kann. Bei der ersten Konsultation mit dem Projektbetreuer wurde das Dashboard vorgestellt und die Schüler entwickelten ihre Ideen und konzipierten ein Pflichtenheft. Im aktuellen Schuljahr entstand folgende Aufgabenstellung:

Abbildung 5: Twitter-Widget

Die Idee des Projekts ist die Erstellung von Widgets für das Dashboard der T-Systems MMS GmbH. Diese sollen den Mitarbeitern verschiedene Funktionen zur Verfügung stellen, für eine Abwechslung im Dashboard sorgen und das Dashboard attraktiver machen. Die Idee gibt es bereits seit 2 Jahren, bisher entstanden ein Twitter-, Speiseplan-, Social-, Blog- und Wetterwidget, sowie eines für die Anzeige der aktuellen Uhrzeit. Unsere Idee war es das Speiseplan- und Social-Widget zu verbessern. Als neue Widgets sollten ein Sudoku- und ein Terminfindungs-Widget entstehen.[AK13]

Da in jedem Jahr Schüler sehr engagiert an diesem Projekt arbeiten, fühlen sie sich auch für die Pflege und Aktualisierung der von ihren Vorgängern erstellten Programme verantwortlich. Außerdem können sie die Erfahrungen ihrer Vorgänger nutzen und deren Ideen weiterentwickeln. Diese Arbeitsweise prägt auch die Projektphasen die alle Schüler in diesen Projektgruppen durchlaufen. Nach der Ideenfindung beginnt die Arbeit mit der Analyse der bestehenden Widgets und deren Anpassung. Danach wird die eigene Idee realisiert. Besonders anspruchsvoll wird die Aufgabe auch dadurch, dass für die Veröffentlichung auf den T-System Dashboard strenge Regeln vorgegeben sind, die die Schüler beachten müssen. Diese betreffen sowohl das Layout, als auch die internen und externe Sicherheitsrichtlinien. Wie alle Widgets müssen auch die Schülerwidgets durch eine Kontrollkommission von T-System freigegeben werden.

2.4 Beispiele 2: i.S.X. Software GmbH - Optimierung des Programms "TeamCal"

Mit der Firma i.S.X. ist das SRZ seit einiger Zeit in gutem Kontakt. Mitarbeiter dieses Unternehmens leiten jedes Jahr den Spezialkurs „Einführung in das Projektmanagement" um die Schüler optimal auf die Arbeit in den Firmen vorzubereiten. Einer dieser Mitarbeiter ist ein ehemaliger SRZ-Schüler und dadurch sehr gut mit den Abläufen im SRZ vertraut.

Die Aufgabenstellung bestand in diesem Fall darin, das Programm „TeamCal", welches für interne Abläufe im Unternehmen genutzt wird, zu optimieren. Es handelt sich um ein Programm zur Zeitplanung für Mitarbeiter. Mit steigender Mitarbeiteranzahl erhöhten sich aber die Zugriffszeiten so dramatisch, dass der Einsatz des Programmes infrage gestellt war. Die Schüler sollten das Programm auf zeitintensive Datenbankzugriffe analysieren und Vorschläge für Veränderungen erarbeiten und umsetzen.

Diese Projektaufgabe zeichnet sich dadurch aus, dass die Schüler an einer Aufgabe arbeiten, deren Relevanz für das Unternehmen sie konkret erkennen können. Damit erhalten sie noch eine zusätzliche Motivation, die Aufgabe termingerecht und in hoher Qualität zu lösen. Dieses Projekt war auch besonders geeignet um die Schüler anzuregen, sich selbständig mit Inhalten der Informatik auseinanderzusetzen, die nicht explizit im aktuell besuchten Kurs behandelt werden.

Im Ergebnis dieser Arbeit konnten die Schüler ein Programm vorlegen, welches eine Geschwindigkeitssteigerung um 30% im Vergleich zum Ausgangswert erreichte. In ihrer Dokumentation haben die Schüler folgende Einschätzung ihrer Arbeit abgegeben.

Unserer Meinung nach waren die Forderungen an uns, mit unserem momentanen Wissensstand, relativ komplex und schwierig. Allerdings haben wir uns dann mit zunehmender Zeit und Hilfe unseres Betreuers immer mehr in das Programm rein gefunden und konnten schließlich doch eine recht gute Arbeit abliefern. Durch die Arbeit an dem Programm bekamen wir einen für uns tieferen Einblick in PHP und MySQL, ... [BBS11]

2.5 Ergebnisse

Die Organsation von Schülerprojekten mit regional ansässigen Firmen hat sich seit 2009 zu einer festen Institution im SRZ entwickelt. Viele neue Kontakte zu Unternehmen sind entstanden und waren Startpunkt weiterführender Kooperationen. Einige Schüler konnten in ihren damaligen Projetfirmen Praktika beginnen oder sind als Werksstudenten tätig.

Zur Weiterzuentwicklung dieser Arbeit ist vorgesehen, im nächsten Schuljahr einen Mitarbeiter für die Betreuung aller Firmenprojekte zu finden. Dadurch sollen Abstimmungsschwierigkeiten zwischen Schüleren, Firmenbetreuern, AG-Leitern und SRZ Mitarbeitern vermieden werden.

Mit der Möglichkeit Projekte in Unternehmen zu realisieren wurde im SRZ eine Schnittstelle zwischen Wirtschaft und Schule geschaffen, die von allen Beteilgten gern genutzt wird und aus unserer Sicht auch in anderen Regionen möglich ist.

In diesem Sinne ist die Arbeit an dem Projekt für die Schüler „*dadurch gekennzeichnet, dass er* [sie] *den Lernenden Raum bietet für intellektuelle Abenteuer und bereichernde soziale Erfahrung, für praktisches Handeln und konkrete Erkenntnisse, kurz: für offene, aber beantwortbare Fragen und für das Leben, so wie es ist.*" [BIL08]

Diese Forderung für einen guten Informatikunterricht aus den Bildungsstandards Informatik wird durch das Konzept der Firmenprojekte am SRZ in hohem Maße erfüllt.

Literaturverzeichnis

[AK13] Adam, M; Körner, N: Entwicklung von Widgets für das Dashboard der T-Systems Multimedia Solutions GmbH – Dokumentation; unveröffentlichtes Material; Dresden 2013

[BBS11] Beckert, J; Beckert, S; Störch, M: Optimierung TeamCal – Dokumentation, unveröffentlichtes Material; Dresden 2011

[BIL08] Bildungsstandards Informatik für die Sekundarstufe I; Empfehlungen der Gesellschaft für Informatik e. V.; Beilage zu LOG IN, 28. Jg. (2008), Heft Nr. 150/151

[HEI05] Heinicke, St.: Das Schülerrechenzentrum Dresden wurde 20; in: LOGIN Nr. 133, LOG IN Verlag Berlin 2005

[HTU03] Heinicke, St.: Timmermann, B.; Unger, M: Konzepte für die Begabtenförderung auf dem Gebiet der Informatik und ihre Umsetzung am Schülerrechenzentrum Dresden in: P. Hubwieser (Hrsg.) Informatische Fachkonzepte im Unterricht, Lecture Notes in Informatics Vol. P 32, Bonn, 2003

[HU07] Heinicke, St.; Unger, M: Kompetenzen im Informatikunterricht und ihre Umsetzung am Schülerrechenzentrum Dresden; in: P. Stechert (Hrsg.) Informatische Bildung in der Wissensgeselschaft, Universitätsverlag Siegen, 2007

[SRZ12] WWW-Seiten des Schülerrechenzentrums Dresden http://www.srz.tu-dresden.de

Fräulein X: Design von außerschulischen Lernräumen zur Förderung der Selbstwirksamkeit im Bereich der Angewandten Informatik

Katharina Weiß, Claudia Hahn, Michael Herczeg

Institut für Multimediale und Interaktive Systeme (IMIS)
Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck
weiss@imis.uni-luebeck.de; hahn@imis.uni-luebeck.de;
herczeg@imis.uni-luebeck.de

Abstract: Dieser Beitrag präsentiert ein Konzept zur Förderung des Interesses an der Informatik, wobei Mädchen eine besondere Zielgruppe darstellen. Vor dem Hintergrund erfolgreicher Veranstaltungen, die vor allem von Schülern nachgefragt wurden, zeichnete sich der Bedarf nach einem Workshop ab, der Mädchen anspricht. Dabei wurden zwei Ziele verfolgt: Erstens sollte eine Lernumgebung geschaffen werden, welche die Vermittlung und Konstruktion grundlegenden Wissens der Informatik unterstützt. Zweitens sollte ein Workshopformat geschaffen werden, das die männlich dominierte Informatik auch für Schülerinnen attraktiv macht und sie zu einer Auseinandersetzung mit dem Thema ermutigt. In eine Rahmengeschichte mit zwischenmenschlicher Thematik wurden Themen wie Verschlüsselung, Algorithmen und formale Logik eingebettet und bearbeitet. Die Workshops wurden mit Hilfe von Fragebögen evaluiert. Dabei konnte über den Zeitraum des Workshops ein Anstieg der Informatik-bezogenen Selbstwirksamkeit beobachtet werden. Die Ergebnisse der Befragung deuten auf eine erfolgreiche Umsetzung des Workshop-Konzepts hin.

1 Hintergrund

Die dynamische Entwicklung im Bereich der digitalen Technologien und der Neuen Medien, die ein wesentlicher und selbstverständlicher Bestandteil des privaten und beruflichen Alltags von Jugendlichen sind und verstärkt sein werden, macht eine Auseinandersetzung mit grundlegenden Funktionalitäten und Funktionsweisen dieser Technologien immer wichtiger. Für Jugendliche gilt es Kompetenzen aufbauen, die sie befähigen, auf fundierten Kenntnissen basierend, Nutzungsentscheidungen zu treffen und neue digitale Technologien zielgerichtet und adäquat einzusetzen. Eine praxisnahe Auseinandersetzung mit Fragestellungen und Herausforderungen der Informatik bildet eine Grundlage für eine kritische Reflektion neuer Entwicklungen auch im Hinblick auf ihre gesellschaftlichen und kulturellen Auswirkungen und lässt Jugendliche das gestalterische Potential neuer digitaler Technologien entdecken.

Hierfür bedarf es Angebote für Jugendliche, innerhalb derer zum einen bewährte pädagogische Konzepte und notwendiges informatisches Fachwissen zusammengeführt werden, zum anderen Raum und Möglichkeiten zur Umsetzung neuer kreativer Ideen geschaffen werden. Im Rahmen des Forschungstransferprojektes LIaS (Lübecker Informatik an Schulen) der Schülerakademie der Universität zu Lübeck wird Jugendlichen eine zeitgemäße Auseinandersetzung mit dem Fachgebiet der Informatik und Querschnittsbereichen ermöglicht, die an die Lebenswelt der Zielgruppe anknüpft.

1.1 Die Schülerakademie und ihre Initiativen

Die LIaS-Initiative der Schülerakademie der Universität zu Lübeck versteht sich als langfristige Schnittstelle zwischen den universitären Studienangeboten, den Lübecker Schulen sowie außerschulischen Lernkontexten für Schülerinnen und Schüler. Ziel ist die Schaffung und Gestaltung von modernen Lehr- und Lernräume für Schülerinnen und Schüler sowie Pädagogen, innerhalb derer Kompetenzen im Umgang mit Neuen Medien und modernen Informations- und Kommunikationstechnologien erworben werden können. Aktuelle Entwicklungen im Bereich Informatik und Interaktiver Medien werden unmittelbar in den Schulalltag oder in außerschulische Lernräume übersetzt und ergänzen bewährte Szenarien der Wissenskonstruktion und des Kompetenzerwerbs durch neue Impulse aus der Forschung. LIaS unterstützt vor allem Jugendliche darin, Fertigkeiten zu entwickeln, die sie befähigen, über eine reine Anwendung digitaler Technologien hinaus, eine aktive Rolle bei der Ausgestaltung und Entwicklung dieser Neuen Medien einzunehmen und diese ihren eigenen Ansprüchen entsprechend reflektiert und kompetent im Alltag nutzen zu können.

Das Informatik-Summer-Camp unserer Schülerakademie ist eine einwöchige Veranstaltung mit verschiedenen Workshops aus unterschiedlichen Themenbereichen der Informatik an. In den letzten fünf Jahren lagen die Teilnehmerzahlen zwischen 40 und 60 Jugendlichen pro Jahr, wobei die Nachfrage inzwischen deutlich das Angebot an zur Verfügung stehenden Plätzen übersteigt. Schülerinnen und Schüler führen unter intensiver Anleitung ein spannendes Projekt aus einem der Bereiche Multimedia, Robotik, Softwaretechnik oder Telematik durch, das von Studierenden der Informatik konzipiert und betreut wird.

1.2 Warum ein Workshop-Angebot speziell für Mädchen?

Im Rahmen der LIaS-Initiative wurden eine Vielfalt an Angeboten für Jugendliche zwischen 14 und 18 Jahren konzipiert und erfolgreich durchgeführt. Trotz einer durchweg hohen Nachfrage bei Jugendlichen aus dieser Altersgruppe blieb die Anzahl der Teilnehmerinnen allerdings immer auf einem niedrigen Stand. Dieses Bild zeigt sich für die Informatik auch in den Ausbildungsberufen oder im akademischen Bereich.

An den Hochschulen in Deutschland wurden 2011 48.423 Studienanfänger im Fach Informatik verzeichnet. Das ist ein Anstieg um fast ein Fünftel (17,8 %) oder mehr als 7.000 verglichen mit dem Jahr 2010. Im akademischen Bereich verzeichneten die Universitäten im Fach Informatik im Jahr 2011einen besonders kräftigen Zuwachs um fast

ein Drittel (29,9 %) auf 25.756 Erstsemester. An den Fachhochulen immatrikulierten sich mit 22.667 Studenten 6,4 % mehr als noch 2010. Allerdings ist die Abbruchquote mit fast 50 % an den Hochschulen weiterhin sehr hoch. Sowohl bei den Ausbildungsberufen als auch an den Hochschulen sind Frauen allerdings weiterhin auffällig in der Minderzahl. Von den Studierenden im ersten Semester ist nur fast jede Fünfte (19,9 %) weiblich. Bei den Auszubildenden im ersten bis dritten Jahr findet sich nur auf knapp jeder zwölften Lehrstelle (8,4 %) eine Frau.[1] Von den insgesamt 19.046 Absolventinnen und Absolventen im Studienbereich Informatik beträgt der Frauenanteil bei den universitären Abschlüssen (Universitätsdiplom oder gleichwertiger Abschluss) 12,2 %, bei den Fachhochschulabschlüssen 15,1 % und bei den Lehramtsabschlüssen 37,1 %.

Eine Vielzahl an Studien setzt sich mit möglichen Ursachen für die niedrige Repräsentanz von Frauen und Mädchen in MINT-Bereichen auseinander und zahlreiche Programme zur Frühförderung von Mädchen und einer stärkere Vernetzung von Frauen in MINT-Berufen wurden in den vergangenen Jahren initiiert. Leitgedanke bei der Konzeption und der Umsetzung des in diesem Beitrag vorgestellten Workshop-Konzepts *Fräulein X* war, dass Mädchen nicht per se weniger Interesse an dem Themengebiet der Informatik haben oder sich von „zu viel Technik“ abschrecken lassen. Der aktuellen JIM-Studie [MFS12] zum Medienumgang 12-bis 19-Jähriger in Deutschland zur Folge besteht die Grundausstattung der Haushalte, in denen 12-bis 19-Jährige leben, aus Computer/Laptop, Handy, Fernseher und Internetzugang. Mediengeräte sind allgegenwärtig und werden für Information, Unterhaltung und Kommunikation genutzt. Hierbei gibt es kaum Unterschiede zwischen den Nutzungshäufigkeiten und –gewohnheiten von Mädchen und Jungen. Bei der Frage nach technischen Kompetenzen als wichtigen Teil der Medienkompetenz von Jugendlichen, die den zielgerichteten und adäquaten Umgang mit Medien gewährleisten, haben Jungen allerdings in nahezu allen abgefragten Kompetenzen mehr praktische Erfahrungen als Mädchen.

Oftmals fehlt ein zielgruppengerechtes motivierendes Design der Angebote, das Mädchen einen Zugang zur praxisnahen Auseinandersetzung mit grundlagenorientierten, aber zeitgemäßen informatischen Inhalten schaffen. Das Workshop-Konzept *Fräulein X* hat das Ziel einen solchen Zugang zu gestalten und die Informatik-bezogene Selbstwirksamkeit der Teilnehmerinnen durch ein ansprechendes Workshop-Design zu stärken.

2 Workshop-Konzept – *Fräulein X*

Das Workshop-Konzept *Fräulein X* bietet Mädchen einen angemessenen Zugang zum Themenfeld Informatik, indem der Lebensweltbezug der vermittelten informatischen Inhalte unmittelbar herausgestellt wird. Mädchen soll ein adäquater Zugang zur Informatik ermöglicht werden, indem sie lernen, diese ihren Bedürfnissen entsprechend wahrzunehmen und ihren Interessen entsprechend einzusetzen.

1 Zur Datenquelle: Grundlage der Angaben sind Auswertungen des BITKOM auf Basis von Daten des Statistischen Bundesamtes sowie der Industrie- und Handelskammern.

2.1 Thema und Rahmengeschichte

Bei der Gestaltung des Fräulein X- Workshops wurden folgende Prinzipien verfolgt: Der Workshop ist inhaltlich anspruchsvoll gestaltet, die Inhalte werden ansprechend präsentiert und in einen lebensweltlichen Kontext gesetzt. Im Rahmen eines Spieles, das die Mädchen durch die Woche begleitet, begegnen die Teilnehmerinnen Aufgaben und Problemstellungen zu Themen wie Aussagenlogik, Verschlüsselungstechniken und Grundlagen des Programmierens am Bespiel von Python (Abb. 1). Eingebettet in eine Rahmengeschichte hilft die Lösung dieser Aufgaben der Akteurin der Geschichte weiter ihr Endziel zu erreichen. Um auf das Workshop Konzept *Fräulein X* im Rahmen des Informatik-Summer-Camps aufmerksam zu machen, wurde bereits im Vorfeld auf eine für Mädchen ansprechende Kommunikation des Angebotes geachtet. Im ersten Jahr ging es dabei vor allem um das Thema „Raffiniert kombinieren mit Informatik“, um in einem narrativen Kontext Wege aufzuspüren, geheime Botschaften zu entschlüsseln und Lügner zu enttarnen. Im darauffolgenden Jahr wurde die Protagonistin *Fräulein X* wieder aufgegriffen, die nun in der virtuellen Welt angekommen ist. Hier ging es dann um Fragestellungen zum Themenkomplex „Sicherheit im Netz“: Wie finden Informationen ihren Weg durch das Internet? Wo lauern Gefahren und was kann Fräulein X tun, um ihnen zu entgehen?

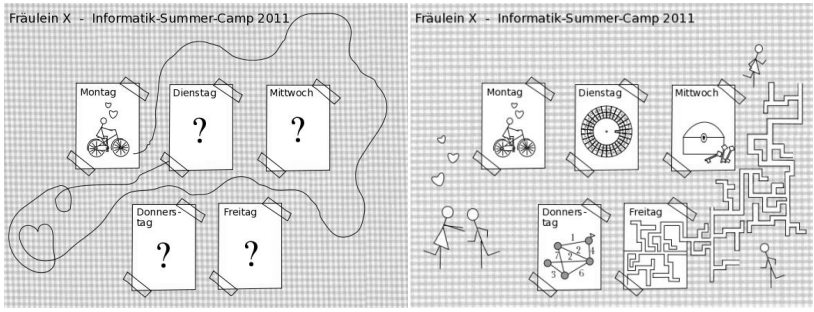


Abbildung 1: Visualisierung des Workshops

2.2 Pädagogische Anforderungen und Struktur des Workshops

Der modulare Aufbau des Workshops unterstützte sowohl die Projektarbeit im Team, ermöglichte den Teilnehmerinnen zudem individuelle Lernpfade im Rahmen des Angebotes selbstbestimmt festzulegen. Jeder Tag hatte einen anderen thematischen Schwerpunkt, der sich auch im Rahmen des Spiels widerspiegelte. Moderierte Phasen wechselten sich mit Phasen ab, innerhalb derer das selbständigen Arbeiten im Team gefordert und gefördert wurde. Es wurde darauf geachtet, den Teilnehmerinnen ausreichend ansprechendes Informationsmaterial zur Verfügung zu stellen, Aufgabenstellung möglichst vielfältig zu gestalten und unterschiedliche Problemlösungsstrategien zu diskutieren und zu unterstützen. Am Ende eines jeden Tages wurde ausreichend Zeit für eine Reflektion und Festigung des erworbenen Wissens und des individuellen Lernfortschritts vorgese-

hen und Teilergebnisse im Plenum vorgestellt und diskutiert. Teil dieses Prozesses war die Arbeit mit dem in unserer Forschungsgruppe entwickelten System *hypervid* [IW12] einem interaktiven System zur browserbasierten Erstellung, Bearbeitung, Verwaltung und Wiedergabe von Hypervideos. Die Teilnehmerinnen konnten begleitend die erworbenen Kompetenzen in kleinen Videofragmenten in einem gestalterisch, kreativen Prozess selbstbestimmt dokumentieren und komplexe Zusammenhänge in einer Hyperstruktur erfassen. Im Rahmen einer Abschlusspräsentation konnte das gemeinsam erstellte Hypervideo die bearbeiteten Themen und einzelne Projektarbeiten in einer komplexen Struktur abbilden und zur weiteren Reflektion gemeinsam mit anderen anregen.

2.3 Lernziele

Ziel des Workshop-Konzeptes war es, eine anspruchsvolle praxisnahe Auseinandersetzung mit informatischen Inhalten zu ermöglichen. Die Teilnehmerinnen wurden spielerisch an die Themen herangeführt. Beim Themengebiet der Aussagenlogik ging es darum, die Auswertung von Vergleichen innerhalb von Programmen verstehen und anwenden zu können. Die Teilnehmerinnen haben im Rahmen des Workshops verschiedene Verschlüsselungstechniken (Cäsar- Chiffre, Vigenére, Diffie-Hellman, Steganographie) kennengerlernt und praktisch erprobt. Es wurden Programmiergrundlagen mit Hilfe von Python erarbeitet, wobei grundlegende Datentypen, Variablen, Funktionen, Schleifen und bedingte Ausdrücke erläutert wurden. Vor dem Hintergrund grundlegenden Informatikwissens wurden auch Themen wie Datenschutz und Soziale Netzwerke sowie die Funktionsweise von Cookies aufgegriffen. Durch die gestalterisch kreative Aufarbeitung des erworbenen Wissens mit Hilfe des interaktiven Hypervideosystems *hypervid* konnten die Teilnehmerinnen ein Grundverständnis von dem Aufbau von Hyperstrukturen, Interaktion und erweiterten Formen der Wissensgenerierung aufbauen.

Ein weiteres Ziel des Workshops war zudem bei den Teilnehmerinnen das Interesse für Informatik zu stärken und eigene Kompetenzen in diesem Bereich zu entdecken. Hier konnte über den Zeitraum des *Fräulein X*-Workshops ein Anstieg der Informatikbezogenen Selbstwirksamkeit beobachtet werden. Die Selbstwirksamkeit ist das zentrale Konstrukt der sozial-kognitiven Theorie Banduras [Ba97]. Dabei handelt es sich um eine subjektive Beurteilung des eigenen Leistungsvermögens bezüglich des Ausführens einer spezifischen Aufgabe oder eines Verhaltens. Sie muss von den Ergebniserwartungen unterschieden werden, also den erwarteten Konsequenzen, die ein Verhalten nach sich zieht. Die Selbstwirksamkeit bezieht sich auf die Überzeugung, ob man etwas tun kann während die Ergebniserwartungen erklären, ob man etwas aufgrund der Konsequenzen auch tun möchte. Die Selbstwirksamkeit ist ein Konstrukt, das gut untersucht und in vielen Bereichen der Psychologie und Pädagogik als relevant angesehen wird, da sie in der Lage ist, Verhalten vorherzusagen. Sie erhöht die Motivation und ist ein Einflussfaktor dafür, welche Aktivitäten eine Person auswählt und durchführt. Personen mit hoher Selbstwirksamkeit weisen ein höheres Durchhaltevermögen auf, setzen sich höhere Ziele [WB89] und erzielen bessere Leistungen, selbst wenn Fähigkeiten und das vorhergegangene Leistungsniveau kontrolliert werden [BL03]. Es handelt es sich um ein dynamisches Konstrukt, welches von vier Informationsarten beeinflusst wird. Die wichtigste Informationsquelle ist das erfolgreiche Ausführen von Handlungen, wobei Erfolge die

Selbstwirksamkeit erhöhen und Misserfolge zu einer Abnahme der Selbstwirksamkeit führen. Eine weitere Informationsquelle sind vikarielle Erfahrungen, etwa bei der Vermittlung von Kompetenzen und soziale Einflussnahmen. Es ist zu erwarten, dass diese Einflussfaktoren bei sorgfältig gestalteten Trainings auftreten, weshalb wir vermuteten, dass die Teilnahme am zielgruppenorientierten Workshop mit einem Anstieg der Selbstwirksamkeit verbunden ist. Genau diese Vermutung wurde in der Evaluation überprüft.

3 Evaluation

In beiden Jahren der Durchführung wurden die Workshops durch die Teilnehmerinnen und Teilnehmer evaluiert. Nachfolgend werden das methodische Vorgehen und die Ergebnisse berichtet. Ziel der Evaluation war es die wahrgenommene Qualität der Veranstaltungen durch die Teilnehmer sowie die Informatik-bezogene Selbstwirksamkeit der Teilnehmer zu erfassen.

3.1 Methodik

Im ersten Jahr (2011) besuchten 15 Mädchen den Workshop, im zweiten Jahr (2012) nahmen ebenfalls 15 Mädchen teil. Im ersten Jahr waren die Teilnehmerinnen im Durchschnitt 14,6 Jahre (*SD*=0,89) alt und im zweiten Jahr 15,1 Jahre (*SD*=0,67). Drei Mädchen haben in beiden Jahren am Workshop teilgenommen. Alle Daten wurden mittels Fragebogen erhoben. Der pre-Messung fand am ersten Tag vor Beginn der Workshops statt, die post-Messung am letzen Tag nach dem Ende der Workshops.

Nach Abschluss der Veranstaltung wurden folgende Aspekte der Workshops von den Teilnehmerinnen bewertet: die Aufgabenschwierigkeit, die Arbeitsgeschwindigkeit, die Erklärungen zu den Aufgaben, die Inhalte des Workshops sowie Spaß und Betreuung während des Workshops. Die Aspekte wurden auf 7-stufigen Likert-Skalen eingeschätzt, deren Polen Adjektive wie *zu niedrig* und *zu hoch* bzw. Beschreibungen wie *keinen Spaß* und *viel Spaß* zugeordnet wurden. Bei diesen Ratings handelt es sich um eine subjektive Bewertung der Teilnehmer, da sie nach ihrer persönlichen Meinung gefragt werden.

Die Selbstwirksamkeit wurde mit einer adaptierten Version der BSW-Skala [ASA00] erfasst. Diese Skala erfasst generelle berufliche Selbstwirksamkeitserwartung und wurde von den Autoren an das Themengebiet Informatik angepasst. Die BSW-Skala wurde anhand verschiedener Außenkriterien validiert und besteht aus sechs Items. Jeweils die Hälfte der Items erfasst motivationale bzw. Aspekte der Fähigkeit. Die Items werden ursprünglich auf einer 5-stufigen Likert-Skala von *stimmt nicht* bis *stimmt genau* beantwortet, die hier auf eine 7-stufige Skala ausgedehnt wurde. Die Hälfte der Items wird revers kodiert. Für die adaptierte Skala lagen die Reliabilitäten (Cronbachs Alphas) für die untersuchte Stichprobe im ersten Jahr bei .50 und .60 für die pre- und post-Messung, im zweiten Jahr jeweils bei .86 und .93, sodass sie als gut und in einem Fall zumindest als akzeptabel anzusehen sind.

3.2 Ergebnisse

Die wesentlichen Aspekte des Workshops wurden durch die Teilnehmerinnen bewertet um die Qualität des Workshopkonzeptes und der Durchführung zu beurteilen. Mit t-Tests für eine Stichprobe wurde untersucht, ob die Mittelwerte signifikant von der Skalenmitte abweichen. Dabei ist zu beachten, dass für einige Items Werte wünschenswert sind, die nicht signifikant von der Skalenmitte verschieden sind, wohingegen für andere Items möglichst hohe Mittelwerte angestrebt werden.

Folgende Ergebnisse wurden für den Fräulein X Workshop ermittelt: In beiden Jahren waren die Urteile zur Aufgabenschwierigkeit und Arbeitsgeschwindigkeit nicht signifikant von der Mitte verschieden. Die Erklärungen zu den Aufgaben wurden im ersten Jahr als in gewissem Maße unzureichend beurteilt, jedoch wiederholte sich diese Beobachtung nicht im zweiten Jahr. Die Urteile zu den Workshop-Inhalten, Spaß und Betreuung während der Workshops lagen in beiden Jahren signifikant über der Skalenmitte. Demnach wurden die Inhalte der Workshops als interessant beurteilt, die Teilnehmer berichteten, dass sie Spaß während der Veranstaltung hatten und die Betreuung durch Studierende wurde als gut beurteilt. Mittelwerte und Standardabweichungen sowie Ergebnisse der t-Tests werden in den Tabellen 1 und 2 berichtet. Die Mittelwerte und zugehörigen 95%-Konfidenzintervalle werden in Abbildungen 2 und 3 dargestellt.

	M	*SD*	
Aufgabenschwierigkeit	3,60	0,99	$t(14) = -1{,}57$, *ns*
Arbeitsgeschwindigkeit	3,64	1,22	$t(13) = -1{,}10$, *ns*
Erklärungen zu den Aufgaben	3,73	0,46	$t(14) = -2{,}26$, $p < .05$
Workshopinhalte	6,27	0,80	$t(14) = 10{,}99$, $p < .001$
Spaß	6,47	0,64	$t(14) = 14{,}93$, $p < .001$
Betreuung	6,60	0,63	$t(14) = 15{,}92$, $p < .001$

Tabelle 1: Evaluation des Workshops 2011 und Signifikanztests (*N*=14 bis 15)

	M	*SD*	
Aufgabenschwierigkeit	3,92	0,79	$t(11) = -0{,}36$, *ns*
Arbeitsgeschwindigkeit	3,88	0,31	$t(11) = -1{,}39$, *ns*
Erklärungen zu den Aufgaben	4,00	0,43	$t(11) = 0{,}00$, *ns*
Workshopinhalte	6,08	1,08	$t(11) = 6{,}66$, $p < .001$
Spaß	6,75	0,45	$t(11) = 21{,}06$, $p < .001$
Betreuung	6,83	0,39	$t(11) = 25{,}26$, $p < .001$

Table 2: Evaluation des Workshops 2012 und Signifikanztests (*N*=12)

Es wurde ein positiver Effekt des Workshops auf die Informatik-bezogene Selbstwirksamkeit der Teilnehmerinnen erwartet. Die Mittelwerte der Messungen vor und nach den Workshops wurden jeweils mit t-Test für abhängige Stichproben untersucht. Die Voraussetzungen für diesen Test (Intervalldatenniveau und Normalverteilung) wurden erfüllt. Der Test untersucht Unterschiede zwischen Messwertpaaren, d.h. den Unterschied der Selbstwirksamkeit zwischen der pre- und post-Messung. Vollständige Daten waren im ersten Jahr für 15 Mädchen und für 12 Mädchen im zweiten Jahr vorhanden. Mittelwerte und Standardabweichungen der Informatik-bezogenen Selbstwirksamkeit werden

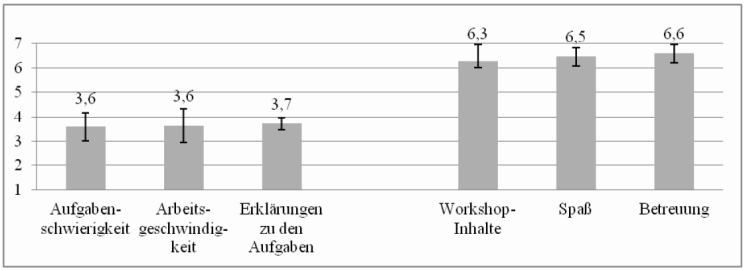


Abbildung 2: Evaluation des Workshops im Jahr 2011, Mittelwerte und 95%- Konfidenzintervalle (N=14 bis 15)

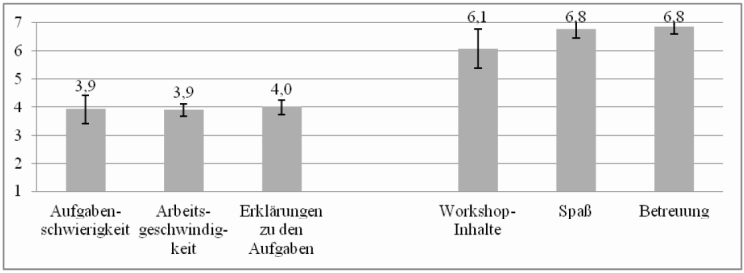


Abbildung 3: Evaluation des Workshops im Jahr 2012, Mittelwerte und 95%- Konfidenzintervalle (*N*=12)

Selbstwirksamkeit und die jeweilige Differenz. In beiden Jahren konnte ein signifikanter Anstieg der Informatik-bezogenen Selbstwirksamkeit der teilnehmenden Mädchen beobachtet werden ($t(14) = -4{,}90$, $p < .001$, $t(11) = -3{,}20$, $p < .01$). Der Test mag darauf hinweisen, dass der Workshop einen positiven Effekt auf die Informatik-bezogenen Selbstwirksamkeit der Teilnehmerinnen hatte. So haben Mädchen nicht nur neue Inhalte aus dem Fachgebiet der Informatik gelernt, sie haben auch die Einschätzung ihres Leistungsvermögens in diesem Bereich dementsprechend in einer positiven Richtung angeglichen. Weiterhin kann der Anstieg der Selbstwirksamkeit in beiden Jahren als Hinweis auf den Erfolg dieses genderspezifischen Workshop-Konzeptes angesehen werden. Natürlich muss hier eingeräumt werden, dass in Verbindung mit der Veränderung der Selbstwirksamkeit eine Kontrollgruppe fehlt.

		M	*SD*
2011	pre Selbstwirksamkeit	3,95	0,62
	post Selbstwirksamkeit	5,12	0,74
2012	pre Selbstwirksamkeit	4,42	1,21
	post Selbstwirksamkeit	5,33	1,25

Tabelle 3: Mittelwerte und Standardabweichungen der Selbstwirksamkeit

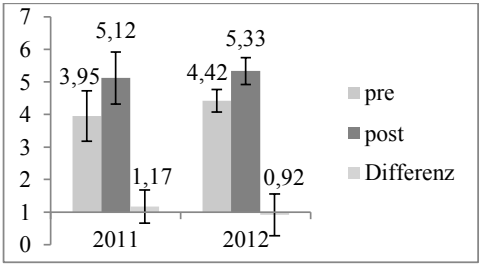


Abbildung 4: Mittelwerte der Selbstwirksamkeit für pre- und post-Messungen, und Differenz zwischen diesen beiden Messungen jeweils mit 95%-Konfidenzintervallen (*N*=15 im Jahr 2011 und *N*=12 im Jahr 2012)

4 Fazit und Ausblick

Das vorgestellte Workshopkonzept Fräulein X verfolgte das Ziel Mädchen zu einer aktiven Auseinandersetzung mit der Informatik zu bewegen, indem ein für sie attraktiver Lernraum geschaffen wurde. Durch eine Verknüpfung der Lebenswelt der Schülerinnen mit informatischen Inhalten konnte ein motivierender Zugang zum Themengebiet der Informatik hergestellt werden. Eine ansprechende Rahmengeschichte, welche die Mädchen durch die Woche führte, erforderte die Lösung herausfordernder logischer Probleme durch die Anwendung informatisch-technischer Kompetenzen. Durch den modularen Aufbau des Workshops war es gewährleistet, dass die Mädchen ausgehend von ihrem individuellen Kenntnisstand und ihrem Lerntempo individuelle Lernpfade verfolgen konnten. Sie konnten zwischen eher kreativen und gestalterischen Aktivitäten oder formal-logischen Aktivitäten, z.B. Programmieren mit Python, wählen. Die Teilnehmerinnen konnten in den Workshop eigene Ideen einbringen und sie in kleinen Projekten im Team realisieren. Sie mussten ihre Aktivitäten eigenständig planen und mit anderen kooperieren. Jeder Tag endete mit einer Reflektion der bearbeiteten Themen und des eigenen Lernfortschritts. Am Ende der Woche wurden die Ergebnisse von den Schülerinnen in einer Präsentation gebündelt und einem interessierten Publikum vorgestellt. Dieser Rahmen diente dazu, dass die Teilnehmerinnen sich ihre neu erworbenen Kompetenzen und das fundierte Wissen nochmals vor Augen führen und dabei eine Expertenrolle einnehmen.

Mit dem Workshop wurde eine Gelegenheit geschaffen, die zu einer aktiven Auseinandersetzung mit dem Themenfeld Informatik anregte und weiteres Interesse am Thema weckte. Da die Selbstwirksamkeit ein wichtiger Prädiktor von Verhalten ist, hat sie auch für die Berufswahl Relevanz. Es kann keine Aussage getroffen werden, welchen Einfluss die Teilnahme am Workshop konkret auf die Berufswahl der Teilnehmerinnen hat. Der beobachtete Anstieg der Informatik-bezogenen Selbstwirksamkeit der Teilnehmerinnen schafft jedoch günstige Bedingungen, eine Laufbahn in der Informatik in Betracht zu ziehen. Im Workshop konnten die Teilnehmerinnen die Rolle der Informatik in ihrem täglichen Leben entdecken, selbst eine aktive Rolle in diesem Themengebiet einnehmen

und das gestalterische Potential digitaler Technologien entdecken. Das Workshop-Konzept wie Fräulein X motiviert Mädchen dazu, sich aus eigenem Antrieb mit einem Thema auseinanderzusetzen, da dieses entsprechend ihrer Interessen gestaltet wurde. Deshalb werden wir das Konzept im Rahmen der LIaS-Initiative weiterentwickeln und die dargestellten Effekte längerfristig untersuchen und weiter verfolgen.

Literaturverzeichnis

[ASA00] Abele, A. E.; Sief, M; Andrä, M.S.: Zur ökonomischen Erfassung beruflicher Selbstwirksamkeitserwartungen – Neukonstruktion einer BSW-Skala. Zeitschrift für Arbeits- und Organisationspsychologie, 44 (3), 145-151, 2000.

[AEF12] Ashcraft, C.; Eger, E.; Friend, M.: Girls in IT: The Facts. National Center for Women & Information Technology (NCWIT), 2012.

[Ba97] Bandura, A.: Self-efficacy: The exercise of control. Freeman, New York, 1997.

[BL03] Bandura, A.; Locke, E. A.: Negative self-efficacy and goal effects revisited. Journal of Applied Psychology, 88 (1), 87-99, 2003.

[Ba10] Barron, B.; Walter, S.; Martin, C. K.; Schatz, C.: Predictors of creative computing participation and profiles of experience in two Silicon Valley middle schools. Computers and Education 54, 178-189, 2010.

[B04] Barron, B.: Learning ecologies for technological fluency: Gender and experience differences. Journal of Educational Computing Research, 31(1), 1-36, 2004.

[Be03] Beyer, S.; Rynes, K.; Perrault, J.; Hay, K.; Haller. S.: Gender differences in computer science students. SIGCSE Bull. 35 (1), 49-53, 2003.

[F04] Faulkner, W.: Strategies of Inclusion: Gender and the Information Society - Final Report, SIGIS, University of Edinburgh, 2004.

[IW12] Ide, M.; Winkler, T.: Hypervideo - Neue ästhetische Projekte in Web 2.0 : Hyperstrukturen in Lernprozessen. In: Lauffer, J.; Röllecke, R. (Hrsg.): Chancen digitaler Medien für Kinder und Jugendliche. Medienpädagogische Konzepte und Perspektiven. Vol. 7, 71-77, kopaed, München, 2012.

[MFS12] Medienpädagogischer Forschungsverbund Südwest: JIM 2012: Jugend, Information, (Multi-)Media. Basisstudie zum Medienumgang 12-bis19-jähriger in Deutschland, Stuttgart, 2012.

[L10] Lasen, M.: Education and career pathways in information communication technology: What are schoolgirls saying?, Computers & Education, 54(4), 1117-1126, 2010.

[SRB02] Schinzel, B., Ruiz Ben, E.: Gendersensitive Gestaltung von Lernmedien und Mediendidaktik: von den Ursachen für ihre Notwendigkeit bis zu konkreten Checklisten, BMBF-Workshop „Gender Mainstreaming in der beruflichen Bildung: Anforderungen an Medienpädagogik und Medienentwicklung“, Berlin, 2002.

[We12] Werner, L.; Denner, J.; O'Connor, L.: Know your students to increase diversity: results of a study of community college women and men in computer science courses. J. Comput. Sci. Coll. 27 (4), 100-111. 2012.

[WB89] Wood, R.; Bandura, A.: Social Cognitive Theory of Organizational Management. Academy of Management Review, 14 (3), 361-384, 1989.

[Qu12] Quaiser-Pohl, C.: Women’s choices in STEM – statistical data and theoretical approaches explaining the gender gap.In: Women’s choices in Europe – Influence of gender on education, occupational career and family development, Waxmann, 183-198, Münster, New York, 2012.

Physical Computing mit „My Interactive Garden“

Mareen Przybylla, Ralf Romeike

Institut für Informatik / Didaktik der Informatik
Universität Potsdam
August-Bebel-Str. 89
14482 Potsdam
[przybyll, romeike]@cs.uni-potsdam.de

Abstract: Informatik ist heutzutage in der Gesellschaft durch interaktive und eingebettete Computersysteme präsenter denn je. Dennoch nehmen viele Schüler Informatik als ein Unterrichtsfach mit abstrakten und realitätsfernen Inhalten wahr. Um dem entgegenzuwirken und Informatik einer breiteren Schülerschaft zugänglich zu machen, wurde das Unterrichtskonzept „My Interactive Garden“ entwickelt, welches eine Gestaltungs-, Programmier- und Lernumgebung beinhaltet. Diese und die vorangegangenen didaktischen Überlegungen werden im Beitrag vorgestellt und ein erprobter Unterrichtsverlauf sowie konkrete Praxiserfahrungen geschildert.

1. Einleitung

Während Informatik für einige Schüler interessant ist und sie die erworbenen Kompetenzen auch privat nutzen, gibt es andere Schüler, die keinen persönlichen Nutzen erkennen und beispielsweise Programmierung als zu kompliziert ansehen [Kn11]. Ein Ziel des Informatikunterrichts sollte es daher sein, die vorhandenen Barrieren zwischen den Schülerlagern abzubauen und allen Schülern sichtbare Erfolge zu ermöglichen. Ein denkbarer Ansatz wird mit der hier beschriebenen Unterrichtsidee verfolgt. Dazu wurde ein Physical-Computing-Baukasten entworfen, mit dem die Schüler nicht nur virtuell grafisch programmieren, sondern darüber hinausgehend auch plastisch künstlerisch-gestalterisch tätig werden können. Physical Computing wird in diesem Zusammenhang daher als das künstlerische Entwickeln interaktiver, physischer Objekte unter Verwendung eines Mikrocontrollers und von Sensoren und Aktoren verstanden. Dies erscheint auch deswegen zeitgemäß, weil den Schülern somit eine Möglichkeit geboten wird, die sie überall umgebenden interaktiven Computersysteme besser zu verstehen und Informatik bewusst in vielen Alltagssituationen wahrzunehmen. Sie erhalten die Gelegenheit, eigene interaktive Systeme zu entwerfen, haben gestalterische Freiheiten und lernen aus konkreten Situationen heraus viele informatische Probleme und Lösungen kennen.

2. Unterrichtsmethode: Informatisches Töpfern

Diese Unterrichtseinheit wurde mit dem Ziel entwickelt, einen motivierenden, kreativitätsfördernden Unterricht im Sinne des Konstruktionismus zu gestalten, um auch weniger computeraffinen Schülern einen attraktiven Zugang zur Informatik anbieten zu kön-

nen. Dazu gehört, dass die Lernenden sich kompetent fühlen, autonom handeln können, den Sinn des Lerngegenstandes erkennen, über domänenspezifisches Wissen verfügen, kreative Prozesse und Arbeitsweisen kennen und beherrschen und greifbare Objekte der realen Welt schaffen (vgl. z. B. [PH91]). In Anlehnung an [RRB08] wurden außerdem folgende Strategien verfolgt: Konzentration auf Themen statt Aufgaben, Kombination von Kunst und informatischem Modellieren, Ermutigung zum Erzählen von Geschichten und Organisation von Ausstellungen statt Wettbewerben. Zusätzlich war auch der Gedanke des *informatischen Töpferns* maßgebend: Die Schüler sollen, analog zu handgemachten Skulpturen aus dem Kunstunterricht, auch selbst hergestellte und programmierte interaktive Objekte aus dem Informatikunterricht mit nach Hause bringen können, welche im konstruktionistischen Sinne untersucht, herumgezeigt und bewundert werden können.

3. Unterrichtsgegenstand: Physical Computing

Physical Computing hat in den letzten Jahren verstärkt an Bedeutung gewonnen. Vor allem Künstler und Designer verwenden programmierbare Hardware, um interaktive physische Systeme zu kreieren, welche die analoge Welt wahrnehmen und auf sie reagieren. Physical Computing kann als kreative Tätigkeit verstanden werden, bei der interaktive Objekte und Installationen unter Zuhilfenahme von Bastel-, Kunst- oder Designmaterial hergestellt werden. Physical Computing überträgt die traditionell virtuellen kreativen Möglichkeiten der Informatik durch Einbeziehung von Aspekten aus Kunst und Design in die reale Welt und kann das Fach vielfältiger und damit attraktiver erscheinen lassen. Im Informatikunterricht wird Physical Computing bisher mit Sensor- und Aktorenboards, mit vorgefertigten Sensoren und Aktoren, sowie mit Mikrocontrollerboards durchgeführt und kann genutzt werden, um eingebettete Systeme zu gestalten.

4. Werkzeug: Baukasten zum Physical Computing

Der speziell auf die Anforderungen zugeschnittene Baukasten zum Unterrichtskonzept „My Interactive Garden“ (MyIG) [PR12] enthält neben Arduino Uno als Mikrocontroller eine aufsteckbare Erweiterung („Shield“), welche genutzt wird, um zahlreiche vorgefertigte Sensoren und Aktoren anschließen und sofort benutzen zu können. Die Schüler erhalten außerdem eine große Vielfalt an Bastel- und Arbeitsmaterialen. Bei der Wahl geeigneter Software wurde vorrangig das Ziel verfolgt, eine intuitive Bedienoberfläche bereitzustellen, welche keine lange Einführung erfordert. Die Entscheidung für einen ersten Unterrichtsversuch fiel somit auf Scratch for Arduino (S4A), eine an Scratch angelehnte und speziell für Arduino aufbereitete Programmierumgebung für Anfänger im Bereich des Physical Computing [Ci12].

5. Unterrichtskonzept

Ziel des Lernens mit MyIG ist die kollaborative Erstellung einer Ausstellung interaktiver Objekte, wie sie in einem futuristischen interaktiven Garten zu finden sein können – ein Rahmen, der vielfältige Projekte ermöglicht und die Kreativität der Schüler anregen soll.

5.1 Angestrebter Kompetenzzuwachs

Das Entwickeln eines Produktes mit MyIG bedeutet, sich mit den Hardwarekomponenten ebenso auseinanderzusetzen wie mit der zu implementierenden Software. Aus den Bildungsstandards Informatik werden nahezu alle Kompetenzbereiche implizit oder explizit angesprochen. Die beschriebene Unterrichtseinheit verfolgt insbesondere das Herausbilden von Kompetenzen in den Inhaltsbereichen Information und Daten, Algorithmen und Informatiksysteme sowie in den Prozessbereichen Modellieren und Implementieren und Begründen und Bewerten.

5.2 Zielgruppe

Als Zielgruppe der Unterrichtsidee werden Schüler der Sekundarstufen I und II mit keinen oder geringen Vorkenntnissen zu grundlegenden algorithmischen Kontrollstrukturen und weiteren Grundbausteinen gesehen. Insbesondere sollen wenig computeraffine Schüler angesprochen und für das Fach begeistert werden. Durch die Vielzahl an möglichen Projekten werden breitgefächerte Interessenbereiche aufgegriffen und mit Informatik verbunden. Schüler, die eine kritische Haltung gegenüber Technik und zum Programmieren haben, können durch den gestaltungsorientierten Zugang einen Nutzen für sich entdecken und mögliche Berührungsängste abbauen.

5.3 Unterrichtsverlauf

Ziel des Unterrichts ist das Entdecken und Erlernen neuer Konzepte im Bereich des informatischen Problemlösens. Insbesondere verwenden die Schüler zur Programmierung mit S4A algorithmische Grundbausteine, testen die eigenen und analysieren und interpretieren die Algorithmen und Programme ihrer Mitschüler. Dabei arbeiten sie explorativ und selbständig. Im Folgenden wird der Ablauf der Unterrichtseinheit zunächst grob skizziert, bevor Erfahrungen der Praxiserprobung geschildert werden. Erwähnte Materialien lassen sich auf der Internetseite http://informatikdidaktik.de/MyIG/ herunterladen. Dort finden sich auch weitere und detailliertere Informationen zur MyIG.

Projektphase	*Details*	*Arbeitsformen, Medien*
Einbettung in Unterrichtsthematik	Als Rahmen eignen sich viele Themengebiete, wie z.B. Informatiksysteme (insbes. Eingebettete Systeme), Algorithmen, Information und Daten u.v.m.	UG[1], LV; Beispielvideos „Magische Blume“, „Geheimnisvolles Lagerfeuer“
Vorstellung Arduino und S4A	Einführung in die Oberfläche von S4A in Verbindung mit Arduino, Besonderheiten gegenüber Scratch hervorheben, Anregungen geben, z.B. wie im angegebenen PDF vorgeschlagen; Aus-	UG, Vorführung von Schülerideen; PDF „Einführung PhysiComp“

[1] Verwendete Abkürzungen: UG (Unterrichtsgespräch), LV (Lehrervortrag), SV (Schülervortrag), EA (Einzelarbeit), PA (Partnerarbeit), GA (Gruppenarbeit), AB (Arbeitsblatt)

Projektphase	*Details*	*Arbeitsformen, Medien*
	führlichkeit an Situation anpassen	
Tinkering	Die Schüler erhalten die Möglichkeit, experimentell alle Bauteile kennenzulernen und auszuprobieren	EA; AB „Tinkering“
Brainstorming	Zum Thema „My Interactive Garden“ Ideen sammeln: je nach Situation in Einzelarbeit, kleinen oder größeren Gruppen, anschließend gemeinsames Schaubild erstellen	GA; AB „Aufgaben“ (auch für weitere Phasen) Aufg. 1
Projektkonzepte entwerfen	Je nach Situation in Einzelarbeit oder Kleingruppen; Schüler finden ein Projekt und erstellen einen groben Plan zur Umsetzung ihrer Idee	EA / PA; AB Aufg. 2 und 3/4 (bei Gruppen)
Konzeptpräsentation	Entworfene Konzepte werden den Mitschülern und der Lehrkraft vorgestellt und anschließend diskutiert, das Konzept wird dann in einen detaillierten (aber vorläufigen) Projektplan überführt	SV, UG (Diskussion)
Arbeitsphase	Schüler arbeiten nach selbst erstellten Zeitplänen; Regelmäßige Evaluationen zu Arbeitsstand, Problemen und Lösungsansätzen, Fragen und Antworten, fachlichen Erkenntnissen um den Zeitplan regelmäßig an sich neu ergebende Bedürfnisse anzupassen - eigenständig (z.B. mit einem Portfolio), in der Gruppe (z.B. gemeinsame Auswertungen zu bestimmten Terminen) oder im gesamten Kurs (z.B. durch Präsentation der Zwischenergebnisse); Je nach Bedarf können Mitschüler und der Lehrer Anregungen geben	EA o. GA; Baukästen, Bastelmaterial, Computer, Papier und Stifte für Skizzen und Planungen
Abschlusspräsentation	Integration der fertigen Installation der Schüler in das Gesamtwerk der Klasse, Projektvorstellung (Geschichte erzählen, Produkt vorführen und Funktionalität erklären), Videodreh, Fragen der Mitschüler und des Lehrers beantworten	SV im Plenum entsprechend AB Aufg. 4/5 (bei Gruppen), Videokameras, Stativ

Tabelle 1: Exemplarischer Unterrichtsverlauf

5.4 Erfahrungseinblick

Das Unterrichtskonzept wurde in einem Schulversuch mit Schülern einer neunten Klasse erprobt. Ihre Vorkenntnisse waren minimal. Die Schüler haben umfangreiche Projekte erstellt und Lernprozesse auf ihrem Niveau durchlaufen, in vielen Situationen ließ sich

ein deutlicher Lernzuwachs beobachten. Bemerkenswert war der Ehrgeiz, den alle Schüler während der Arbeit an ihren Projekten entwickelten. Bei auftauchenden Problemen versuchten sie immer zuerst sich selbst zu helfen und fragten bei Bedarf ihre Mitschüler, ehe sie auf die Lehrkraft zukamen. In den seltensten Fällen wurden Ideen verworfen, meist setzten die Schüler sich so lange mit ihren Probleme auseinander, bis sie zu einer akzeptablen Lösung kamen. Als die Schüler mit ihren Projekten in der ursprünglich geplanten Fassung fertig waren, fingen sie an, diese zu erweitern. Zusätzliche Sensoren und Aktoren wurden verbaut und die Programme angepasst. Schließlich wurden die einzelnen Objekte zu einer großen interaktiven Installation zusammengeführt. So entstand ein interaktiver Garten, in dem beispielsweise die Anzahl der in einem Briefkasten befindlichen Briefe als Botschaft an ein Gartentor gesendet wird, welches über die Blinkgeschwindigkeit einer LED Auskunft darüber gibt. Zur Zusammenführung der einzelnen Objekte mussten sich die Schüler gegenseitig ihre Programme erklären, so dass alle im Bild über die verschiedenen Funktionen waren. Sie haben sich intensiv darüber beraten, welche Interaktionen zwischen den Objekten sinnvoll und realisierbar sind, haben die anstehenden Aufgaben untereinander verteilt und sich gegenseitig bei der Umsetzung unterstützt. Die abschließenden Projektpräsentationen wurden von den Schülern gefilmt. Dieses entstehende Video ist einerseits eine virtuelle Ausstellung, andererseits auch ein Lernprodukt, welches die Schüler mit nach Hause nehmen und ihren Eltern, Verwandten und Freunden vorführen können. Vor Beginn und nach Abschluss des Projekts füllten die Schüler Umfragebögen aus, in denen sie unter anderem zu ihrer Wahrnehmung des Informatikunterrichts befragt wurden. Besonders auffällig war, dass sie den Unterricht mit dem Baukasten durchweg viel positiver bewerteten, als den vorangegangenen Unterricht. Insbesondere Fragen zum kollaborativen Lernen, zum selbstbestimmten Lernen, zur Kreativität und zum Spaß am Fach wurden von allen Schülern positiv bewertet. Auch der Ansatz des informatischen Töpferns wurde in den Umfragen bestätigt. Die Mehrzahl der Schüler bejahte die Frage, ob sie Produkte aus dem Physical-Computing-Projekt bereits Anderen vorgeführt haben. Aus einem an den Umfragebogen angehängten Quiz wurde außerdem deutlich, dass alle Schüler das EVA-Prinzip verinnerlicht hatten, obwohl dies nicht explizit thematisiert wurde. Auch den Unterschied zwischen analogen und digitalen Daten haben die meisten Schüler erfasst, ebenfalls ohne dass dies besprochen wurde. Die Begriffe Informationen und Daten können sie hingegen nur teilweise voneinander abgrenzen.

Literaturverzeichnis

[Ci12] Citilab – Projecte Scratch. Scratch for Arduino. Abgerufen am 10. Juni 2013 von http://seaside.citilab.eu/scratch/arduino

[Kn11] Knobelsdorf, M.: Biographische Lern- und Bildungsprozesse im Handlungskontext der Computernutzung. Dissertation, Berlin, 2011.

[PH91] Papert, S.; Harel, I.: Situating Constructionism. In (Papert, S. & Harel, I. Hrsg.): Constructionism, Ablex Publishing Corporation, Norwood, 1991.

[PR12] Przybylla, M.; Romeike, R.: My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education. In: Kynigos, C et al: Proceedings of Constructionism 2012. Athen. S. 395-404.

[RRB08] Rusk, N.; Resnick, M.; Berg, R.; Pezalla-Granlund, M.: New Pathways into Robotics: Strategies for Broadening Participation. In: Journal of Science Education and Technology , 17 (1), 2008; S. 59-69.

Eine IniK Unterrichtseinheit für die Oberstufe zu Themen der Theoretischen Informatik am Kontext der Mensch-Maschine-Kommunikation mit gesprochener Sprache

Sven Alisch

Gymnasium Lohbrügge
Binnenfeldredder 5
21031 Freie und Hansestadt Hamburg
s.alisch@gyloh.de

Abstract: Die Theoretische Informatik ist in den Rahmenplänen der Bundesländer mit unterschiedlicher Intensität verankert. In Hamburg wird sie innerhalb eines anwendungsorientierten Informatikunterrichts am Beispiel der automatisierten Sprachübersetzung vermittelt. Dieser anwendungsorientierte Ansatz führte in der Vergangenheit zu Problemen, die vom Autor untersucht worden sind. Die Erkenntnisse dieser Untersuchung und ein neues Unterrichtskonzept auf der Basis von „Informatik im Kontext“ (IniK), werden in diesem Praxisbericht vorgestellt.

1 Ausgangslage und Motivation

Theoretische Konzepte der Informatik gehören zu den Grundsätzen und Standards der Informatik in der Schule (vgl. [Bil08]). Eine Analyse der Rahmenpläne der Länder zeigt, dass die Themen:

- Alphabete, Wörter und Zeichen (AWZ)
- Syntax und Semantik (SUS)
- Formale Grammatiken (G)
- Ableitungen (A)
- Rekursionen (R)
- Notationsformen (NF)
- Endliche Automaten (EA)
- Parser (PA)
- Chomsky-Hierarchie (CH)

mit unterschiedlicher Intensität in allen Bundesländern relevant sind (siehe Tabelle 1).

Tabelle 1: Theoretische Konzepte der Informatik in den Rahmenplänen der Länder

Länder	AWZ	SUS	G	A	R	NF	EA	PA	CH
BaWü[BPIa]	K	K	K	K	K	K	W	W	K
Bayern[BPIb]	X	X	X	X	X	X	X	X	W
Berlin[SfB]	X	X	X	X	X	X	X	X	W
Brandenburg[MfB]	X	X	X	X	X	X	X	X	W
Bremen[Löw09]	X	X	X	X	X	X	X	X	X
Hamburg[AJSS09]	W	W	W	W	W	W	W	W	W
Hessen[Kul10]	X	X	X	X	X	X	X	W	X
MV[SfB]	X	X	X	X	X	X	X	X	W
Niedersachsen[Kul04]	X	X	K	X	K	X	K	K	K
NRW[MfSuW99]	X	X	X	K	K	K	X	X	K
RP[MfB10]	X	X	X	K	X	X	X	X	K
Saarland[Sch10]	X	X	X	X	X	X	X	W	X
Sachsen[fBuS11]	W	W	W	W	W	W	K	K	W
Sachsen-Anhalt[EGH$^+$03]	W	W	W	W	W	W	W	K	K
SH[MfB02]	K	K	K	K	K	K	X	K	K
Thüringen[TMfB12]	X	X	X	X	K	X	X	K	K

Legende:

X : Pflicht

W : Kann unterrichtet werden

K : Keine Angaben

PA : Ein Parser wird implementiert

In Hamburg werden die Inhalte der theoretischen Informatik in der Oberstufe innerhalb eines ganzen Semesters im Rahmen eines anwendungsorientierten Informatikunterrichtes am Beispiel der automatisierten Sprachübersetzung vermittelt. Dabei werden am Anfang kleine Wort-zu-Wort Übersetzer mit Hilfe einer funktionalen Programmiersprache (entweder Scheme oder Haskell) geschrieben. Später wird ein Parser funktional modelliert und ein Satz unter Zuhilfenahme einer Grammatik auf Akzeptierbarkeit geprüft.
Auf Grund von persönlichen, nicht zufriedenstellenden Erfahrungen mit diesem anwendungsorientierten Ansatz entwickelte der Autor dieser Arbeit eine neue, am Unterrichtskonzept „Informatik im Kontext“ (IniK) orientierte Unterrichtseinheit zur Mensch-Maschine-Kommunikation mit gesprochener Sprache. Um den pädagogischen Mehrwert dieses neuen Ansatzes nachzuweisen, führte er eine Voruntersuchung durch, in der Hamburger Kolleginnen und Kollegen interviewt wurden.

Die interviewten Kolleginnen und Kollegen mussten folgende Kriterien erfüllen:

- einen ordentlichen Informatikabschluss einer Universität besitzen
- Erfahrungen mit dem Unterrichten von Schülerinnen und Schülern einer Oberstufe
- Erfahrungen mit dem anwendungsorientierten Informatikunterricht in Hamburg

- unterrichtliche Erfahrungen mit dem Anwendungskontext Sprachübersetzung

Die Interviewpartner wurden durch einen Aufruf über eine Mailingliste, in der alle Hamburger Lehrerinnen und Lehrer erreichbar sind, gefunden. Nach diesem Aufruf erklärten sich sechs Kolleginnen und Kollegen bereit, an diesem Experteninterview teilzunehmen. Die Auswertung dieser Interviews ergab, dass Sprachübersetzung ein möglicher Anwendungsbereich ist, der klein und überschaubar ist. Demgegenüber stehen folgende negative Aspekte:

- keine Schülermotivation am Anfang als Einstieg in das Thema
- keine persönlichen Erfolgserlebnisse, weil es kein individuelles Projekt gibt (alle arbeiten am Gleichen)
- keine wirkliche Modellierung und kein Spielen mit formalen Grammatiken und formalen Sprachen möglich
- die Komplexität der funktionalen Programmierung
- eine Entwicklungsumgebung die keine Probleme visualisiert
- Widersprüche mit dem Rahmenplan, der fordert, dass „die Mathematik [nicht] im Zentrum steht“(vgl. [AJSS09]), es aber durch das funktionale Paradigma unweigerlich dazu kommt
- das Implementieren eines Parsers ohne die Konzepte der theoretischen Informatik kennenzulernen und Verbindungen zu anderen Themen herzustellen
- ein Springen zu anderen Themen innerhalb des Curriculums (teilweise zu nicht verwandten Themen) führt zu Verwirrungen bei den Schülerinnen und Schülern
- nicht alle Ziele des Hamburger Rahmenplans in Verbindung mit den theoretischen Inhalten der Informatik werden erreicht und sind überhaupt erreichbar

2 Praxiserfahrungen

2.1 Informatik im Kontext - IniK

Die im Kapitel 1 gefundenen Defizite dieses anwendungsorientierten Unterrichtskonzeptes waren die Basis für einen neuen Ansatz von IniK. Im Gegensatz zum anwendungsorientierten Ansatz ist IniK ein phänomenologischer Ansatz (vgl. [DKW11], S. 102). Ausgangspunkt eines Unterrichts auf Grundlage von IniK ist immer ein Phänomen. Des Weiteren ist die Mehrdimensionalität charakteristisch für eine IniK-Einheit. Eine IniK Unterrichtseinheit kann aus vier Kernphasen (vgl. [KWSS09], S. 9)

1. Begegnungsphase (L1)

2. Neugier- und Planungsphase (L2)
3. Erarbeitungsphase (L3)
4. Vernetzungs- und Vertiefungsphase (L4)

bestehen. In der Begegnungsphase werden die Schülerinnen und Schüler (SuS), mit einem Phänomen konfrontiert, das sie auch in ihrer Lebenswelt kennen. Es werden hier erste Fragestellungen und eine Leitfrage, die während der gesamten Einheit diskutiert wird, entwickelt. In der Neugier- und Planungsphase werden erste Antworten auf die Fragen gesucht und der unterrichtliche Verlauf für die Erarbeitungsphase geplant. In der Erarbeitungsphase eignen sich die SuS die Inhalte der Informatik meistens unter Zuhilfenahme spezieller Software an. Die letzte Phase, auch des öfteren mit Dekontextuierungsphase bezeichnet, stellt Verbindungen zu den Basiskonzepten der Informatik her.

2.2 Lerngruppen und Rahmenbedingungen

Die Unterrichtseinheit wurde bisher dreimal im Unterricht des Autors durchgeführt. Dabei handelte es sich jeweils um einen vierstündigen Informatikkurs auf erhöhtem Niveau (ehemalige Leistungskurse). Diese Informatikkurse sind einem naturwissenschaftlichen Profil zugehörig, d.h. die SuS hatten neben Informatik noch die Fächer Physik und Erdkunde belegt. Die SuS waren i.d.R. sehr an Informatik interessiert. Entsprechend hoch war ihre Motivation.

2.3 Unterrichtsphasen der Lerneinheit

Im folgenden wird die Lerneinheit vorgestellt. Die vollständige Lernverlaufsplanung kann unter [Ali13] nach einer Registrierung eingesehen werden. Die Darstellung orientiert sich an den verschiedenen Phasen aus Kapitel 3.1 (L1-L4).

L1/L2: Einstieg in die Welt der Sprachdialogsysteme

Die SuS sahen hier zunächst einen kleinen Ausschnitt aus einem interaktiven Kinofilm (vgl. [Ali13]). Dieser Einstieg in die Thematik diente der Motivierung und Aktivierung der SuS. Des Weiteren begegneten die SuS durch diesen Film dem Phänomen „Computer und Sprache“. Da dieser Film eine Art Idealzustand der Spracherkennung zeigte, bekamen die SuS anschließend den Dialog eines Sachsen mit der deutschen Bahn ausgehändigt, einer Art Telefoncommedy, die vom Radio PSR ausgestrahlt wurde. Sie hörten den Dialog und bekamen den Auftrag ihn auf typische Missverständnisse zwischen der Kommunikation von Mensch und Maschine zu untersuchen. Im Anschluss daran wurden Fragen der SuS gesammelt, weil sich hier erste Möglichkeiten ergaben, die Mehrdimensionalität des Kontextes Mensch-Maschine-Kommunikation (hier Sprachdialogsystemen im Besonderen) aufzuzeigen. Die Fragen wurden an einem Whiteboard, o.ä. gesammelt und vom

Autor an der Tafel sortiert. In der Vergangenheit konnten die Fragen zu folgenden fünf Themenbereichen zugeordnet werden.

- Wie funktioniert Spracherkennung?
- Wie funktioniert Sprachsynthese (-ausgabe)?
- Programmierung eines Sprachdialogsystems
- Praktischer Einsatz von Sprachdialogsystemen
- Einfluss und Auswirkung von Sprachdialogsystemen auf die Gesellschaft

Aus den Fragen konnten drei Dimensionen erschlossen werden. Das war die technisch-physikalische, die informatische und die soziale Dimension. Die SuS wurden nach dieser Phase in Kleingruppen aufgeteilt und gingen ihren Fragen durch Recherchen im Internet und unterstützenden Materialien nach. Anschließend wurden die Antworten auf diese Fragen durch kleine Schülervorträge realisiert. Beim zweiten Durchgang erwies sich die Ausstellungsmethode als gangbare Alternative, in der die SuS während ihrer Arbeitszeit Plakate (auf Flipcharts) anfertigten. Anschließend begutachteten die SuS während eines Rundganges ihre Plakate untereinander. Diese Methode erwies sich als besonders geeignet, weil die SuS miteinander ins Gespräch kamen und der Autor nur in Detailfragen angesprochen wurde.

L3: Erarbeiten der Grundlagen eines Sprachdialogsystems

Die SuS erarbeiteten sich während dieser Phase die Grundlagen der Programmierung eines Sprachdialogsystems mit Hilfe von *Voice-XML (VXML)*, einem XML-Dialekt. Dabei kamen eine eigens für diese Lerneinheit entwickelte Unterrichtssoftware *inES* zum Einsatz, die sie bei ihrer Arbeit unterstützten. *inES* ist die Abkürzung für ***in****tegrierte* ***E****ntwicklungs-umgebung für* ***S****prachen*, eine Software, die im Jahre 2008 im Rahmen einer Diplomarbeit von Hilger (vgl. [Hil08]) als Prototyp entwickelt wurde. 2009 wurde sie vom Autor dieser Arbeit in Java von Grund auf neu entwickelt und ist unter [Ali13] erhältlich. Sie ermöglicht in dieser Phase eine hohe Schüleraktivität. In Einzel-, bzw. Partnerarbeit erarbeiteten sich die SuS unter Zuhilfenahme eines Arbeitsblattes die Modellierung und Programmierung eines Sprachdialoges. Dabei lernten die SuS die ersten Begriffe Formulare, Menüs, Feldvariablen, Aktionen (Ereignisse) und Übergänge kennen (siehe Abbildung 1).

L3: Grenzen dieser Systeme und Lösungen für deren bedingter Überwindung

Die SuS merkten schnell, dass es sehr aufwendig ist, große Sprachdialoge zu bauen. Selbst die Informationen die ein Mensch in einem Satz formuliert, müssen in einem Sprachdialog mit Hilfe mehrerer Feldvariablen abgefragt werden. Dadurch werden sowohl die Modelle als auch die Quelltexte der Sprachdialoge sehr unübersichtlich. Die SuS entwickelten selbstständig den Wunsch nach einer Veränderung, bzw. nach einer Optimierung ihrer Sprachdialoge. Mit der Blitzlichtmethode konnten die Grenzen dieser Sprachdialoge schnell gesammelt und an einem Medium, wie z.B. einem Whiteboard gesammelt werden.

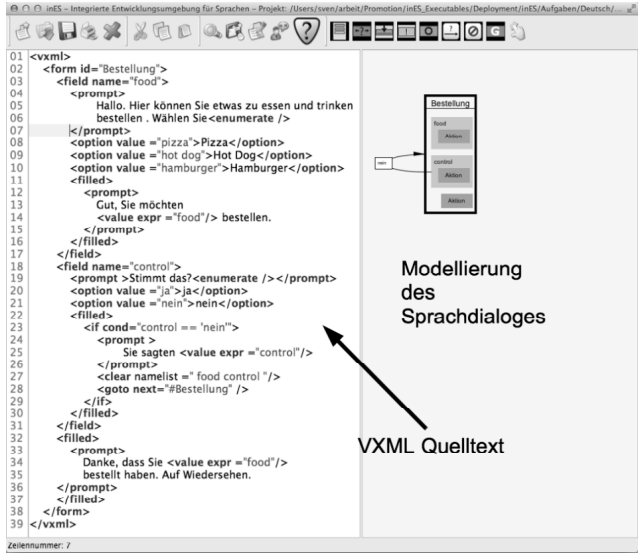


Abbildung 1: Oberfläche von inES

Diese Phase diente der Visualisierung bestehender Grenzen einfacher Sprachdialogsysteme und brachte vor allem zum Vorschein, dass hauptsächlich die Einzelworterkennung, das Fakt-für-Fakt-Abfragen, Floskeln, Dialekte und auch die Länge der Dialoge zum Problem werden. Es ergaben sich zwei weitere Dimensionen (Sprache und Wirtschaftlichkeit) die im Folgenden diskutiert wurden. Direkt daran wurde ein Audioclip von Crealog (vgl. [Ali13]) vorgespielt, in dem aktuelle Möglichkeiten, wie z.B. das Erkennen und das Entnehmen wichtiger Details eines natürlich gesprochenen Satzes, erfahren werden konnten. Diese Phase diente der Ergänzung des Phänomens durch eine wichtige Komponente, die Erkennung und Verarbeitung natürlich gesprochener Sprache. Gemeinsam wurde mit den SuS eine weitere Leitfrage formuliert:

> **Wie ist es möglich, dass Sätze erkannt und die gesuchten Informationen (z.B. Wochentag, Datum) automatisiert herausgefunden werden können?**

Der Autor führte die SuS mit Hilfe einer kleinen Präsentation, die thematisch eng mit dem Audioclip verbunden war, in das Thema ein. Dabei wurde eine erste kleine Grammatik entwickelt, ohne hier bereits den Grammatikbegriff zu definieren, denn die Priorität lag hier vor allem in der für die SuS besseren induktiven Vorgehensweise. Daran schloß sich eine Phase an, in der sich die SuS selbstständig in Einzel- oder Partnerarbeit mit Hilfe von Arbeitsblättern und Videotutorien mit der Erweiterung von inES, dem ***inES Grammatik Editor***, kurz *inGE*, vertraut machten. Die SuS modellierten ohne den Grammatikbegriff explizit zu kennen, erste einfache reguläre Grammatiken mit Hilfe von Syntaxdiagrammen,

die sie in inGE grafisch zusammensetzen konnten. Diese Phase wurde mit einem kleinen Test zur Überprüfung des Verständnisses seitens der SuS, beendet. Erst danach wurde

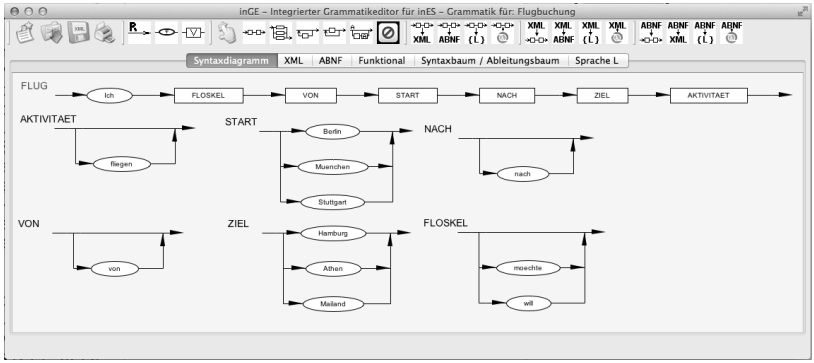


Abbildung 2: Oberfläche von inGE

der Grammatikbegriff durch eine weitere Präsentation eingeführt. Hier musste vom Autor nachgeholfen werden, da die mathematischen Formalismen i.d.R. keinem SuS geläufig waren. Des Weiteren konnten durch diese Phase erste Antworten auf die Leitfrage gefunden und seitens der SuS formuliert werden.

L3: Grammatiken in verschiedenen Grammatiksprachen modellieren

Grammatiken für Sprachdialoge werden i.d.R. mit zwei Grammatiksprachen modelliert. Entweder kommt die XML-artige ***S**peech **R**ecognition **G**rammar* (kurz SRGS) vom W3C-Konsortium oder die ***A**ugmented **B**ackus **N**aur **F**orm* (kurz ABNF), in Deutschland auch als Erweiterte Backus Naur Form bekannt, zum Einsatz. inGE unterstützt neben dem Modellieren von Syntaxdiagrammen auch das Modellieren in den beiden Grammatiksprachen *SRGS*-XML und *ABNF*. Ersteres kam dabei im Unterricht des Autors vermehrt zum Einsatz, denn die SuS waren zu diesem Zeitpunkt mit XML-Dialekten vertraut, weil sie diese Dialekte bereits beim Programmieren der Sprachdialoge kennengelernt hatten. Das Ziel dieser Phase war es, Gemeinsamkeiten und Unterschiede durch die SuS herausfinden zu lassen. Bei den Gemeinsamkeiten erkannten die SuS in allen Grammatiksprachen die grundlegenden Grammatikbegriffe Terminal, Variable (Non-Terminal), Regeln (Produktionen) und Startregel wieder. Die Unterschiede liegen nur in der Notation. Gleichzeitig diente diese Phase der Festigung des zuvor erworbenen Wissens. Es wurde durch die immer wiederkehrende Anwendung dieser Begriffe der Umgang mit formalen Grammatiken trainiert und gefestigt.

L4: Grammatiken helfen dabei, das tägliche digitale Leben zu vereinfachen

Mit dieser Phase wurde die Dekontextuierung und damit die Vernetzungsphase eingeleitet. Es eröffneten sich hier zwei Möglichkeiten. Erstens die Vernetzung mit anderen in

der Informatik gebräuchlichen Anwendungen für Sprache und zweitens die zugrundeliegenden Basiskonzepte. Das Suchen nach Informationen gehörte für die SuS ebenfalls zu ihrer Lebenswelt. Der überwiegende Teil der SuS hatte schon nach Informationen auf der rechnereigenen Festplatte oder bei Google gesucht. I.d.R. ohne es zu Wissen, hatten die SuS mit regulären Ausdrücken gearbeitet. Das sollte für diese Phase zum Aufhänger genutzt werden, um die Suche nach Informationen mit Mitteln der theoretischen Informatik durchzuführen.
Die SuS bearbeiteten während dieser Phase ein Arbeitsblatt und formulierten dabei Suchanfragen mit Hilfe regulärer Ausdrücke. Zunächst arbeiteten sie dabei wieder mit inES, weil die Umgebung für sie vertraut war. Danach benutzten sie eine Konsole (bei Linux) oder die Eingabeaufforderung (bei Windows), um entweder mit den Tools *grep*, bzw. *findstr*, Informationen in Textdateien zu suchen. Hier ergab sich die Möglichkeit auch die rechtliche Dimension des Kontextes aufzumachen, denn die Suchanfragen wurden über E-Mails durchgeführt. Die Frage nach der Identifizierung von Spammails war hier für diese Phase von Bedeutung.

L4: Sprache mit Hilfe von endlichen Automaten verarbeiten (Parsen)

An dieser Stelle sollte auf die Leitfrage zurückgeführt werden, nämlich wie es überhaupt möglich ist, Sprache zu verarbeiten. Das theoretische Konzept eines endlichen Automaten wurde an dieser Stelle erarbeitet. Dabei galt es auch die Parallele zwischen den Sprachdialogsystemen und den endlichen Automaten herzustellen. Damit sollten die SuS erkennen, dass sie, ohne es zu wissen, die ganze Zeit über bereits endliche Automaten modellierten. Ein Sprachdialogsystem kann auch abstrakt als endlicher Automat, der über Zustände (dort die Formulare und Felder) und eine Überführungsfunktion (dort die Übergänge) aufgefasst werden. Nach dem mit einer Präsentation das Modell des endlichen Automaten besprochen wurde, erstellten die SuS endliche Automaten mit Hilfe des Tools JFLAP[1] (vgl. [Uni13]). Damit die Kontinuität gewahrt blieb, sollten die endlichen Automaten nur die Sätze verarbeiten, für die die SuS bereits in den vergangenen Stunden Grammatiken modellierten, damit der Zusammenhang zwischen formalen Sprachen, formalen Grammatiken und endlichen Automaten erkennbar wurde. Die unterschiedlichen Lösungen wurden im Anschluss daran diskutiert. Danach musste dieses Wissen wieder in Übungen gefestigt werden.

L4: Sprachklassen der Chomsky Hierarchie

Bis zu diesem Zeitpunkt bewegten sich die SuS nur in der Welt der regulären Sprachen. Doch die SuS kennen aus ihrer Lebenswelt auch noch andere Sprachen. Dafür wurden die SuS in dieser Phase mit einer kleinen Aufgabe konfrontiert, nämlich der Entwicklung eines „Sprachgesteuerten Mathehausaufgabenlösers“. Im Mittelpunkt stand dabei nicht der Sprachdialog selbst, sondern die Grammatik. Mit inGE mussten die SuS eine Grammatik modellieren, die die Erkennung von mathematischen Termen ermöglichte. Im Unterricht des Autors wurde hierfür keine Hilfe vorgegeben. Zwei Schülergruppen hatten die Lösung nach kurzer Zeit heraus. Es ist nicht das Ziel dieser Phase, dass alle Kleingruppen die

[1] Java Formal Languages and Automata Theorie Program

Lösung selbstständig herausfinden. Vielmehr sollten hier aktive Gespräche unter den SuS entstehen, die auch zu guten Teillösungen führen können. Anschließend sollen mehrere Schülerlösungen diskutiert werden. Dabei wird immer wieder überprüft, in wie fern bestimmte Terme erkennbar oder auch nicht erkennbar sind. Diese Phase war notwendig, damit Modellierungstechniken für Grammatiken gefestigt werden konnten. Danach bekamen die SuS den Auftrag, für diese Sprache einen endlichen Automaten für deren Verarbeitung in der Anwendung JFLAP zu entwickeln. Nach einer Frustphase kamen automatisch immer mehr SuS zu der Erkenntnis, dass für die Verarbeitung dieser Sprache unendlich viele Zustände benötigt werden und das ein Widerspruch zur Definition des endlichen Automaten sei. Hier ergab sich die Möglichkeit, diese *„komische mathematische Sprache“* genauer anzuschauen und den Kellerautomaten zu thematisieren. Der Kellerautomat konnte als Erweiterung des endlichen Automaten durch einen Kellerspeicher (engl. Stack) eingeführt werden. Des Weiteren konnte damit den SuS sehr exemplarisch vor Augen geführt werden, dass diese mathematische Sprache auch über kontextfreie Grammatiken erzeugt werden, aber nicht durch einen endlichen Automaten interpretiert werden kann. Dadurch konnte die Chomsky Hierarchie induktiv aufgebaut werden.

L4: Unsere natürliche Sprache

Die letzte Phase der Unterrichtseinheit beschäftigte sich mit der natürlichen Sprache, dem Deutschen, weil nicht der Eindruck entstehen durfte, unsere Sprache sei komplett durch reguläre Grammatiken modellierbar und nur eine mathematische Sprache würde dem entgegenstehen. Zuerst bekamen die SuS ein Puzzlespiel ausgeteilt. Die Puzzleteile beinhalteten Terminale, Variablen (Non-Terminale), Regeln, eine Startregel, Wörter und die Überführungspfeile. Der Auftrag bestand im Zusammensetzen des Puzzles. Dieses Puzzle wurde von Kleingruppe mit 4 SuS pro Gruppe zusammengesetzt. Die Methode wählte der Autor, weil die SuS über die Puzzleteile diskutieren sollten. Andererseits eignet sich dieser Methodenwechsel dafür, um sich ohne einen Rechner mit formalen Grammatiken zu beschäftigen und aus Diskussionen untereinander herauszufinden, wie das Puzzle zusammengesetzt werden müsste. Die Überprüfung dieser Grammatik erfolgte wieder mit Hilfe von *inGE*. Die SuS sollten ihre Ergebnisse in *inGE* übernehmen und überprüfen, ob denn die Sätze des Arbeitsblattes erkennbar sind. Des Weiteren sollten sie die Grammatik so erweitern, dass der Satz:

> **die Schüler schreiben eine Klausur über Vererbung im Raum 105**

erkennbar wird. Außerdem sollte herausgefunden werden, was denn das Besondere an diesem Satz sei, in dem sie den dazugehörigen Ableitungsbaum in *inGE* erzeugten. Die Schüler erkannten zwei Interpretationsmöglichkeiten. Der Satz kann einmal so:

> Das Thema der Klausur ist die Vererbung und die Klausur wird im Raum 105 geschrieben.

oder so:

> Die Klausur geht über die Vererbung die im Raum 105 stattfand.

interpretiert werden. In beiden Fällen kommen unterschiedliche Ableitungsbäume zustande, die mit *inGE* entsprechend visualisiert wurden (siehe Abbildung 3 und Abbildung 4).

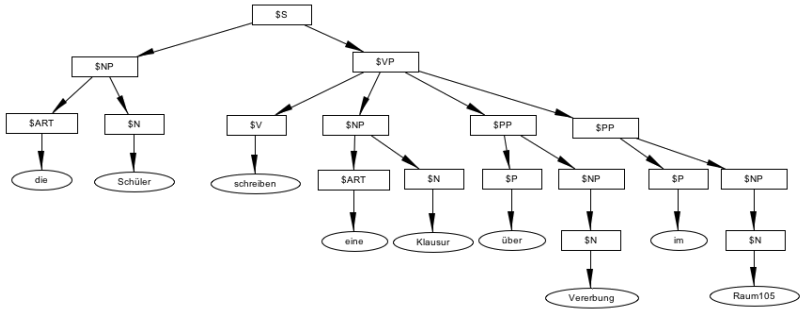


Abbildung 3: Ableitungsbaum der ersten Interpretation - beide Präpositionen sind gleichrangig

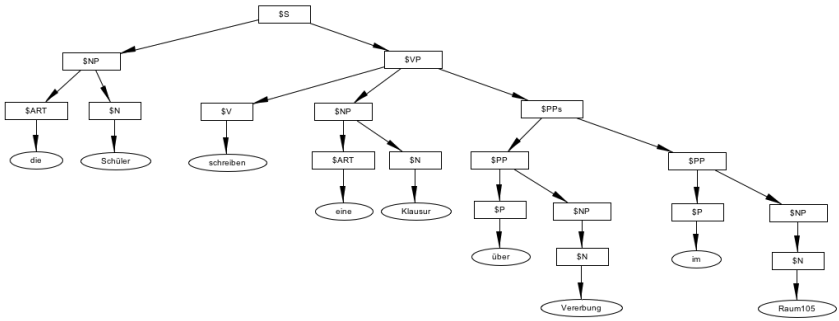


Abbildung 4: Ableitungsbaum der zweiten Interpretation - die zweite Präposition bestimmt die erste näher

Die Schüler kamen zur Erkenntnis, dass kontextfreie Grammatiken nicht mehr für die Modellierung ausreichend sind, sobald die semantische Ebene mit berücksichtigt wird. Zum Beispiel erfordert die Wahl des richtigen Artikels für männliche, weibliche und sächliche Substantive kontextsensitive Grammatiken (siehe Präsentation unter [Ali13]), also Grammatiken, in denen die Variable auf der linken Seite einer Regel nicht mehr allein steht, sondern abhängig von weiteren Variablen wird.

3 Ausblick

Diese Unterrichtseinheit wurde bisher nur vom Autor selbst erprobt. Der nächste Schritt ist die weitere wissenschaftliche Untersuchung dieser IniK-Lerneinheit unter dem Aspekt des pädagogischen Mehrwertes. Dafür müssen als nächstes geeignete Fragebögen und weitere Leitfadeninterviews entwickelt werden. Diese Lerneinheit wird darüber hinaus im kommenden Schuljahr von weiteren Kolleginnen und Kollegen in den Bundesländern Hamburg, Mecklenburg-Vorpommern, Brandenburg und Berlin getestet werden.

Literatur

[AJSS09] Claus Albowski, Michael Janneck, Jan Schöttler und Monika Seiffert. Bildungsplan Gymnasiale Oberstufe. http://www.hamburg.de/contentblob/1475204-/data/informatik-gyo.pdf, 2009.

[Ali13] Sven Alisch. inES - integrierte Entwicklungsumgebung für Sprachen. http://www.ines-ide.de, 07 2013.

[Bil08] Arbeitskreis Bildungsstandards. Grundsätze und Standards für die Informatik in der Schule - Bildungsstandards Informatik für die Sekundarstufe 1, 2008.

[BPIa] Bildungsplan Informatik - Baden-Württemberg. http://www.lehrer.uni-karlsruhe.de/~za171/FS_Informatik/gy_s_inf.pdf.

[BPIb] Bildungsplan Informatik - Bayern. http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/-index.php?StoryID=26193.

[DKW11] Ira Diethelm, Jochen Koubek und Helmut Witten. IniK - Informatik im Kontext. *LOG IN*, 2011.

[EGH+03] Mario Eschrich, Hannes Gutzer, Henry Herper, Hans Lehmann und Jörn Zuber. Rahmenrichtlinien Gymnasium - Sachsen-Anhalt. http://www.bildung-lsa.de/pool/RRL_Lehrplaene/infogym.pdf, 2003.

[fBuS11] Sächsischen Staatsinstitut für Bildung und Schulentwicklung. Lehrplan Gymnasium - Sachsen. http://www.bildung.sachsen.de/apps/lehrplandb-/downloads/lehrplaene/lp_gy_informatik_2011.pdf, 2011.

[Hil08] Sabrina Hilger. *Konzeption und Implementierung einer didaktischen Entwicklungsumgebung für Sprachdialogsysteme*. Dissertation, Universität Hamburg, 2008.

[Kul04] Kultusminister. *Einheitliche Prüfungsanforderungen in der Abiturprüfung*. Luchterhand, 2004.

[Kul10] Hessisches Kultusministerium. Lehrplan Informatik - Hessen. http://verwaltung.hessen.de/irj/HKM_Internet?-cid=48a34f21388de135d056cf8266b8b151, 2010.

[KWSS09] Jochen Koubek, Helmut Witten, Peter Schulze und Carsten Schulte. Informatik im Kontext (IniK). 2009.

[Löw09] Wolfgang Löwer. Bildungsplan Informatik - Bremen. http://www.lis.bremen.de/sixcms/media.php/13-/INF_GyQ_2009.pdf, 2009.

[MfB] Jugend und Sport Land Brandenburg Ministerium für Bildung. Rahmenlehrplan Informatik - Brandenburg. http://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/-rahmenlehrplaene_und_curriculare-_materialien/-gymnasiale_oberstufe/curricula/2011/Informatik-VRLP_GOST_2011_Brandenburg.pdf.

[MfB02] Forschung und Kultur des Landes Schleswig-Holstein Ministerium für Bildung, Wissenschaft. Lehrplan Informatik - Schleswig-Holstein. http://lehrplan.lernnetz.de/index.php?wahl=109, 2002.

[MfB10] Jugend und Kultur Ministerium für Bildung, Wissenschaft. Rahmenplan Informatik - Rheinland-Pfalz. http://informatik.bildung-rp.de/lehrplaene.html, 2010.

[MfSuW99] Wissenschaft und Forschung des Landes Nordrhein-Westphalen Ministerium für Schule und Weiterbildung. Rahmenrichtlinien Gymnasium - NRW. http://www.standardsicherung.schulministerium.nrw.de-/lehrplaene/upload/lehrplaene_download/-gymnasium_os/4725.pdf, 1999.

[Sch10] Robert Scharner. Stand der Informatik in der Schule in Hessen, Rheinland Pfalz und dem Saarland. http://ddi.cs.uni-potsdam.de/Lehre/ddi2/Dossiers2010/Scharner2010.pdf, 2010.

[SfB] Jugend und Sport Berlin Senatsverwaltung für Bildung. Rahmenlehrplan Informatik - Berlin. http://www.berlin.de/imperia/md/content/sen-bildung/unterricht/lehrplaene/sek2_informatik.pdf-?start&ts=1283429474&file=sek2_informatik.pdf.

[TMfB12] Wissenschaft und Kultur Thüringer Ministerium für Bildung. Lehrplan Informatik - Thüringen. https://www.schulportal-thueringen.de/media/detail?tspi=3657, 2012.

[Uni13] Duke University. JFLAP - Java Formal Languages and Automata Theory. http://www.jflap.org, 07 2013.

CrossRoads – die Straßenkreuzung, die es in sich hat

Ein anwendungsorientierter Einstieg in die Informatik in der Sek. I

Antje Bertsch

AnBerIT UG – e-learning & computer-based training
vormals Realschule Mülheim-Kärlich
Alte Wiese 5
56337 Simmern / Ww
antje.bertsch@anberit.de
www.anberit.de

Abstract: Die ProzessDatenVerarbeitung PDV ist seit vielen Jahren ein von Schülern und Lehrern geschätztes Thema, das im Unterricht des Wahlpflichtfachbereiches der Sek I einen festen Platz erobert hat und in der letzten Zeit auch verstärkt im Ganztagsschulangebot aufgegriffen wird. Die in diesem Workshop vorgestellte Unterrichtsreihe ist das Ergebnis mehrjähriger und aktueller Unterrichtserprobung.

1 Konzept der Unterrichtsreihe

Das Ziel dieser für Schüler ab der Klassenstufe 8 geplanten Unterrichtsreihe ist es, die Schüler anhand von Alltagssituationen, die wohlbekannt sind, an das Erkennen der dahinter liegenden Ablaufstrukturen heranzuführen. Die Vermittlung der algorithmischen Grundstrukturen ist das Hauptanliegen. Das Erlernen einer speziellen Programmiersprache ist nicht das Ziel.

Da die Unterrichtsreihe für die Arbeit in der Sek I konzipiert ist, kommt als Software die grafische Programmieroberfläche *ROBO PRO* [1] von *fischertechnik* zum Einsatz, die es erlaubt, die Programmablaufpläne der Vorgänge auf dem Bildschirm darzustellen. Die Überprüfung der erstellten Steuerungsprogramme mit Hilfe des Modells ***CrossRoads*** bietet dem Schüler ein unmittelbares Erfolgserlebnis.

Bei der sonst üblichen Simulation von Alltagssituationen mit den bekannten Modellen der Computing-Baukästen von *fischertechnik* müssen die Modelle zunächst von den Schülern aufgebaut werden, bevor es an das Programmieren geht. Die Zahl der möglichen Fehlerquellen beim Aufbau der Modelle ist groß – doch die Enttäuschung der SchülerInnen, über eine Fehlfunktion beim Simulationstest ist noch viel größer.

[1] Handbuch zu ROBO PRO http://www.die.informatik.uni-siegen.de/tl_files/pdf/lehre/Didaktik1_Uebung/2012-13/01-Bedienungsanleitung.pdf

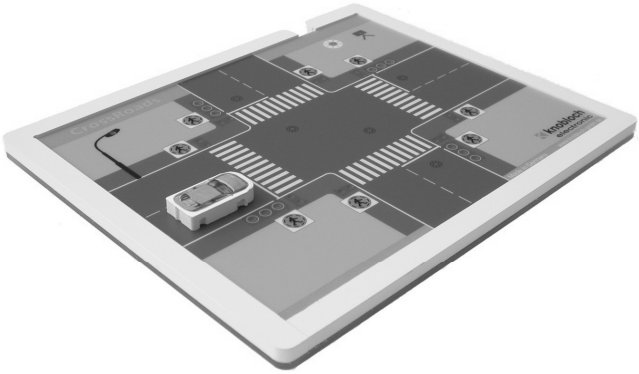
In dem Funktionsmodell ***CrossRoads*** sind alle Aktoren und Sensoren fest eingebaut, so dass die o.g. Fehlerquellen und Enttäuschungen ausgeschlossen sind.

Durch das eingebaute USB-Interface werden keine weiteren Komponenten, wie Netzgerät oder Interface, benötigt: ***plug & learn*** lautet die Devise, denn die Stromversorgung des knapp A4 großen Modells erfolgt direkt über die USB-Schnittstelle des PC's, an den das Modell angeschlossen ist.

Erst wenn die SchülerInnen mit der Programmierumgebung *ROBO PRO* gut vertraut sind und es gelernt haben, die zu steuernden Abläufe gedanklich richtig zu strukturieren und zu vernetzen, folgt in einer zweiten Unterrichtseinheit die Anwendung mit den Modellen von *fischertechnik*.

1.1 Beschreibung des Modells [2]

Das Modell beinhaltet eine Straßenkreuzung mit 4 Verkehrsampeln, 8 Fußgängerampeln und Eingängen zum Abfragen der Fußgängertaster. Abb. 1.1



Magnetsensoren in der Fahrbahn (Kanaldeckel) reagieren auf das *Magnetauto*, so dass Geschwindigkeitsübertretungen und „Rotsünder" mit diesen Hall-Sensoren durch entsprechende Abfragen im Programm erfasst werden können, die dann den roten Blitz der Radarfalle auslösen – ein besonderes Highlight für die SchülerInnen.

Außerdem ist über einen Lichtsensor eine Straßenlaterne steuerbar, so dass auch die Umstellung der gesamten Anlage auf Nachtbetrieb simuliert werden kann.

Das Modell bietet somit ein breites Spektrum an Programmierungsmöglichkeiten mit großem Realitätsbezug.

[2] Knobloch Electronic Produktions- und Vertriebsgesellschaft mbH http://www.knobloch-gmbh.de

1.2 Themen der Unterrichtsreihe

- Einführung in die Arbeitsweise mit *ROBO PRO*
- Interfaceeinstellungen des Modells ***CrossRoads***
- Einfache Ampelschaltungen
- Ampelschaltungen mit Bedarfsabfrage (Unterprogramme)
- Kombination von Ampelzyklen an einer Kreuzung
- Ampeln mit Radarfalle (Geschwindigkeitsüberwachung – Rotsünder)
- Umschaltung auf Nachtbetrieb

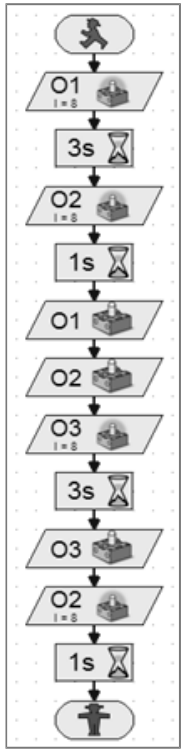


Abb. 2.1

2 Konzept des Workshops

In diesem Workshop wird zunächst eine vollständige und mehrfach erprobte Unterrichtsreihe zur Einführung in das Programmieren mit dem Funktionsmodell ***CrossRoads*** vorgestellt.

Anschließend werden die TeilnehmerInnen Gelegenheit haben, fertige Programmabläufe mit den Modellen an PC-Arbeitsplätzen zu analysieren.

Nach einer kurzen Einführung in die Interfaceeinstellungen mit der Belegung der Ein- und Ausgänge können die TeilnehmerInnen eigene Programme anhand der Aufgaben in dem bereitgestellten Schülerarbeitsheft entwickeln.

Ausführliche Handreichungen und Beispielprogramme für die Umsetzung im Unterricht werden in digitaler Form zum Mitnehmen im Workshop und auch zum späteren Download im Internet bereitgestellt.

Teilnehmerzahl

ca. 20 Personen

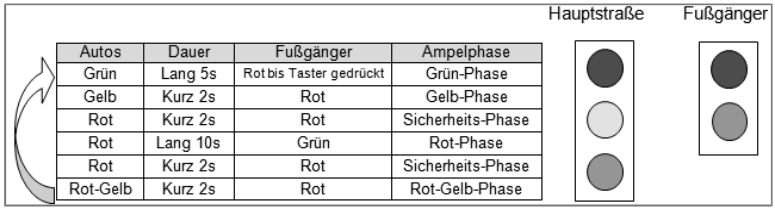
Die Teilnehmerzahl sollte aufgrund der gewünschten praktischen Arbeit nicht zu hoch sein, die Zahl hängt u.a. aber auch von dem bereitgestellten Raum und von der dort vorhandenen Anzahl an PC-Arbeitsplätzen ab.

3 Aufgabenbeispiele

3.1 Ampel mit Taster zur Bedarfsumschaltung für Fußgänger

Es soll nun ein Programm für eine Ampel an einer stark befahrenen Straße erstellt werden, die nur dann auf rot umspringt, wenn ein Fußgänger den Taster drückt, weil er die Straße überqueren will. In der Praxis werden Sicherheits-Phasen eingebaut, warum macht man das?

Tabellarische Beschreibung der Ampelphasen: Abb. 3.1



Autos	Dauer	Fußgänger	Ampelphase
Grün	Lang 5s	Rot bis Taster gedrückt	Grün-Phase
Gelb	Kurz 2s	Rot	Gelb-Phase
Rot	Kurz 2s	Rot	Sicherheits-Phase
Rot	Lang 10s	Grün	Rot-Phase
Rot	Kurz 2s	Rot	Sicherheits-Phase
Rot-Gelb	Kurz 2s	Rot	Rot-Gelb-Phase

Schreibe das Programm zunächst für die Hauptstraße links und den einen Fußgängerüberweg mit den beiden dort eingetragenen Fußgängerampeln. Notiere an den Ampelbildern neben der Tabelle oben die jeweiligen Ausgänge und Eingänge des Interface, die du benutzt.

Normalerweise hat die Hauptstraße grün, damit die Autos fahren,
dann sind die Ausgänge O3 und O6 am Standardinterface IF1 eingeschaltet

die Fußgänger haben rot – Ausgänge EM2 O1 und EM3 O1 ein

ein Fußgänger drückt den Taster z. B. Eingang I1
(oder I2 oder später auch I3 oder I4)

die Hauptstraße wird auf gelb geschaltet
Ausgänge O3 und O6 der Grünphase aus
Ausgänge O2 und O5 der Gelbphase ein

die Hauptstraße wird auf rot geschaltet
Ausgänge O2 und O5 der Gelbphase aus
Ausgänge O1 und O4 der Rotphase ein
die Fußgänger haben zur Sicherheit noch rot

die Ampel der Hauptstraße bleibt 10 Sekunden rot,
gleichzeitig haben die Fußgänger grün
Ausgänge EM2 O2 und EM3 O2 der Fußgänger
für 10 Sekunden ein

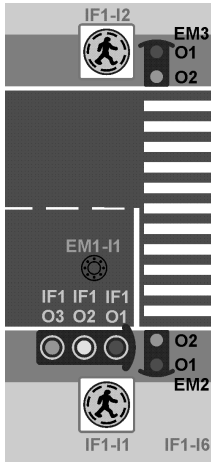


Abb. 3.2

Wechsel der Fußgänger auf rot,
die Hauptstraße bleibt als Sicherheitsphase noch kurz auf rot

dann folgt für die Autos die rot-gelb Phase –
die Ausgänge O2 und O5 werden zugeschaltet

Ausschalten der Ausgänge von rot und gelb
und Rückkehr an den Anfang des Programms

Entwickle das Programm in ROBO Pro, speichere unter *P4_Haupt+FG_Bedarf*
und teste dein Programm.

Jetzt wird deutlich, wie sinnvoll es ist, wiederkehrende Prozesse als Unterprogramme anzulegen und bei Bedarf im Hauptprogramm einzufügen. Nutze die Anleitung auf der vorigen Seite, die erklärt, wie man ein Unterprogramm einfügt, hier z.B. *Fußgänger links* .

Ergänzung

Erweitere das Programm so, dass auch die Taster des Zebrastreifens auf der rechten Seite der Hauptstraße mit abgefragt werden und dass dann auch parallel zur linken Seite die Fußgängerampeln auf der rechten Seite mit auf grün umgeschaltet werden.

Sobald also ein Fußgänger einen der 4 Taster an den Zebrastreifen links (I1 oder I2) oder rechts (I3 oder I4) drückt, sollen die Ampeln der Hauptstraße auf rot wechseln und beide Zebrastreifen sollen für eine kurze Zeit grün bekommen. Speichere dieses Programm unter *P5_Haupt+FG_Bedarf_alle*

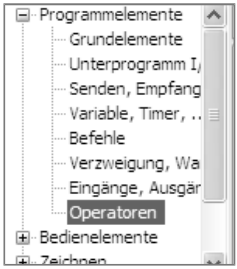
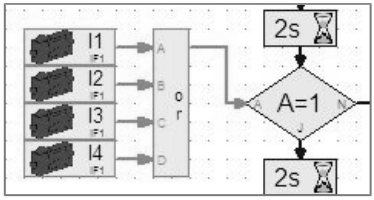


Abb. 3.3

Hier kann eine weitere Funktion von ROBO Pro genutzt werden und zwar die **logischen Operatoren**. Dazu muss im Programm aber der **Level 3** eingestellt werden.

Die Anzahl der Eingänge der Operatoren wird über die rechte Maustaste im Kontextmenü eingestellt

Die Taster findet man im Punkt Eingänge, Ausgänge.



In Verzweigung, Warten befindet sich das Element, das hier anstelle eines Tasters als Verzweigung im Programm eingefügt werden muss.

Abb. 3.4

3.2 Radarfalle für Rotsünder, die bei gelb-rot noch über die Kreuzung fahren

Hier laufen drei Programme zeitgleich und parallel ab:

1. Das Hauptprogramm *PR8_Kreuzung_Zebra*, das schon im Kapitel 5.3 besprochen wurde. Es muss um das hier rechts abgebildete Befehlselement erweitert werden, damit erst nach der Grünphase der Hauptstraße die Überwachung beginnt.

2. Ein Programm, das den Sensor EM1-I1 links in der Hauptstraße abfragt und den Wert 1 an den Eingang C weitergibt, wenn ein Auto den Sensor aktiviert.

3. Ein Programm, das den Radarblitz EM1-O7 auslöst und anschließend den Operator-Eingang C wieder auf 0 setzt.

Mit dem Wechsel der Hauptstraße von grün auf rot durch das Unterprogramm *Wechsel_Haupt_Neben* wird die erste Variable und damit der **Eingang A des Operators** auf 1 gesetzt, um die Überwachung während der Rotphase der Hauptstraße vorzubereiten. Dieser Wert wird für die Abfrage durch den **Operator and** gebraucht, denn nur wenn alle drei Eingänge des **and-Operators** den Wert **1** haben, wird der Radarblitz ausgelöst.

Fährt ein Auto (Magnet) über den Sensor EM1-I1, obwohl die Hauptstraße schon im Übergang auf die Rotphase ist, dann erhält auch der **Operator-Eingang C** den Wert 1.

Der mittlere **Operator-Eingang B** erhält das Signal 1, wenn ein Auto auch über den Sensor EM1-I5 in der Kreuzung fährt. Nur wenn die drei hier beschriebenen Bedingungen für die Operator-Eingänge gleichzeitig zutreffen, wird der Radarblitz ausgelöst.

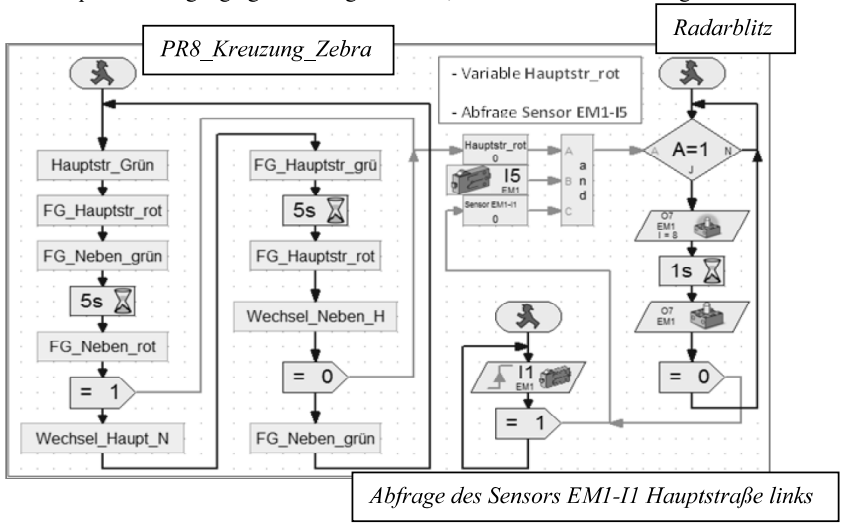


Abb. 3.5

4 Hinweise

Der Workshop stellt eine ideale Ergänzung des von Herrn Carsten Müller (Universität Bayreuth) eingereichten Workshops dar, denn wir benutzen das gleiche Modell auf zwei sehr unterschiedlichen Niveaustufen im Hinblick auf den unterrichtlichen Einsatz und das Alter der Schüler.

Besonders für KollegInnen, die den Vorteil sehen, mit einer Investition (Anschaffung der Modelle CrossRoads) gleich mehrere Ebenen des Informatikunterrichts interessant und anwendungsorientiert an ihren Schulen zu gestalten, dürften beide Workshop-Angebote von Interesse sein.

Darüber hinaus ist ein weiterer Workshop zu diesem Thema vorgesehen, und zwar auf der *Fachtagung der Informatiklehrerinnen und -lehrer in Schleswig-Holstein und Hamburg*, die direkt im Anschluss an die **Infos 2013** am Samstag 28.09.2013 am gleichen Ort stattfinden wird. Darin wird ein Projekt für SchülerInnen der Orientierungsstufe vorgestellt, in welchem die ersten Schritte zur Erstellung einfacher Programme und deren Visualisierung an dem kleinen Modell ***TrafficLights*** gezeigt werden (ebenfalls mit der grafischen Software *ROBO PRO*). Dieses Projekt fand im vergangenen Jahr an der Universität Koblenz im Rahmen der "Kinder-Uni" mit 9-12-jährigen SchülerInnen sehr gute Resonanz.

Für den Einsatz des Modells ***CrossRoads*** müssen auf dem Rechner die Treiber installiert sein, die bei der Installation der Software *ROBO PRO* mitgeliefert werden. Unter win 7 werden die Treiber beim Anschluss der Modelle automatisch erkannt bzw. installiert, da sie die dafür erforderliche Signierung haben. Unter win XP muss beim ersten Anschluss eines Modells auf dem PC die Treiberinstallation durch einen User mit Adminrechten durchgeführt werden, danach läuft alles Weitere auch ohne Adminrechte.

Grunderfahrungen des Informatikunterrichts – ein Beitrag zur Frage der Allgemeinbildung von Informatik

Bernd Bethge

Staatliches Studienseminar für Lehrerausbildung
- Lehramt an Gymnasien -
Gustav-Freytag-Str. 6
99096 Erfurt
b.bethge@asg-erfurt.de

Michael Fothe

Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Ernst-Abbe-Platz 2
07743 Jena
michael.fothe@uni-jena.de

Abstract: Die Konzeption der Grunderfahrungen des Mathematikunterrichts (Heinrich Winter; 1995) lässt sich erfolgreich auf die Informatik übertragen. Das Resultat des Transfers sind Grunderfahrungen des Informatikunterrichts, die in kompakter Form den Beitrag der informatischen Bildung an der Allgemeinbildung beschreiben.

1 Zielstellung und Vorgehen

Dieser Aufsatz befasst sich mit dem Bildungswert des Schulfachs Informatik. Konkret werden *Grunderfahrungen des Informatikunterrichts* entwickelt, die dieses Fach Schülerinnen und Schülern ermöglicht und die andere Unterrichtsfächer nicht oder nur marginal vermitteln können. Ausgangspunkt dafür sind Überlegungen zum Wert informatischer Bildung und die *Grunderfahrungen des Mathematikunterrichts*, die Heinrich Winter im Jahr 1995 formulierte und begründete. [Wi95] Sie umreißen den Anteil des Mathematikunterrichts an der Allgemeinbildung bis zum Abitur und sind inzwischen weitgehend anerkannt. Sie fanden z. B. Eingang in die Bildungsstandards der KMK im Fach Mathematik für die allgemeine Hochschulreife. [Km12, S. 9] Als Voraussetzung für den Transfer aus der Mathematik in die Informatik werden die Intentionen der vorliegenden Grunderfahrungen ermittelt und auf die Informatik angewandt. Die gewonnenen Grunderfahrungen des Informatikunterrichts werden anschließend anhand von vorliegenden Zielbeschreibungen zur informatischen Bildung kritisch reflektiert, um deren Wert und Relevanz einschätzen zu können. Eine erste Version der Grunderfahrungen wurde auf der 11. Tagung der GI-Fachgruppe „Informatik-Bildung in Berlin und Brandenburg“ am 1. März 2012 zur Diskussion gestellt. [Gi12] Seinerzeit hatte noch keine kritische Reflexion in systematischer Form stattgefunden; diese erfolgte erst beim Vorbereiten dieses Aufsatzes und hatte Änderungen an den Grunderfahrungen von 2012 zur Folge.

Im Frühstadium dieses Aufsatzes erfolgte ein Meinungsaustausch zum Thema mit Reinhard Oldenburg, für den die Autoren danken.

2 Zum Wert informatischer Bildung

Dieser Abschnitt beschreibt informatische Bildung von außen, und zwar bildungstheoretisch mit dem Erziehungswissenschaftler Wolfgang Klafki und anhand eines internationalen Konsenspapiers der OECD zu Schlüsselkompetenzen. [Kl07] [Oe05]

Nach Klafki soll ein modernes Bildungskonzept sowohl für die Gegenwart als auch für die Zukunft ein umfassendes Konzept allgemeiner Bildung darstellen. Allgemeinbildung zuallererst als „Bildung für alle" (S. 53). Zweitens Allgemeinbildung als „Bildung im Medium des Allgemeinen", des Verbindlichen, d. h. „als Aneignung der die Menschen gemeinsam angehenden Frage- und Problemstellungen ihrer geschichtlich gewordenen Gegenwart und der sich abzeichnenden Zukunft und als Auseinandersetzung mit diesen gemeinsamen Aufgaben, Problemen, Gefahren. Dabei geht es *auch* um die Auseinandersetzung mit in der Geschichte bereits entwickelten Denkergebnissen und Lösungsversuchen, […] um die [...] sich Bildenden [...] zum Begreifen und zur Gestaltung ihrer historisch vermittelten Gegenwart und ihrer jeweiligen Zukunft in Selbstbestimmung, Mitbestimmung und Solidarität freizusetzen." (S. 53) Solche Aufgaben, Probleme, Gefahren und Möglichkeiten sind heute global, erfordern eine Sichtweise auf den „Welt-Horizont" (S. 54). Allgemeinbildung ist drittens „als Bildung in allen Grunddimensionen menschlicher Interessen und Fähigkeiten" (S. 54) zu verstehen. Die dritte Dimension von Allgemeinbildung scheint eher in Beziehung zur Gestaltung von konkretem Unterricht und Lernarrangements zu stehen, während in der zweiten Dimension der Inhalt allgemeiner Bildung umrissen wird. Klafki nennt explizit als einen Faktorenkomplex, der zur Globalisierung beiträgt: Der von der „technisch-industrielle[n] Entwicklung […] untrennbare Aufbau immer weiterreichender Informations- und Kommunikationssysteme" (S. 80) und hier unter anderem „elektronisch gesteuerte und vernetzte Übermittlungs- und Speicherungssysteme" (S. 80) – mit anderen Worten: Informatiksysteme. Wir brauchen in einem zukunftsorientierten Bildungssystem auf allen Schulstufen und in allen Schulformen eine gestufte, kritische informations- und kommunikationstechnologische Grundbildung als Moment der Allgemeinbildung. Klafki fordert, Kindern und Jugendlichen die Auseinandersetzung mit den Faktorenkomplexen und epochalen Schlüsselproblemen in der Schule zu ermöglichen, um ihnen allgemeine Bildung in den genannten Dimensionen zu erlauben.

Beim Ansatz der OECD steht die Frage „Welche Kompetenzen benötigen wir für ein erfolgreiches Leben und eine gut funktionierende Gesellschaft?" im Mittelpunkt der Betrachtungen. [Oe05, S. 6] Es werden drei Kategorien von Schlüsselkompetenzen angegeben; die Reflexivität wird dabei als „Kern der Schlüsselkompetenzen" herausgestellt (S. 10). Sehen wir uns die Kompetenzkategorien unter dem Blickwinkel der informatischen Bildung genauer an (was im OECD-Text nicht getan wird). In der Kompetenzkategorie 1 „Interaktive Anwendung von Medien und Mitteln (Tools)" heißt es unter anderem: „die Medien, Mittel und Werkzeuge (Tools) für eigene Zwecke einsetzen und anpassen" (S. 12). In dem Zusammenhang werden im OECD-Text auch Computer genannt. Insbesondere das Anpassen von Computern an die eigenen Erfordernisse ist eine originäre Aufgabe im Rahmen der informatischen Bildung. Die Kompetenzkategorie 1 umfasst als erste Kompetenz die Fähigkeit zur interaktiven Anwendung von Sprache, Symbolen und Text; dies wird im OECD-Text mit aktivem mündlichen und schriftlichen Sprachgebrauch und dem vielseitigen Anwenden mathematischer

Fähigkeiten untersetzt. In diesen Bereich gehören jedoch auch relevante Themen, mit denen man im Alltag und im Berufsleben zu tun hat, und die der informatischen Bildung zuzuordnen sind, wie zum Beispiel der Aufbau von E-Mail- und WWW-Adressen, die Struktur von QR-Codes und Regeln zum Erzeugen guter Passwörter. Man denke auch an die praktisch bedeutsamen Dokumentenbeschreibungssprachen und Abfragesprachen (siehe unten im Abschnitt 4). Sprachen besitzen eine zentrale Bedeutung in der Informatik; Sprache ist sogar eine der drei fundamentalen *Master*ideen der Informatik. [SS11, S. 69 f.] Als zweite Kompetenz wird die Fähigkeit zur interaktiven Nutzung von Wissen und Informationen aufgezählt. Und weiter heißt es: „Diese Schlüsselkompetenz setzt eine kritische Reflexion über die Natur der Informationen als solche – ihre technische Infrastruktur sowie ihren sozialen, kulturellen und ideologischen Kontext und ihre Tragweite voraus." (S. 13) Auch dies ist ein wesentlicher Aspekt von informatischer Bildung. Schließlich gehört zur Kompetenzkategorie 1 als dritte Kompetenz auch die Fähigkeit, Technologien interaktiv anzuwenden, wobei betont wird, dass die bloße Anwendung nicht ausreicht, „die Anwender [müssen] sich mit ihrer Beschaffenheit [der Technologie] und ihrem Potenzial auseinandersetzen". Dies sind Aufgaben informatischer Bildung. Die Kompetenzkategorien 2 und 3 beschreiben Selbst- und Sozialkompetenzen im Zusammenhang mit dem Interagieren in heterogenen Gruppen und dem eigenständigen Handeln. Beides erfordert Situationen, in denen in Gruppen agiert, gelernt, kooperiert und arbeitsteilig gehandelt wird, in denen Prozesse gestaltet und reflektiert werden. Dies ist kennzeichnend für Gruppen- und Projektarbeitsphasen im Informatikunterricht. Insgesamt zeigt sich, dass die informatische Bildung unverzichtbare Beiträge bei der Realisierung der OECD-Zielstellungen zu erbringen hat.

3 Entwicklung der Grunderfahrungen des Informatikunterrichts

Mathematikunterricht ist dadurch allgemeinbildend, dass er drei Grunderfahrungen ermöglicht [Wi95]:

(M1) „Erscheinungen der Welt um uns, die uns alle angehen oder angehen sollten, aus Natur, Gesellschaft und Kultur, in einer spezifischen Art wahrzunehmen und zu verstehen,

(M2) mathematische Gegenstände und Sachverhalte, repräsentiert in Sprache, Symbolen, Bildern und Formeln, als geistige Schöpfungen, als eine deduktiv geordnete Welt eigener Art kennen zu lernen und zu begreifen,

(M3) in der Auseinandersetzung mit Aufgaben Problemlösefähigkeiten, die über die Mathematik hinaus gehen, (heuristische Fähigkeiten) zu erwerben."

Die Grunderfahrungen sind „vielfältig miteinander verknüpft".

Beim Übertragen der Konzeption der Grunderfahrungen des Mathematikunterrichts in die Informatik sollen die folgenden Intentionen erhalten bleiben: Die Grunderfahrung M1 thematisiert Anwendungen der Mathematik und damit Weltverstehen. Die Grunderfahrung M2 charakterisiert Mathematik als eine Welt eigener Art, die deduktiv geordnet ist, und die sich mit spezifischen Gegenständen und Sachverhalten befasst. Die Grunderfahrung M3 thematisiert das Entwickeln von Kompetenzen, die auch außerhalb

der Mathematik Anwendung finden können. Zusätzlich zu den Intentionen sollen der grundsätzliche Aufbau und die Art der Formulierung weitgehend erhalten bleiben.

In der Informatik geht es zentral um die Informationsverarbeitung (auch Komplexitätsbewältigung) mithilfe von Informatiksystemen, sodass es naheliegend ist, sich in der ersten Grunderfahrung I1 darauf zu beziehen: *Informatikunterricht ist dadurch allgemeinbildend, dass er als Grunderfahrung ermöglicht, Informatiksysteme und ihre Wirkungen in unterschiedlichen Lebensbereichen zu entdecken, zu verstehen und zu bewerten.*

Zentrale Gegenstände der Informatik sind Algorithmen und Daten. In der Informatik wird ein Problem häufig dadurch gelöst, dass ein Algorithmus oder Modell entwickelt und als Informatiksystem implementiert wird. Die zweite Grunderfahrung I2 lautet daher bei Berücksichtigung der o.g. Intention: *Informatikunterricht ist dadurch allgemeinbildend, dass er als Grunderfahrung ermöglicht, zu erkennen, dass sich Handlungen, die man tut oder plant, als Algorithmen formulieren und ggf. weiter in Programme überführen lassen, dass sich Realitätsausschnitte durch Modellierung für ein Informatiksystem aufbereiten lassen und dass Informatiksysteme von Menschen gestaltet sind.* In der zweiten Grunderfahrung geht es um das Erkennen (eigentlich auch um das Staunen), dass es überhaupt möglich ist, Teile der Realität in ein Informatiksystem abzubilden. Des Weiteren geht es darum zu erkennen, dass es unterschiedliche Problemlösungen geben kann – je nach dem Bearbeiter mit seinen Stärken, Schwächen, Ideen, Motivationen usw.

Ein wesentliches Ziel von Informatikunterricht ist das Entwickeln von Kompetenzen. [Gi08] [Km04] [Kl03] Daher lässt sich als Grunderfahrung I3 formulieren: *Informatikunterricht ist dadurch allgemeinbildend, dass er als Grunderfahrung ermöglicht, in der Auseinandersetzung mit Aufgaben Problemlösefähigkeiten zu erwerben, die inner- und außerhalb des Informatikunterrichts und auch außerhalb der Schule anwendbar sind.*

Wir fassen die Grunderfahrungen des Informatikunterrichts zusammen:

Informatikunterricht ist dadurch allgemeinbildend, dass er drei Grunderfahrungen ermöglicht:

(I1) Informatiksysteme und ihre Wirkungen in unterschiedlichen Lebensbereichen zu entdecken, zu verstehen und zu bewerten,

(I2) zu erkennen, dass sich Handlungen, die man tut oder plant, als Algorithmen formulieren und ggf. weiter in Programme überführen lassen, dass sich Realitätsausschnitte durch Modellierung für ein Informatiksystem aufbereiten lassen und dass Informatiksysteme von Menschen gestaltet sind,

(I3) in der Auseinandersetzung mit Aufgaben Problemlösefähigkeiten zu erwerben, die inner- und außerhalb des Informatikunterrichts und auch außerhalb der Schule anwendbar sind.

Die Grunderfahrungen sind vielfältig miteinander verknüpft.

Die drei Grunderfahrungen sind im Augenblick als Thesen anzusehen. Im Abschnitt 4 werden die Thesen überprüft.

4 Kritische Reflexion zu den entwickelten Grunderfahrungen

Nachfolgend werden Beziehungen zwischen den Grunderfahrungen des Informatikunterrichts einerseits und den GI-Empfehlungen zu Bildungsstandards Informatik, den einheitlichen Prüfungsanforderungen der KMK in der Abiturprüfung Informatik (EPA Informatik), dem IniK-Projekt, den Kriterien von Heymann und den „Ludwigsfelder Thesen" andererseits hergestellt. Dabei soll sich zeigen, in welchem Maße die entwickelten Grunderfahrungen treffsicher für das Schulfach Informatik sind. Um subjektive Einflüsse bei den Zuordnungen (siehe unten) weitgehend auszuschließen, wurden diese von beiden Autoren unabhängig vorgenommen; bei unterschiedlichen Zuordnungen erfolgte eine weitere gemeinsame Analyse.

Bei den GI-Empfehlungen zu Bildungsstandards Informatik handelt es sich um Mindeststandards. [Gi08] Sie beschreiben Grundsätze eines guten Informatikunterrichts und geben an, über welche Kompetenzen jede Schülerin und jeder Schüler – und zwar unabhängig von der Schulform – auf dem Gebiet der informatischen Bildung am Ende der 7. und am Ende der 10. Jahrgangsstufe verfügen sollte. Nachfolgend werden Kompetenzen, die in den GI-Empfehlungen angegeben sind, den drei Grunderfahrungen zugeordnet.

Der Grunderfahrung I1 lassen sich die folgenden Kompetenzen zuordnen:

Schülerinnen und Schüler

- lesen und verstehen Handlungsvorschriften für das Arbeiten mit Informatiksystemen
- unterscheiden Eingaben und Ausgaben realer Automaten
- identifizieren unterschiedliche Zustände realer Automaten
- beschreiben Zustandsübergänge realer Automaten und die Eingaben, die sie ausgelöst haben
- erläutern das Prinzip der Eingabe, Verarbeitung und Ausgabe von Daten (EVA-Prinzip) als grundlegendes Arbeitsprinzip von Informatiksystemen
- benennen wesentliche Bestandteile von Informatiksystemen
- erkennen den Grundaufbau von Informatiksystemen in Alltagsgeräten wieder
- erschließen sich selbstständig neue Anwendungen und Informatiksysteme
- beschreiben ihren Umgang mit Informatiksystemen aus ihrer eigenen Lebenswelt
- bewerten die Auswirkungen der Automatisierung in der Arbeitswelt
- erkennen die Notwendigkeit einer verantwortungsvollen Nutzung von Informatiksystemen
- betrachten Informatiksysteme und Anwendungen unter dem Aspekt der zugrunde liegenden Modellierung
- untersuchen bereits implementierte Systeme
- stellen Vermutungen über Zusammenhänge und Lösungsmöglichkeiten im informatischen Kontext dar

Der Grunderfahrung I2 lassen sich die folgenden Kompetenzen zuordnen:

Schülerinnen und Schüler

- stellen Information in unterschiedlicher Form dar

- benennen und formulieren Handlungsvorschriften aus dem Alltag
- lesen formale Darstellungen von Algorithmen und setzen sie in Programme um
- benutzen die algorithmischen Grundbausteine zur Darstellung von Handlungsvorschriften
- entwerfen Handlungsvorschriften als Text oder mit formalen Darstellungsformen
- überführen umgangssprachlich gegebene Handlungsvorschriften in formale Darstellungen
- analysieren Sachverhalte und erarbeiten angemessene Modelle
- beurteilen das Modell, die Implementierung und die verwendeten Werkzeuge kritisch

Der Grunderfahrung I3 lassen sich die folgenden Kompetenzen zuordnen:

Schülerinnen und Schüler:

- entwerfen, implementieren und beurteilen Algorithmen
- geben Problemlösungen in einer Dokumentenbeschreibungssprache, Abfragesprache oder Programmiersprache an
- interpretieren Fehlermeldungen bei der Arbeit mit Informatiksystemen und nutzen sie produktiv
- wählen problemadäquate Anwendungen selbstständig aus
- analysieren Sachverhalte und erarbeiten angemessene Modelle
- setzen einfache Datenmodelle in relationale Modelle um und realisieren diese mit einem Datenbanksystem
- beurteilen das Modell, die Implementierung und die verwendeten Werkzeuge kritisch
- planen Arbeitsabläufe und Handlungsfolgen
- verknüpfen informatische Inhalte und Vorgehensweisen mit solchen außerhalb der Informatik
- kooperieren in arbeitsteiliger Gruppenarbeit
- reflektieren gemeinsam Ansatz, Ablauf und Ergebnis des Projekts
- verwenden elektronische Plattformen (Schulserver, Internetplattform) zum Austausch und zur gemeinsamen Bearbeitung von Dokumenten

Auch wenn keine Vollständigkeit bei den Zuordnungen angestrebt wurde, wird deutlich, dass ein Informatikunterricht, der auf den GI-Empfehlungen beruht, wesentliche Beiträge zur Entwicklung der Grunderfahrungen I1, I2 und I3 in der Sekundarstufe I erbringen kann.

Eine Analyse der EPA Informatik [Km04] liefert ein weitergehendes Ergebnis. Wie die nachfolgende Tabelle belegt, lassen sich sämtliche Kompetenzen und Inhalte, die in den EPA Informatik angegeben sind, den Grunderfahrungen zuordnen (die Abkürzungen sind den Anhängen 1 und 2 aus [Fo08] entnommen).

	Fachliche und methodische Kompetenzen der EPA Informatik	Fachliche Inhalte der EPA Informatik
I1	A3c) A3d) A4a) A4e)	B2a) B2b) B2d) B2e) B3a) B3b) B3c) B3d)
I2	A1a) A1b) A2c) A3c) A4d)	B1a) B1b) B1d) B1e) B1f) B1g) B1h) B1i) B2b) B2c) B2d) B2f)
I3	A1c) A2a) A2b) A3a) A3b) A3e) A3f) A4b) A4c)	B1a) B1b) B1c) B2b) B2f) B3c)

Auf einen speziellen Aspekt soll hingewiesen werden: Grenzen der Berechenbarkeit sind nach den EPA Informatik ein fachlicher Inhalt der Abiturprüfung. Ziel ist dabei letztlich eine begründete Positionsbestimmung zwischen den Polen „Technikfeindlichkeit“ und „Computergläubigkeit“. [Fo10, S. 134] Die Grunderfahrung I1 kann den Aufbau von Computergläubigkeit, die Grunderfahrung I2 den Aufbau von Technikfeindlichkeit verhindern helfen.

Kennzeichnend für die Initiative *IniK – Informatik im Kontext* [DK11] sind drei Prinzipien für die Gestaltung von Informatikunterricht: Orientierung an Kontexten, Orientierung an den GI-Empfehlungen zu Bildungsstandards Informatik und methodische Vielfalt. Die Orientierung an lebensweltlichen und schülerbedeutsamen Kontexten lässt sich der Grunderfahrung I1 zuordnen. Kompetenzen, die in den GI-Empfehlungen aufgeführt sind, lassen sich allen drei Grunderfahrungen zuordnen (siehe oben). Methodenvielfalt im Sinne des konstruktivistischen Lernansatzes kann die Kompetenzentwicklung im Sinne der Grunderfahrung I3 fördern. Ausgearbeitete Beispiele finden sich auf der Website des Projekts [In12] und als Beilagen zur Fachzeitschrift LOG IN.

Hans Werner Heymann formulierte 1997 sieben Kriterien für die Einordnung des Bildungswertes von Unterrichtsfächern: Lebensvorbereitung, Stiftung kultureller Kohärenz, Weltorientierung, Anleitung zum kritischen Vernunftgebrauch, Entfaltung von Verantwortungsbereitschaft, Einübung in Verständigung und Kooperation sowie Stärkung des Schüler-Ichs. [He97] [Wi03] Der Versuch von Fachleitern für Informatik, diese Kriterien als Maß für den Allgemeinbildungsgrad von Informatikunterricht zu nutzen, führte zu vier Thesen. [Lu03] Nachfolgend werden die entwickelten Grunderfahrungen des Informatikunterrichts zu den „Ludwigsfelder Thesen“ in Beziehung gesetzt.

Die Grunderfahrung I1 findet sich in der Begründung der 1. „Ludwigsfelder These“ teilweise wieder: „Die Kenntnis, Anwendung und kritische Reflexion der grundlegenden Konstruktionsprinzipien von Informatiksystemen dient daher der Lebensvorbereitung und der Orientierung in einer von diesen Systemen geprägten Welt.“ Die Grunderfahrung I1 geht dabei vom Phänomen, die 1. „Ludwigsfelder These“ von den Konstruktionsprinzipien aus.

Beziehungen eines Aspekts der Grunderfahrung I2 zur 2. „Ludwigsfelder These“ lassen sich erkennen: „Der Informatikunterricht trägt entscheidend zur Entwicklung der Lernenden zu mündigen Bürgern bei, indem sie erkennen, dass Informatiksysteme von Menschen gestaltet sind. Sie reflektieren im Unterricht ihre eigenen exemplarischen Erfahrungen mit der Gestaltung von diesen Systemen.“ In der Grunderfahrung I2 werden die grundlegende Einsicht, dass sich Handlungen überhaupt in Algorithmen und

Programme überführen lassen, und die Sinnhaftigkeit von Modellierungen als wesentlich herausgestellt.

In der 4. „Ludwigsfelder These“ wird der Blick auf Projekte im Informatikunterricht unter dem Aspekt der Entwicklung von Sozialkompetenz gerichtet und in dem Zusammenhang auch der Beitrag für die Organisation der eigenen Arbeit und lebenslanges Lernen betont. Dies findet sich in der Grunderfahrung I3 wieder, in der es ganz allgemein um den Erwerb von Problemlösefähigkeiten geht, zu denen unter anderem die Bereitschaft und Befähigung der Zusammenarbeit mit anderen gehört.

In der Begründung der 3. „Ludwigsfelder These“ wird ausgeführt: „Insbesondere im Informatikunterricht erwerben die Lernenden unter Verwendung unterschiedlicher Paradigmen der Modellierung die Fähigkeit, Ausschnitte aus Alltagssituationen zielgerichtet abzugrenzen, zu strukturieren und formal zu beschreiben und darüber zu kommunizieren.“ Modellierung findet sich in den Grunderfahrungen I2 und I3 wieder. Als Grunderfahrung I2 sollen Schülerinnen und Schüler erkennen, dass Modellierung überhaupt möglich und sinnvoll ist, die Grunderfahrung I3 thematisiert Problemlösefähigkeiten, die Modellierungstechniken beinhalten.

Zusammenfassend lässt sich feststellen, dass die Grunderfahrungen und die „Ludwigsfelder Thesen“ nicht deckungsgleich sind; sie lassen sich jedoch aufeinander beziehen.

5 Resümee

Die Konzeption der Grunderfahrungen lässt sich vom Mathematikunterricht auf den Informatikunterricht übertragen. Reflexionen zum Arbeitsergebnis (unter anderem die Bezüge zu den EPA Informatik) machen deutlich, dass die Grunderfahrungen I1, I2 und I3 wirklich von besonderer Bedeutung für den Informatikunterricht sind. Sie beschreiben in kompakter Form den Beitrag der informatischen Bildung an der Allgemeinbildung. Das Vorgehen in zwei Schritten hat sich als sinnvoll erwiesen. Im ersten Schritt wurden die Grunderfahrungen als Thesen generiert; es handelte sich um eine begründete Transferleistung aus der Mathematik in die Informatik. Im zweiten Schritt konnten die Thesen verifiziert werden. Das Übertragen von Arbeitsergebnissen in ein anderes Schulfach ist garantiert nicht immer sinnvoll. In unserem Fall erhielten wir jedoch ein interessantes und brauchbares Arbeitsergebnis.

Referenzen

[DK11] Diethelm, I.; Koubek, J.; Witten H.: IniK - Informatik im Kontext - Entwicklungen, Merkmale und Perspektiven. In: LOG IN, 31. Jg. (2011/2012), H. 169/170, S. 97-105.

[Fo08] Fothe, M.: Bildungsstandards Informatik für die Sekundarstufe II – Vorüberlegungen zur Entwicklung. In: Brinda, T. u.a. (Hrsg.): Didaktik der Informatik – Aktuelle Forschungsergebnisse. Lecture Notes in Informatics, Bonn 2008, S. 107-116. http://subs.emis.de/LNI/Proceedings/Proceedings135/gi-proc-135-010.pdf

[Fo10] Fothe, M.: Kunterbunte Schulinformatik – Ideen für einen kompetenzorientierten Unterricht in den Sekundarstufen I und II. LOG IN Verlag, Berlin 2010.

[Gi08] Gesellschaft für Informatik (Hrsg.): Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. Empfehlungen der Gesellschaft für Informatik e. V. Beilage zu LOG IN, Heft 150/151 (2008). http://www.informatikstandards.de/

[Gi12] http://hyfisch.de/Fachgruppe/tagung11/fothe

[He97] Heymann, H. W.: Allgemeinbildung als Aufgabe der Schule und als Maßstab für Fachunterricht. In: Heymann, H. W. (Hrsg.): Allgemeinbildung und Fachunterricht. Bergmann + Helbig, Hamburg 1997, S. 7-17.

[In12] http://www.informatik-im-kontext.de

[Kl07] Klafki, W.: Neue Studien zur Bildungstheorie und Didaktik. Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik. Beltz Verlag, Weinheim und Basel ^{6}2007.

[Kl03] Klieme, E. et al.: Zur Entwicklung nationaler Bildungsstandards. Eine Expertise. Bildungsreform Band 1. BMBF, Berlin 2003. http://www.bmbf.de/pub/zur_entwicklung_nationaler_bildungsstandards.pdf

[Km04] Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik. Beschluss vom 1.12.1989 i. d. F. vom 5.2.2004. Beschlüsse der Kultusministerkonferenz. Luchterhand, 2004. http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1989/1989_12_01-EPA-Informatik.pdf

[Km12] http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2012/2012_10_18-Bildungsstandards-Mathe-Abi.pdf

[Lu03] Ludwigsfelder Thesen. In: LOG IN, 23. Jg. (2003), H. 124, S. 33. http://ddi.cs.uni-potsdam.de/HyFISCH/Informieren/politik/LudwigsfelderThesen2003.pdf

[Oe05] OECD: Definition und Auswahl von Schlüsselkompetenzen, Zusammenfassung 2005. http://www.oecd.org/pisa/35693281.pdf

[SS11] Schubert, S.; Schwill, A.: Didaktik der Informatik. Spektrum Akademischer Verlag, Heidelberg ^{2}2011

[Wi95] Winter, H.: Mathematikunterricht und Allgemeinbildung. In: Mitteilungen der Gesellschaft für Didaktik der Mathematik Nr. 61 (1995), S. 37-46.

[Wi03] Witten, H.: Allgemeinbildender Informatikunterricht? Ein neuer Blick auf H. W. Heymanns Aufgaben allgemeinbildender Schulen. In: Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht. 10. GI-Fachtagung Informatik und Schule INFOS 2003. Lecture Notes in Informatics, Bonn 2003, S. 59-75. http://subs.emis.de/LNI/Proceedings/Proceedings32/GI-Proceedings.32-7.pdf

Die angegebenen Internetquellen wurden zuletzt am 20. Juni 2013 geprüft.

Kryptographie im Wahlpflichtbereich der Sekundarstufe I

Katrin Büttner
Mittelschule
„J. W. v. Goethe“
01809 Heidenau
buettner@ibisath.info

Thomas Knapp
Mittelschule
Kötzschenbroda
01445 Radebeul
knapp@ibisath.info

Abstract: Im sächsischen Lehrplan des Pflichtfaches Informatik ist in jeder Klassenstufe ein Wahlpflichtbereich zum Thema „Verschlüsseln von Informationen“ vorgegeben. Im Workshop soll vorgestellt werden, wie dieser Wahlpflichtbereich über alle Klassenstufen der Sekundarstufe I hinweg aufeinander aufbauend behandelt werden kann. Die Autoren greifen dabei auch auf Ergebnisse von Lehrerfortbildungen zurück.

1 Situationsbeschreibung

Im Lehrplan der sächsischen Mittelschule sind in Klasse 7 bis 10 je zwei Unterrichtsstunden für den Wahlpflichtbereich „Verschlüsseln von Informationen“ vorgesehen. Die Inhalte werden mit den folgenden Übersichten (links: verpflichtende Bestandteile, rechts: Hinweise) vorgegeben.

Klassenstufe 7

Kennen ausgewählter Codes und Chiffren unter historischem Aspekt Anwenden eines Codes auf das Codieren und Decodieren und einer Chiffre auf das Chiffrieren und Dechiffrieren einfacher Botschaften	Unterschied zwischen Codierung und Chiffrierung verdeutlichen Morse-Code, Blindenschrift Caesar-Chiffrierung

Klassenstufe 8

Einblick gewinnen in das maschinelle Codieren und Chiffrieren von Texten mithilfe von Algorithmen	Code-Tabellen: Binär-Code, ASCII-Code Chiffrier-Geräte: Chiffrier-Scheibe, Skytale Realisierung mit Programmierumgebung, Makros in Anwendungsprogrammen

Klassenstufe 9

Kennen von Verschlüsselungen beim Datenaustausch in Netzen	Senden von verschlüsselten E-Mails, öffentliche, nichtöffentliche Schlüssel

Klassenstufe 10

Anwenden des Versteckens von Nachrichten in Texten und Bildern	Steganografie

Die Lernzielebenen (*Einblick gewinnen, Kennen*, Übertragen, Beherrschen, *Anwenden*, Beurteilen/Sich positionieren, Gestalten/Problemlösen) beschreiben die Tiefe der Ausprägung und die Art der Behandlung des ausgewiesenen fachlichen Inhalts und werden im Lehrplanwerk ausführlich beschrieben [LP04].

2 Inhalte

Im Workshop werden wir uns an den oben vorgegeben Inhalten orientieren, dabei aber verschiedene methodische Umsetzungen aufzeigen. So kann das Material in Vertretungsstunden verwendet werden, es ist aber auch möglich den Umfang bis zu einem Projekt hin auszubauen.

2.1 Grundlagen – klassische Verfahren

Vor der eigentlichen Beschäftigung mit Verschlüsselungsverfahren ist es erforderlich, klare Begriffe beim Lernenden zu erzeugen. Diese sind aus unserer Sicht Sender, Empfänger, Nachricht, Klartext, Geheimtext und Schlüssel. Besonders wichtig ist hierbei die klare Unterscheidung zwischen Codieren und Chiffrieren, da diese von den Schülern umgangssprachlich oft synonym verwendet werden. Hier bietet es sich an, eine Vielzahl von Beispielen aus der Erfahrungswelt der Schüler zu besprechen, z.B. Blindenschrift, Morsealphabet oder Geheimschriften.

Als Einstieg ins Verschlüsseln ist das Cäsar-Verfahren gut geeignet, da es leicht verständlich ist. Bei der Abarbeitung durch die Lernenden wird schnell deutlich, dass ein hohes Maß an Konzentration notwendig ist, besonders beim Arbeiten mit der Vigenère-Verschlüsselung.

Zur Erleichterung für die Schüler haben wir als Hilfsmittel sowohl eine Tabelle als auch die Cäsarscheibe benutzt (Abb. 1) und zusätzliche farbige Kennzeichnung von Klartext, Geheimtext und Schlüssel gefordert.

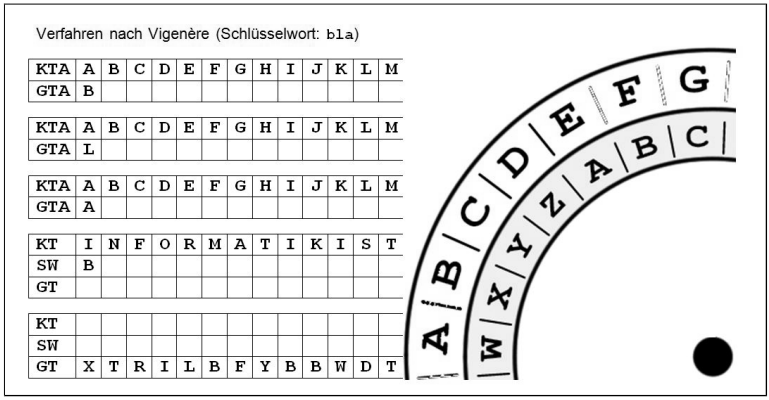


Abb. 1: Tabelle vs. Cäsar-Scheibe

2.2 Standardsoftware

Unter der Voraussetzung, dass die Schüler das Cäsar-Verfahren bereits „am Papier“ gelernt und geübt haben kann man sowohl Textverarbeitung als auch Tabellenkalkulation nutzen, um Handlungsschritte zu automatisieren.

In der Textverarbeitung soll das automatische „Suchen und Ersetzen“ zum Einsatz kommen (Abb. 2). Auf den ersten Blick erscheint das sehr einfach. Bei genauerer Betrachtung bemerken die Lernenden, dass ein einfaches Ersetzen zu Fehlern führt. Eine korrekte Ersetzung ist nur durch die Einführung eines Sonderzeichens möglich, das *nicht* im Alphabet enthalten ist.

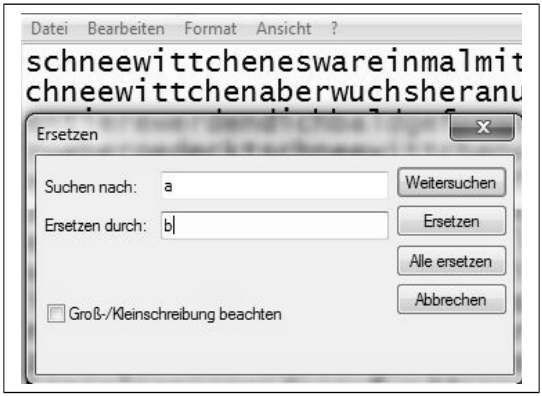
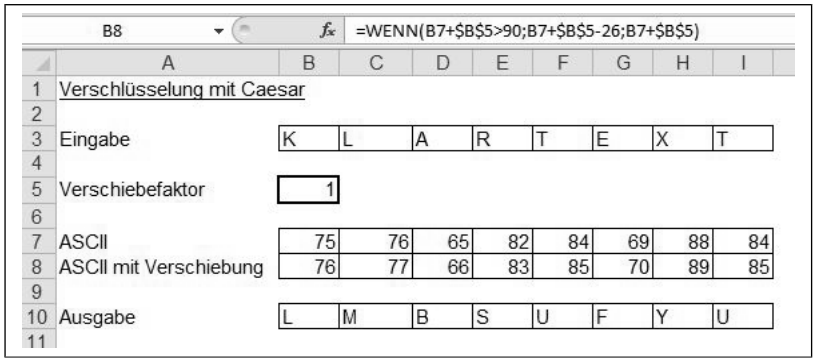


Abb. 2: Klartext mit Fenster „Suchen – Ersetzen“

In der Tabellenkalkulation ist Voraussetzung, dass die Lernenden mit Formeln und Funktionen arbeiten können. Zunächst soll eine Tabelle entworfen werden, die übersichtlich Klartext, Schlüssel und Geheimtext darstellt. Zur Berechnung der Zeichen des Geheimtextes sind die Funktionen ZEICHEN(), CODE() und WENN() notwendig. Zunächst sollten die Funktionen einzeln verwendet werden (Abb. 3), letztendlich können alle Berechnungen in einer einzigen Formel geschachtelt werden. Zusätzlicher Aufwand ist notwendig um die zyklische Vertauschung beim Erreichen des Endes des Alphabetes zu realisieren.



	A	B	C	D	E	F	G	H	I
1	Verschlüsselung mit Caesar								
2									
3	Eingabe	K	L	A	R	T	E	X	T
4									
5	Verschiebefaktor	1							
6									
7	ASCII	75	76	65	82	84	69	88	84
8	ASCII mit Verschiebung	76	77	66	83	85	70	89	85
9									
10	Ausgabe	L	M	B	S	U	F	Y	U
11									

Abb. 3: Verschlüsselung mit der Tabellenkalkulation

2.3 moderne Verfahren

Nach der Behandlung klassischer Verschlüsselungsverfahren sollen die Lernenden auch Grundlagen moderner Verfahren kennen lernen. Die benötigten mathematischen Kenntnisse übersteigen allerdings das Niveau der Sekundarstufe I.

Auf spielerische Weise kann den Lernenden das Grundprinzip der asymmetrischen Verschlüsselung verdeutlicht werden. Die Grundbegriffe public key und private key haben wir am Beispiel des Fahrradschlosses erläutert. (Abb. 4)

- Geöffnetes Schloss = public key
  (jeder kann es schließen = eine Nachricht verschlüsseln)

- Zahlenkombination/Schlüssel für das Schloss = private key
  (nur der Eigentümer kennt die Zahlenkombination/hat den Schlüssel und kann das Schloss wieder öffnen und somit die Nachricht entschlüsseln.)

Zur Demonstration der Verfahren bieten sich jetzt verschiedenste Tools an (siehe Kapitel 2.4).

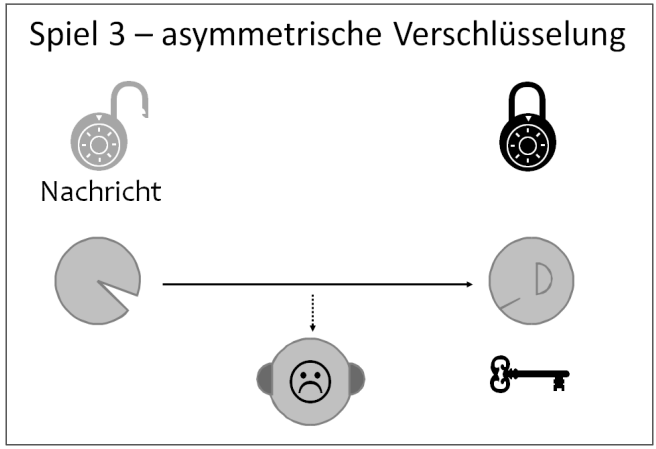


Abb. 4: Schematische Darstellung für asymmetrische Verschlüsselung

Eine weitere interessante Möglichkeit über Verschlüsselung zu sprechen ist die Steganographie (Abb. 5). Die meisten Lernenden haben bereits mit Geheimtinte geschrieben oder geheime Zeichen an Hauswänden gesehen, sie kennen somit das Grundprinzip. Das versteckte Übertragen von Nachrichten kann man an historischen Beispielen beschreiben (und recherchieren) lassen oder aktuelle Software nutzen, um z.B. Grafikdateien mit verstecktem Text selbst herzustellen (siehe Kapitel 2.4).

Abb. 5: Beispiel für angewandte Steganographie [MF]

2.4 spezielle Software und Tools

Zur Veranschaulichung verschiedenster Bereiche des Themas „Verschlüsselung von Informationen“ können unterschiedliche Softwareprodukte eingesetzt werden:

- Textverarbeitung zur Ermittlung der Buchstabenhäufigkeit (Ersetze „A“ durch „A“ und notiere die Anzahl der Ersetzungen)
- Tabellenkalkulation zur Berechnung und Darstellung der Buchstabenhäufigkeit
- Cryptool [CT] zum
    - Ver- und Entschlüsseln mit verschiedenen klassischen und modernen Verfahren
    - Analysieren und Entschlüsseln unbekannter Geheimtexte
    - Demonstration ausgewählter Verfahren z.B. RSA (der mathematische Aufwand entfällt, das Verfahren tritt in den Vordergrund)
- S-Tools [ST] zum Verstecken von Nachrichten in Bildern
- QR-Code-Scanner und Generator [QR] zum Decodieren und Erstellen eigener Codes

3 Kryptografisches für „Zwischendurch“

3.1 Energizer

„Aktivierungsspiele sind nicht der eigentliche Gegenstand des Unterrichts sondern werden zu unterschiedlichen Zwecken und bei verschiedenen Gelegenheiten als pädagogisches Mittel in den Unterricht eingestreut, um sich zu entspannen, Dampf abzulassen, Ermüdung vorzubeugen.“ [CA91]

In kurzweiliger Form können einfache Verfahren behandelt werden:

- Blindenschrift (ein Kontakt zum Blinden- und Sehbehindertenverband ist sehr hilfreich)
- Freimaureralphabet
- Fackeltelegraf [FT], das „Spielen“ des Telegrafen erfordert ein besonders klares Verständnis der Aufgabe eines jeden Mitspielers (Abb. 6) Vorsicht: Im Zimmer lieber Besenstiele statt Fackeln verwenden. ☺

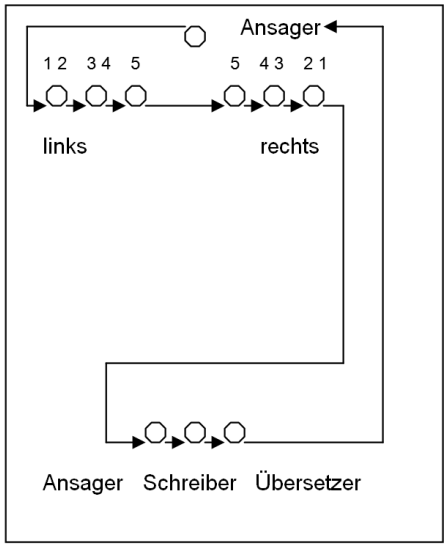


Abb. 6: Grundpositionen beim Einsatz des Fackeltelegrafen

3.2 Gruppenpuzzle

Die Betrachtung von historischen Geräten für die Verschlüsselung und Übertragung von Nachrichten sollte nicht außer Acht gelassen werden (z.B. Skytale, Zeigertelegraf, Enigma, QR- Code). Hierbei ist vor allem viel Rechercheaufwand notwendig, der durch verschiedene Methoden realisiert werden kann. Wir haben uns für die Methode des Gruppenpuzzles entschieden. Voraussetzung für ein erfolgreiches Arbeiten ist eine geeignete Auswahl von Texten und eine klare Struktur in der Aufgabenstellung (Abb. 7)

Verschlüsselungsverfahren und -geräte

	Skytale	Enigma
Zeitliche Einordnung		
Was ist es?		
Wie/ wozu genutzt?		
Bild		

Abb. 7: Ausschnitt aus dem Arbeitsblatt zum Gruppenpuzzle

Literaturverzeichnis

[LP04] Sächsisches Staatsministerium für Kultus, Comenius Institut (Hrsg.): Lehrplan Mathematik unter:
http://www.bildung.sachsen.de/apps/lehrplandb/
ODER
http://www.bildung.sachsen.de/apps/lehrplandb/downloads/lehrplaene/lp_ms_informatik_2009.pdf
(10.04.2013)

[ST] S-Tools, Steganographie-Tool zum Verstecken von Texten in Grafiken
http://download.chip.eu/de/S-Tools-4.00_41920.html
(26.05.2013)

[CT] Cryptool – Open-Source-Programm für Kryptographie und Kryptoanalyse
http://www.cryptool.org/de/
(26.05.2013)

[QR] QR-Code-Generator (ein Beispiel)
http://qrcode.kaywa.com/
(26.05.2013)

[MF] Mikrofilm/künstliche DNA
http://www.dna-schutz.de/kdna-identifizierung.html
(27.05.2013)

[FT] Veröffentlichung zum Fackeltelegraf im Durchgeführten Neigungskurs Kryptografie
http://www.sn.schule.de/~knapp/index.php?auswahl=nk_ft
(27.05.2013)

[CA91] nach Carol Apacki; 1991
www.smv.bw.schule.de/smv-spiele/spiele.pdf
(27.05.2013)

lili – eine Programmierumgebung für den Schulunterricht als Webanwendung

Michael Cyruk, Gregor Große-Bölting

Christian-Albrechts-Universität zu Kiel
24098 Kiel
mic@informatik.uni-kiel.de
ggb@informatik.uni-kiel.de

Abstract: *lili* ist eine webbasierte Programmierumgebung für die Sekundarstufe I, die Blockly und Kara miteinander kombiniert: In *lili* werden Programme per *drag and drop* zusammengestellt und programmiert wird eine virtuelle zwei-dimensionale Welt. Da das System webbasiert ist, können die Schülerinnen und Schüler sowohl in der Schule als auch zuhause arbeiten; der jeweilige Arbeitsstand wird im System abgelegt, eine Installation entfällt. Lehrkräfte können aus gegebenen Aufgabensätzen auswählen, eigene erstellen und Teilnehmer zu Kursen einladen.

lili ist eine Programmierumgebung für den Schulunterricht in Form einer Web-anwendung. Sie zeichnet sich aus durch:

- eine einfach verständliche grafische Welt, in der die namensgebende Libelle Lili programmiert wird. Diese Umgebung ist der von *Kara*[1] nachempfunden.
- den visuellen Programmeditor *Blockly*[2], der Programmieren mit der Maus per d*rag and drop* ermöglicht, ähnlich zu *Scratch*[3].
- eine besondere Lehrerkomponente. *lili* ist nicht auf einen gegebenen Aufgabenkatalog beschränkt, vielmehr hat das System den Charakter einer Plattform. Lehrkräfte können mit Hilfe eines grafischen Editors neue Aufgaben anlegen und zu sinnvollen Übungen zusammenfassen. Diese können sie dann an ihre Schülerinnen und Schüler weitergeben, indem sie sie über das System zu einem virtuellen Kurs einladen.

Seit etwa zwei Jahren erlebt die „Open Education"-Bewegung im Online-Bereich einen Aufschwung, ausgelöst durch Angebote wie beispielsweise *Codecademy*[4], *Udacity*[5], *Coursera*[6] und die *Khan Academy*[7]. Diesen Diensten ist gemein, dass sie sich Webtechniken und Webinfrastruktur zu Nutze machen. Die Vorteile liegen auf der Hand: So

[1] http://www.swisseduc.ch/informatik/karatojava/ (Abruf: 13.09.2013)
[2] https://code.google.com/p/blockly/ (Abruf: 12.09.2013)
[3] http://scratch.mit.edu/ (Abruf: 12.09.2013)
[4] http://www.codecademy.com/ (Abruf: 12.09.2013)
[5] https://www.udacity.com/ (Abruf: 12.09.2013)
[6] https://www.coursera.org/ (Abruf: 12.09.2013)
[7] https://www.khanacademy.org/ (Abruf: 12.09.2013)

müssen Lehrerinnen und Lehrer sich nicht mit der Installation und Wartung von Software beschäftigen und Schülerinnen, Schüler und Lehrkräfte können von jedem Ort auf das Angebot zugreifen. Das Ziel von *lili* ist es, diese Vorteile mit bewährten Ansätzen für Desktop-Lernsoftware zu verbinden, um so einen möglichst großen Komfort und Lernerfolg zu gewährleisten.

In diesem Aufsatz werden zunächst Kara und Blockly, die Ausgangspunkte für *lili* kurz beschrieben. Danach wird *lili* selbst vorgestellt, insbesondere wird die Lehrerkom-ponente erläutert. Der Aufsatz schließt mit einer kurzen Diskussion zur Weiterentwick-lung von *lili* ab.

Das System wurde im Rahmen zweier sechsmonatiger Bachelorarbeiten gemeinsam von Michael Cyruk [Cy13] und Gregor Große-Bölting [Gr13] erstellt. Es ist nicht nur mit Kara und Blockly, sondern auch mit *Scratch*, *snap!*[8] (ehemals *BYOB*), dem *MIT App Inventor*[9], *Karel the Robot* und *Turtle Geometry* verwandt [RNH04, S. 11f.].

1 Hintergrund: Kara und Blockly

1.1 Was ist Kara?

Kara ist eine Lernsoftware, die an der ETH Zürich von Jürg Nievergelt, Werner Hartmann, Raimond Reichert, Markus Brändle und Tobias Schlatter entworfen wurde. Die Software präsentiert ein zweidimensionales Feld mit verschiedenen Objekten – einmalig vorkommendes und vom Nutzer steuerbares Objekt ist ein Käfer namens Kara. Dieser Käfer kann von Programmen über das Feld bewegt werden, das heißt er kann sich geradeaus bewegen und sich auf der Stelle drehen. Über Sensoren kann Kara andere Objekte seiner Welt, wie Bäume, Käfer und Blätter, wahrnehmen. Ein Sensor ist dabei eine Funktion, die als Rückgabe einen Wahrheitswert liefert [RNH04, S. 27ff.]. Es besteht außerdem die Möglichkeit, dass Kara die Welt manipuliert, indem sie Blätter aufnimmt oder ablegt und Pilze verschiebt (indem sie gegen sie läuft).

Karas Welt ist ein Torus, das heißt, läuft Kara über den Rand der einen Seite, erscheint der Käfer auf der gegenüberliegenden Seite wieder [RNH04, S. 28f.]. Neben der Einflussnahme auf Karas Bewegung und Welt durch den Programmcode besitzt der Nutzer außerdem die Möglichkeit das Feld direkt zu manipulieren, indem er neue Objekte hinzufügt oder bestehende verschiebt. Für die Lehre und das Lernen sind dem Programm eine Reihe Aufgaben beigegeben, die sich jeweils auf verschiedenen Konfigurationen des Feldes ausprobieren lassen. Das Programmieren geschieht in einem zusätzlichen Fenster, das gleichzeitig eine integrierte Entwicklungsumgebung mit direkter Möglichkeit zur Kompilierung darstellt.

Von Kara sind inzwischen eine ganze Reihe von Varianten erschienen, die das Programmieren in verschiedenen Sprachen (Java, Ruby, Python, usw.) und mit ver-

[8] http://byob.berkeley.edu/ (Abruf: 12.09.2013)
[9] http://appinventor.mit.edu/ (Abruf: 12.09.2013)

schie-denen zu Grunde liegenden Konzepten ermöglichen (bspw. Multi-Kara für Nebenläufigkeit oder das ursprüngliche Kara, das auf endlichen Automaten aufbaut) [Sw07].

1.2 Was ist Blockly?

Blockly ist eine grafische Programmiersprache. Der Unterschied zu einer „konventionellen" Programmiersprache besteht darin, dass der Anwender nicht einen Programmcode über die Tastatur eingeben muss, sondern einzelne Programmier-„Bausteine" per *drag and drop* zusammenziehen kann. Die einzelnen Bausteine oder Blöcke verfügen dabei über Konnektoren, die das Zusammenfügen nur dann ermöglichen, wenn das Ergebnis ein syntaktisch korrektes Konstrukt darstellt (wie sich Puzzlestücke nur in einer spezifischen Weise zu einem Puzzle zusammensetzen lassen). Das heißt, dass diese Art des Programmierens Syntaxfehler ausschließt, was für Ein-steiger in die Programmierung sehr angenehm ist: Statt auf fehlende Klammern, kann sich auf Programmierkonzepte konzentriert werden [DG12].

Das Besondere an Blockly ist, dass es sich um kein „fertiges Programm", sondern um eine Bibliothek handelt, die für eigene Anwendungen angepasst werden kann. Eine weitere Besonderheit stellt die Tatsache dar, dass es sich bei Blockly um eine Anwendung handelt, die in JavaScript geschrieben ist und vollständig im Browser ausgeführt wird; für die Ausführung ist kein Server von Nöten, sie kann lokal auf dem eigenen Rechner geschehen [Fr13a].

Ein in Blockly erstelltes Programm kann in eine konventionelle Programmiersprache übersetzt werden; dafür bringt Blockly so genannte Generatoren mit, die das Übersetzen in JavaScript und Python ermöglichen. Das Schreiben eigener Generatoren für weitere Sprachen ist für einen versierten Nutzer mit ein wenig Aufwand möglich. Als Daten-austauschformat und zur Speicherung ist zudem die Übertragung in XML vorgesehen [Fr13a].

2. lili

In diesem Abschnitt werden wir erklären, wie Schülerinnen und Schüler mit *lili* programmieren und wie Lehrkräfte mit der Software arbeiten. Anschließend wird der aktuelle technische Stand des Systems beschrieben.

2.1 Wie programmiert man mit lili? - Eine Einführung

Abbildung 1 zeigt eine bereits für Fortgeschrittene konzipierte Aufgabe, die einem Aufgabenkatalog entstammt, der an der Christian-Albrechts-Universität zu Kiel im Rah-men des Schnupperstudiums Informatik zum Einsatz kommt. Die Programmierumge-bung besteht aus drei Teilen:

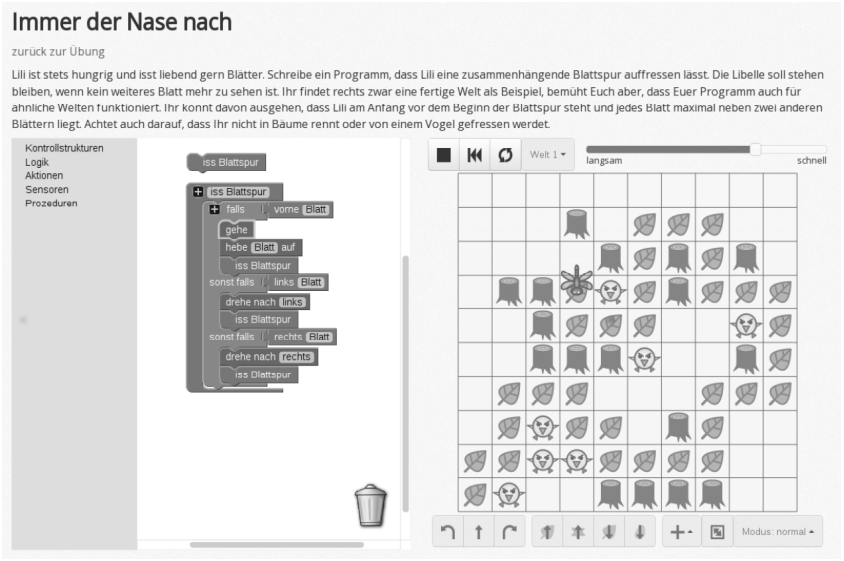


Abbildung 1: Beispielaufgabe „Immer der Nase nach“

(1) Oben gibt es neben dem Titel der aktuellen Aufgabe eine kurze **Aufgaben-beschreibung**, die vermittelt, was das Ziel dieser Aufgabe ist. In diesem Fall soll die Schülerin oder der Schüler ein Programm erstellen, welches Lili dazu bringt einem Pfad aus Blättern zu folgen und diese dabei aufzufressen.

(2) Links ist der **Programmeditor**. Hier kommt das angesprochene Blockly zum Einsatz. Blöcke können aus Kategorien ausgewählt werden, wie zum Beispiel „Kontrollstrukturen“ für Schleifen und Verzweigungen oder „Logik“ für logische Ausdrücke. Auf der weißen Arbeitsfläche ist ein fertiges Blockly-Programm zu sehen, welches die Beispielaufgabe mit Hilfe rekursiver Aufrufe der Prozedur „iss Blattspur“ löst.

(3) Rechts ist ***lilis* Welt** zu sehen, welche Karas nachempfunden ist. Der Käfer ist hier eine Libelle, neben Bäumen stellen auch Vögel ein unüberwindbares Hindernis dar. Außer Blätter kann die Libelle Lili auch Eier legen und Blumen aufsammeln. Über dem Spielfeld sind Kontrollelemente, um ein Programm auszuführen: Man kann es mit einem Knopf starten und stoppen, das Spielfeld nach der Ausführung wieder zurücksetzen, zwischen mehreren Beispielwelten hin- und herschalten und den Programmablauf verlangsamen oder beschleunigen. Die Steuerelemente links unter dem Spielfeld dienen dazu Lili manuell zu bewegen, was vor allem Anfänger in die Programmierumgebung einführen soll. Rechts unten gibt es Knöpfe, um das Spielfeld anzupassen, genauer, um neue Elemente auf das Spielfeld zu setzen, diese zu bewegen oder zu löschen und die Größe des Spielfeldes anzupassen. So kann man mit mehr als den zu einer Aufgabe gegebenen Beispielwelten Algorithmen auf Herz und Nieren testen.

Den Kern der Programmierumgebung bilden dabei der Programmeditor und die Blöcke, mit denen man letztendlich programmiert, weshalb wir diesen an dieser Stelle genauer vorstellen wollen.

Um die Konzepte von Kara und Blockly zusammenzuführen, benötigen wir zunächst Aktionen und Sensoren. Diese sind in Abbildung 2 zu erkennen.

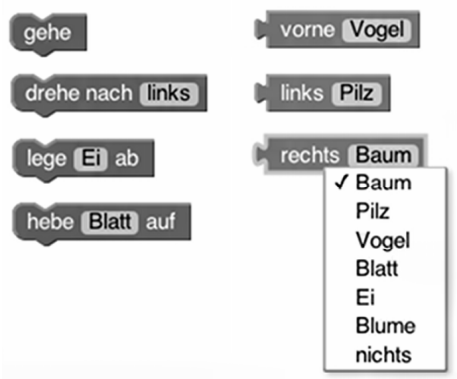


Abbildung 2: Die in *lili* zur Verfügung stehenden Aktionen (links) und Sensoren (rechts). Die Blöcke können mit Hilfe von Dropdown-Menüs angepasst werden.

Die der Libelle zur Verfügung stehenden Aktionen sind mehr oder weniger aus Kara übernommen worden. Nur aufgrund der zusätzlichen Gegenstände auf dem Feld ergeben sich beim Aufheben und Ablegen Unterschiede:

- Lili kann wie Kara einen Schritt nach vorne gehen, sofern kein Hindernis sie behindert.
- Zudem kann sie sich nach links und nach rechts drehen. Die Richtung kann einfach anhand eines Dropdown-Menüs verändert werden, welches erscheint, wenn der entsprechende Block angeklickt wird.
- Schließlich kann sie nicht nur Blätter aufheben, sondern auch Blumen. Ablegen kann sie diese aber nicht mehr. Stattdessen kann sie Eier legen und weiterhin Blätter auf freien Feldern platzieren. Auch hier dient ein Menü zur Auswahl der Elemente.

Die Sensoren sind Funktionen, die Wahrheitswerte liefern, und stellen Lilis Sinne dar. Die Libelle kann für jedes Element, das auf dem Spielfeld auftreten kann, beantworten, ob es vor ihr liegt. Das heißt sie kann erkennen, ob sich ein Baum, ein Pilz, ein Vogel, ein Blatt, ein Ei, eine Blume oder auch nichts vor ihr befindet. Das gleiche gilt für Gegenstände links und rechts neben der Libelle. Sie kann daher jedes Element auf dem Spielfeld gleich gut erkennen. Die zu prüfenden Elemente werden dabei explizit angegeben. Dazu werden sie wie bei den Aktionsblöcken aus einem Dropdown-Menü ausgewählt.

Die Form der Aktions- und Sensorblöcke deutet bereits an, wie diese zu nutzen sind: Die Einkerbungen an den oberen und unteren Kanten der Aktionsblöcke zeigen, dass diese vertikal untereinander gesteckt werden können. Es handelt es sich um Anweisungen, die sequenziell ausgeführt werden. Die Sensoren haben dagegen keine Einkerbungen. Stattdessen besitzen sie links eine Ausstülpung, so dass ein Sensorblock einem Puzzleteil ähnelt. Die Ausstülpung deutet an, dass die Blöcke einen Rückgabewert haben, in diesem Fall „wahr“ oder „falsch“, je nachdem, ob die spezifizierte Bedingung erfüllt ist oder nicht.

Die Sensorblöcke können allein in Verbindung mit den Aktionen nicht sinnvoll eingesetzt werden. Blockly aber bietet einige vorgefertigte Blöcke an, die dies ändern. Diese sind in Kategorien eingeteilt, welche je nach Bedarf hinzugefügt werden können, unter anderem Blöcke zur Steuerung des Kontrollflusses, Prozedurblöcke, Blöcke für logische Ausdrücke, Variablen, Text, Farben und mathematische Operationen. Abbil-dung 3 zeigt einige Beispiele, die sich bereits gut in *lili* einsetzen lassen.

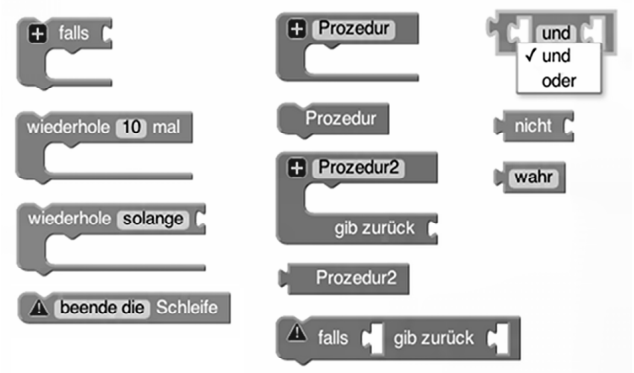


Abbildung 3: Einige vorgefertigte Blöcke aus Blockly lassen sich bereits gut mit den Aktionen und Sensoren kombinieren: Kontrollstrukturen (links), Prozeduren (Mitte) und logische Blöcke (rechts).

- **Kontrollstrukturen** Dies sind Blöcke, die Verzweigungen, While- und For-Schleifen nachbilden. Anweisungen, die innerhalb einer dieser Blöcke platziert werden, werden unter bestimmten Umständen ausgeführt. Die Blöcke können beliebig viele Anweisungen aufnehmen. Die Aussparungen der Blöcke „falls“ und „wiederhole solange“ passen zu den Sensoren, sie können hier als Bedingung für die (wiederholte) Ausführung eingesetzt werden.
- **Prozeduren** Im Blockly-Editor können Prozeduren ohne Rückgabewert und Funktionen mit Rückgabewert definiert werden. Diese können frei benannt werden. Mit der Definition einer Prozedur wird automatisch ein Block generiert, der denselben Namen trägt wie die Definition der Prozedur selbst, um diese ausführen zu können. Auf diese Weise kann wiederverwendbarer Code in Blockly erzeugt werden. Außerdem ermöglichen Prozeduren Rekursion.

– **Logik** Mit Hilfe der Logikblöcke können Sensoren kombiniert und negiert werden. Die logischen Konstanten „wahr“ und „falsch“ sind ebenfalls gegeben.

Blöcke, die mit einem Pluszeichen versehen sind, lassen sich außerdem über so genannte Mutatoren dynamisch erweitern [Fr13b]. Abbildung 4 zeigt den Dialog, der sich öffnet, sobald auf das Pluszeichen geklickt wurde. Nach demselben Prinzip wie in Blocklys Hauptansicht kann man in der Dialogblase weitere Blöcke mit dem dort dargestellten falls-Block verbinden, die dem falls-Block in der Hauptansicht neue Verzweigungen hinzufügen. Entfernt man die Blöcke im Dialog, so werden auch die Verzweigungen aus der Hauptansicht wieder entfernt.

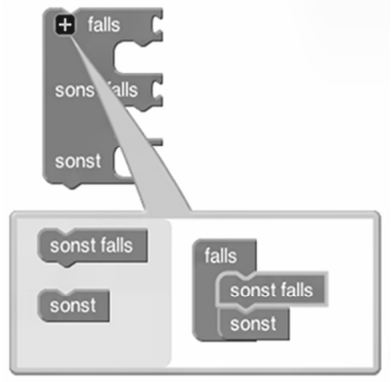


Abbildung 4: Mit Hilfe von Mutatoren lassen sich einige Blöcke erweitern. Der Block „falls“ erlaubt beispielsweise das dynamische Hinzufügen von Verzweigungen.

Mit der Kombination aus den selbst definierten Sensoren und Aktionen mit den Konstrukten, die Blockly bereitstellt, ist es also möglich die Libelle in ihrer Welt zu programmieren.

In der vorgestellten Beispielaufgabe werden einige dieser Konzepte genutzt, wie zum Beispiel die Mutatoren für den falls-Block. Auch eine Prozedur wird definiert. Man beachte, dass die Definition der Prozedur „iss Blattspur“ auf derselben Arbeitsfläche geschieht, auf der auch das Hauptprogramm entsteht. Wenn sich nichts anderes auf der Arbeitsfläche befindet, wird „iss Blattspur“ nicht ohne Weiteres ausgeführt. Deswegen befindet sich über der Definition noch ein Block, der die Prozedur auch tatsächlich aufruft.

2.2 Wie arbeitet eine Lehrkraft mit lili?

Um mit *lili* sinnvoll programmieren zu können bedarf es guter Aufgaben. Möchte eine Lehrkraft nicht nur aus dem bereits bestehenden Katalog von Aufgaben wählen, kann sie über das Webinterface **neue Aufgaben** anlegen. Eine Aufgabe besteht aus einem Titel,

einer Beschreibung, einer Auswahl von Blockkategorien und (mehreren) Beispielwelten, die einfach mit einem WYSIWYG-Editor zusammengeklickt werden.

Auf diese Weise erstellte Aufgaben lassen sich daraufhin zu sinnvollen **Übungen** zusammenfassen. Auch eine Übung erhält einen Titel und eine Beschreibung, in welcher man das Thema der Übung und ihr Ziel anreißen kann. Des Weiteren können Lehre-rinnen und Lehrer die Sichtbarkeit Ihrer Übungen festlegen: Entweder sind diese für jeden öffentlich einsehbar, nur für eingeloggte Benutzer zu erkennen oder auch nur privat für den Benutzer, der die Übung angelegt hat. Die letzte Option ist vor allem dann sinnvoll, wenn man neue Übungen vorbereiten möchte, diese aber erst später für den Unterricht freigegeben werden sollen.

Abbildung 5: Kurse erstellen in drei Schritten: Aufgaben entwerfen (links), Aufgaben zu Übungen zusammenstellen (rechts oben), Teilnehmer zu einem Kurs einladen (rechts unten)

Um die Übungen schließlich noch an ihre Schülerinnen und Schüler weiterzugeben, erstellen Lehrkräfte einen **Kurs**. Dieser bekommt einen Namen und eine Liste von Übungen zugewiesen. Die Übungen erscheinen dann auf den persönlichen Übersichts-seiten der eingeladenen Schülerinnen und Schüler. So sieht jeder Nutzer nach dem Login sofort als erstes, welche Übungen noch anstehen.

2.3 Aktueller technischer Stand

Die hier vorgestellte Software wurde im Rahmen eines Bachelorprojekts entwickelt. Inzwischen ist sie unter dem Titel *lili* unter der URL https://lili.informatik.uni-kiel.de erreichbar. Nutzer können sich dort registrieren, aus bestehenden Übungen wählen und Aufgaben mit Blockly lösen. Dabei wird über das animierte Spielfeld ein direktes visuelles Feedback gegeben, ob die Aufgabe gemäß der Aufgabenstellung gelöst werden konnte. Programme können gespeichert und im Nachhinein überarbeitet werden. Es gibt zudem eine Übersicht über schon gelöste Aufgaben. Für Lehrkräfte besteht, wie in 2.2 beschrieben, außerdem die Möglichkeit neue Aufgaben zu erstellen, diese zu Übungen zusammenzufassen und die Übungen an Kurse, das heißt an Gruppen von Schülerinnen und Schülern zu geben (dazu ist ein spezieller Zugang notwendig, der aktuell noch bei uns per Mail angefragt werden muss) .

lili ist in der Testphase. Die Software wurde bisher in einzelnen Klassen genutzt und auf dem derzeit verwendeten Testserver können etwa vier bis fünf Klassen gleichzeitig arbeiten. Die serverseitige Datenhaltung betrifft vor allem die Sicherung der erstellten Aufgaben, Spielfelder und Programme. Die Speicherung personenbezogener Daten soll dagegen auf ein Minimum beschränkt werden. Dazu gehört zum Beispiel die Empfehlung ein Pseudonym als Nutzernamen zu wählen.

3. Weitere Entwicklung

Auch wenn *lili* bereits einige grundlegende Funktionen bietet, ist die Entwicklung noch nicht abgeschlossen. Aktuell gibt es drei Stoßrichtungen, die für den weiteren Ausbau unseres Services maßgeblich sind:

(1) Die Entwicklung von *lili* geht weiter und soll in eine **offene Betaversion** münden. Vornehmliches Ziel ist es dabei zu ermitteln, ob die Software (in der jetzigen Form) von Schülern und Lehrern angenommen wird und wo es die Notwendigkeit zu Verbesserungen und Erweiterungen des aktuellen Stands gibt. Zahlreiche weitere Features sind geplant oder befinden sich aktuell in der Umsetzung. Insbesondere die Lehrerschnittstelle soll noch weiter ausgebaut werden. Dazu zählt beispielsweise es Lehrkräften zu ermöglichen, dass sie die Lösungen der Schülerinnen und Schüler über das Webinterface einsehen und kommentieren können. Bei der weiteren Entwicklung sind wir auf Feedback und Vorschläge angewiesen, wie *lili* sinnvoll erweitert und noch besser auf die Bedürfnisse der Nutzer angepasst werden kann.

(2) Die im Abschnitt 2.1 skizzierte Beispielaufgabe zeigt mit Verwendung von Rekursion bereits das schwierigste Thema, welches mit *lili* momentan behandelt werden kann. lili bietet zwar auch Variablen, diese können bisher aber nur mit Wahrheitswerten belegt werden. Es stellt sich daher die Frage, ob weitere Datentypen sinnvoll eingebunden werden können. Datenstrukturen wie Listen können noch gar nicht genutzt werden. Nebenläufigkeit ist zudem ein spannendes Thema, welches bisher nicht behandelt wird. Neben diesen **Erweiterungs-möglichkeiten** sollte es langfristig aber auch ein Konzept dafür geben, wie man von Blockly ausgehend in andere

„echte“ Programmiersprachen übergehen kann. Ein Ansatz könnte der Export von Blockly-Code in andere Programmiersprachen sein. Intern wird in *lili* sowieso JavaScript eingesetzt, so dass man dies auch nach außen sichtbar machen könnte.

(3) Bei der Entwicklung war es uns wichtig, eine Plattform zu schaffen, die von Lehrerinnen und Lehrern hinsichtlich ihrer eigenen Vorstellungen genutzt werden kann. Es besteht zwar die Möglichkeit bestehende Aufgabensätze zu übernehmen; es ist aber ebenso wenig ein Problem eigene Aufgaben zu erstellen (und diese – sofern es gewünscht ist – zu teilen). Zur Zeit wird an der Christian-Albrechts-Universität zu Kiel im Rahmen einer weiteren Bachelorarbeit ein Aufgabensatz zu *lili* erstellt, welcher der technischen Implementierung ein **didaktisches Konzept** gegenüber stellen wird. Dieses soll aufzeigen, was man mit *lili* erreichen kann und so Anregungen für eigene Übungsaufgaben liefern.

Literaturverzeichnis

[Cy13] Cyruk, Michael: Blockly für den Schulunterricht – Entwurf und Implementierung einer grafischen Programmierumgebung als Teil einer Lernplattform. Bachelorarbeit, Kiel, 2013. URL: https://lili.informatik.uni-kiel.de/static/theses/ausarbeitung_mic.pdf (Abruf: 12. 09. 2013).

[DG12] Dalinghaus, Klaus ; Gieseke, Werner: Von Scratch über BYOB nach Java. In: LOG IN (2011/2012), Nr. 172/173, S. 101–113.

[Fr13a] Fraser, Neil: Installation – Installing and Customizing Blockly. Stand: 26.07.2013. URL: http://code.google.com/p/blockly/wiki/Installation, (Abruf: 13.09.2013).

[Fr13b] Fraser, Neil: Creating Mutators – Make blocks deeply configurable. Stand: 16.08.2013. URL: https://code.google.com/p/blockly/wiki/CreatingMutators (Abruf: 13. 09. 2013).

[Gr13] Große-Bölting, Gregor: Blockly für den Schulunterricht – Konzeptionierung und serverseitige Implementierung einer Lernplattform. Bachelorarbeit, Kiel, 2013. URL: https://lili.informatik.uni-kiel.de/static/theses/ausarbeitung_ggb.pdf (Abruf: 12. 09. 2013).

[RNH04] Reichert, Raimond ; Nievergelt, Jürg ; Hartmann, Werner: Programmieren mit Kara: Ein spielerischer Zugang zur Informatik. 2. Berlin : Springer, 2004 (eXamen.press).

[Sw07] SwissEduc (Hrsg.): Programmieren lernen mit Kara. Stand: 03.08.2007. URL: http://www.swisseduc.ch/informatik/karatojava/index.html, (Abruf: 13.09.2013).

App Entwicklung im Unterricht mit dem App Inventor

Matthias Ehmann, Carsten Müller

Didaktik der Informatik
Universität Bayreuth
Universitätsstraße 30
95447 Bayreuth
matthias.ehmann@uni-bayreuth.de
carsten.mueller@uni-bayreuth.de

Abstract: Graphische Programmierumgebungen minimieren die Syntaxproblematik, die insbesondere Einsteiger in das Thema Algorithmik oft vor zusätzliche Hürden stellt. Mit dem MIT App Inventor gibt es ein entsprechendes Werkzeug zur Entwicklung von Anwendungen auf mobilen Geräten. Der Artikel motiviert den Einsatz dieser Umgebung im (Anfangs-)Unterricht, stellt sie anhand zweier Beispiele vor und schildert Erfahrungen aus der Arbeit mit Schülern.

1 Werkzeugwahl bei der Umsetzung von Algorithmen im Anfangsunterricht

Der Bereich der Algorithmik ist für Schülerinnen und Schüler im Informatikunterricht oftmals einer der ersten Berührungspunkte mit dem Problemlöseprozess der Informatik.

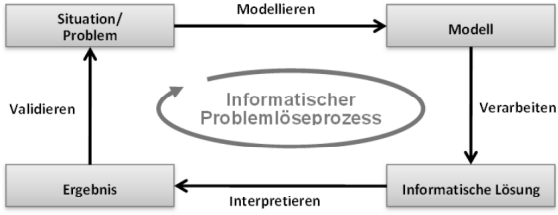


Abbildung 1: Informatischer Problemlöseprozess

Dabei wird deutlich, dass die Beschreibung eines Ablaufs gewissen formalen Regeln gehorchen muss, um die spätere Ausführbarkeit auch zu gewährleisten. Zur Ablaufmodellierung kommen häufig Struktogramme bzw. Programmablaufpläne zum Einsatz.

Die Durchführung des Verarbeitungsschrittes ist sehr eng mit der gewählten Programmierumgebung verknüpft. Es steht dabei eine Vielzahl von Sub- bzw.

Minilanguages in verschiedensten Ausprägungen zur Verfügung, die zu einer Einführung in die Umsetzung von Abläufen geeignet sind. Ein grundlegender Unterschied besteht in der Art der Eingabe der Abläufe. Auf der einen Seite stehen dabei textorientierte Werkzeuge, wie beispielsweise Robot Karol[1]. Andererseits gibt es graphisch orientierte Umgebungen, bei denen Ablaufbeschreibungen durch das Zusammensetzen von Puzzleteilen entstehen. Typische Vertreter sind hier Scratch[2] mit seinen Derivaten oder Squeak Etoys[3].

Beim Blick in den Unterrichtsalltag, beispielsweise einer siebten Klasse des bayerischen Gymnasiums, ergibt sich bei der Verwendung von Robot Karol folgende Situation [Sc09]: Die Lernenden werden in Eigenarbeitsphasen mit Syntaxproblemen und daraus resultierenden Fehlermeldung konfrontiert, die sie oft selbst nicht interpretieren bzw. beseitigen können. Folge sind häufige Rückfragen bei der Lehrkraft. Diese unterbricht darauf immer wieder den Arbeitsablaufs für die ganze Klasse um die aufgetretenen Schwierigkeiten im Klassenverband zu bündeln und zu klären. Damit kann in diesem Fall die Programmierumgebung selbst die Ursache für Hürden in der Umsetzung von Abläufen werden und demotivierend wirken. Es besteht die Gefahr, dass der Unterricht hier zum Programmiersprachenunterricht wird.

Graphische Entwicklungswerkzeuge können diese Problematik weitgehend eliminieren, da hier syntaktisch nicht verknüpfbare Elemente gar nicht zusammengefügt werden können. Damit tritt das Werkzeug mehr in den Hintergrund, der korrekten Semantik kann mehr Aufmerksamkeit geschenkt werden und die Konzepte der Algorithmik treten auch bei der Umsetzung verstärkt in den Vordergrund. [Mo11]

Ein Kritikpunkt an graphischen Entwicklungsumgebungen, die speziell auf das Kennenlernen von Algorithmik und Programmierung im unterrichtlichen Kontext abgestimmt sind, ist die fehlende Aufwärtskompatibilität zu "echten" Programmiersprachen und Umgebungen in der Softwareentwicklung. Die Kritik eines Schülers: „Das ist doch eher für den Kindergarten." – so gehört in einer neunten Realschulklasse bei der Verwendung von Scratch – geht in dieselbe Richtung. Aber bei der Softwareentwicklung gewinnen modellgetriebene Ansätze und Umgebungen immer mehr an Bedeutung. Dabei wird aus formalen Modellen automatisch lauffähige Software generiert. Auch bei Scratch ist das Erstellen des Programms vergleichbar mit der Erstellung eines Struktogramms. So sollten die beiden Kritikpunkte nicht als Aufforderung zur Rückkehr zu textorientierten Hochsprachen bzw. entsprechenden Sublanguages verstanden werden. Vielmehr sollten sie als Impuls aufgefasst werden, für den Unterricht auch in höheren Jahrgangsstufen nach geeigneten graphischen Programmierumgebungen Ausschau zu halten, die motivierend auf die Lernenden wirken.

[1] http://www.schule.bayern.de/karol/
[2] http://scratch.mit.edu/
[3] http://www.squeakland.org/

2 Der MIT App Inventor

Die Nutzung mobiler Kommunikationsgeräte wie Smartphones und Tablets gehört für viele Jugendliche heute zum Alltag. Sie benutzen täglich verschiedene Apps, um mit anderen zu kommunizieren, zu recherchieren oder zu spielen. Eine aktuelle, fundierte Bestätigung dafür findet man in der JIM-Studie 2012 [MFS12]. Sie bestätigt auch ein grundsätzliches Interesse Jugendlicher, sich mit technischen Geräten zu beschäftigen. Ein Blick hinter die Kulissen von mobilen Anwendungen kann einen motivierenden Kontext darstellen, um sich näher mit der Algorithmik im Unterricht zu befassen. Allerdings waren bisherige Entwicklungsumgebungen und Programmierwerkzeuge für mobile Anwendungen recht komplex und insbesondere für Einsteiger im Bereich der Algorithmik ungeeignet (beispielsweise Android SDK in Verbindung mit Eclipse oder Xcode für iPhone/iPad).

Google hat mit dem App Inventor eine graphische Programmierumgebung zur Entwicklung mobiler Anwendungen für die Android Plattform entwickelt. Mittlerweile wird das Projekt am MIT Media Lab weitergeführt[4]. Dort wird auch entsprechend aufbereitetes didaktisches Material zum Einsatz des App Inventors im Unterricht entwickelt. Beispiele aus der Praxis bestätigen den oben angesprochenen motivierenden Kontext, den die App Entwicklung mit dem App Inventor im Unterricht haben kann [Wol11].

Der App Inventor verfolgt ein ähnliches Grundkonzept wie Scratch. Abläufe werden durch das Aneinanderfügen elementarer Puzzleteile modelliert und somit auch direkt umgesetzt. Damit wird die eingangs angesprochene Syntaxproblematik weitgehend eliminiert. Im Unterricht bleibt es der Lehrkraft überlassen, ob mobile Geräte verwendet werden. Durch den in der Entwicklungsumgebung vorhandenen Emulator ist das Ausführen und Testen der erstellten Apps auch ohne reales Endgerät – teilweise sogar mit virtuellen Sensordaten – möglich.

Der Zugriff auf Sensordaten eines mobilen Gerätes und deren Verarbeitung macht den App Inventor auch aus Sicht des Physical Computings interessant und bietet damit fächerübergreifende Einsatzmöglichkeiten im Unterricht. So ergeben sich beispielsweise durch die Verwendung des GPS-Sensors Anknüpfungspunkte zu den Fächern Geographie, Mathematik und Physik.

Der App Inventor unterstützt parallele Abläufe und ist Ereignis-gesteuert.

2.1 Verwendung und Aufbau der Entwicklungsumgebung

Der App Inventor ist eine Webanwendung, benötigt aber, um Apps auf Android-Geräten live zu testen bzw. zu übertragen und für den Betrieb des Android Emulators lokal installierte Komponenten. Eine gute Übersicht über die Voraussetzungen und Vorbereitung eines Rechners zur Verwendung mit der Entwicklungsumgebung bietet die

[4] http://appinventor.mit.edu

App Inventor Webseite in der Kategorie „Explore – Setup“[5]. Nach der Installation der lokalen Komponenten ist die Umgebung unter http://beta.appinventor.mit.edu verfügbar.

Zur Verwendung von App Inventor ist beim Start das Anmelden über ein Google Konto nötig. Sowohl dies als auch die Notwendigkeit einer bestehenden Internetverbindung ist für die Arbeit im Unterricht problematisch. Eine Lösungsmöglichkeit, die den lokalen Betrieb erlaubt, wird in 2.2 vorgestellt.

Die Entwicklungsarbeit mit dem App Inventor ist zweigeteilt. Nach dem Erstellen eines neuen Projekts befindet man sich in der Designer-Komponente, die die Gestaltung der App-Oberfläche gestattet. Die eigentliche Programmierung von Abläufen erfolgt über den sog. Blocks Editor, der als Java Web Start Anwendung gestartet wird. Es stehen in verschiedenen Kategorien Strukturelemente zur Verfügung, die als Puzzleteile in den Programmbereich gezogen werden können.

Abbildung 2: Komponenten des App Inventors: Designer (oben), Blocks Editor

Zum Testen vom Apps während der Entwicklung kann aus dem Blocks Editor heraus ein Emulator gestartet werden („New emulator“). Die aktuell entwickelte App kann durch „Connect to Device …“ mit dem Emulator verbunden werden. Darüber hinaus ist es auch möglich ein reales Android-Gerät zu verwenden. Dieses kann entweder per USB-Kabel mit dem Entwicklungsrechner verbunden sein. Dazu sind auf dem Rechner die passenden Android Debug Bridge (ADB) Treiber für das mobile Gerät nötig und auf diesem muss die Option USB Debugging aktiviert sein. Die komfortablere Lösung zur Anbindung besteht in einer Wireless LAN Verbindung. Eine ausführliche Beschreibung aller Anschlussalternativen findet man auf der App Inventor Webseite[6].

[5] http://appinventor.mit.edu/explore/setup-mit-app-inventor.html
[6] http://appinventor.mit.edu/explore/setup-device.html

2.2 Lokale Verwendung des App Inventors

Wie bereits erwähnt, erscheint es sinnvoll für unterrichtliche Belange, eine lokale Installation der App Inventor Umgebung zu nutzen. Das App Inventor 4 All Projekt[7] stellt basierend auf den Quellen des MIT eine lokal lauffähige Server-Umgebung zur Verfügung, die Anpassungen für Windows enthält.

Die Installation gestaltet sich einfach. Das heruntergeladene Archiv muss nur entpackt werden. Zu beachten ist, dass die JAVA_HOME Umgebungsvariable auf ein installiertes Java Development Kit zeigt und der Pfad das bin-Verzeichnis des JDK enthält.

Zum Starten der Server-Komponenten werden nacheinander die AppEngine (Startskript StartAI.cmd im Ordner AppEngine) und der BuildServer[8] (Startskript launch-buildserver.cmd bzw. launch-buildserver32.cmd im Ordner BuildServer) in Betrieb genommen. Die lokale Entwicklungsumgebung steht dann unter http://localhost:8888 zur Verfügung. Der Startprozess kann natürlich in ein einzelnes Startskript ausgelagert werden.

Mit dieser Konfiguration auf jedem Schülerrechner ist es möglich ohne Anlegen von Benutzerkonten lokal zu arbeiten. Zu beachten ist, dass nach Beenden der Serverdienste alle Projekte gelöscht werden. Allerdings ist aus dem Frontend der Export der Projekte durch den Benutzer jederzeit möglich.

3 App Inventor im Unterricht

Im Folgenden wird anhand von zwei Projektbeispielen der Umgang mit der App Inventor Umgebung gezeigt.[9] Die Projekte sind so gewählt, dass Sie als komplexere Beispiele direkt nach der Einführung der Kontrollstrukturen im Unterricht verwendet werden können. Es ist möglich, dass in der Einführungsphase die algorithmischen Grundstrukturen mit einer anderen Programmierumgebung umgesetzt wurden und der App Inventor als neues Werkzeug dient. Die Autoren haben mit Schülern einer 8. Klasse des bayerischen Gymnasiums Erfahrung gesammelt, die im Vorjahr mit Scratch bzw. Robot Karol gearbeitet haben und jetzt im Rahmen eines Profilkurses mit dem App Inventor arbeiten.

Insbesondere das erste Beispiel, das hier auch ausführlicher beschreiben wird, ist auch zum Einstieg in die Algorithmik geeignet. Hieran können nach und nach die Kontrollstrukturen motiviert und ihre Umsetzung vorgenommen werden. Ein Teilnehmer an einer Lehrerfortbildung, die von den Autoren angeboten wurde, hat dieses Konzept mit einer 7. Klasse an einem Bayreuther Gymnasium bereits erfolgreich umgesetzt.

---

[7] http://sourceforge.net/projects/ai4a-configs/

[8] In der aktuellen Version 1.4.7 ist im Startskript fälschlicherweise die Bibliothek Guava mit der Version 11.0.1 eingebunden, obwohl Version 14.0.1 mitgeliefert wird. Hier muss lediglich die Versionsnummer im Skript ersetzt werden.

[9] Die Beispielprojekte sind unter http://did.inf.uni-bayreuth.de/infos2013/appinventor verfügbar.

3.1 Zeichen App

Das erste Szenario beschäftigt sich mit der Erstellung einer einfachen Zeichen App.[10] Dabei soll der Finger als Malwerkzeug genutzt werden. Einstellmöglichkeiten betreffen Strichstärke, Zeichen- und Hintergrundfarbe. Das Userinterface wird mit dem Designer erstellt, die Logik des Programmablaufs wird im Blocks Editor implementiert. Durch die Ereignis-gesteuerte Programmierung ergibt sich bei Modellierung und Umsetzung der Abläufe eine übersichtliche Grundstruktur.

Meilenstein 1

In einem ersten Meilenstein soll die Oberfläche mit einem Zeichenbereich und zwei Buttons ausgestattet werden, die zum Setzen der Hintergrundfarbe dienen.

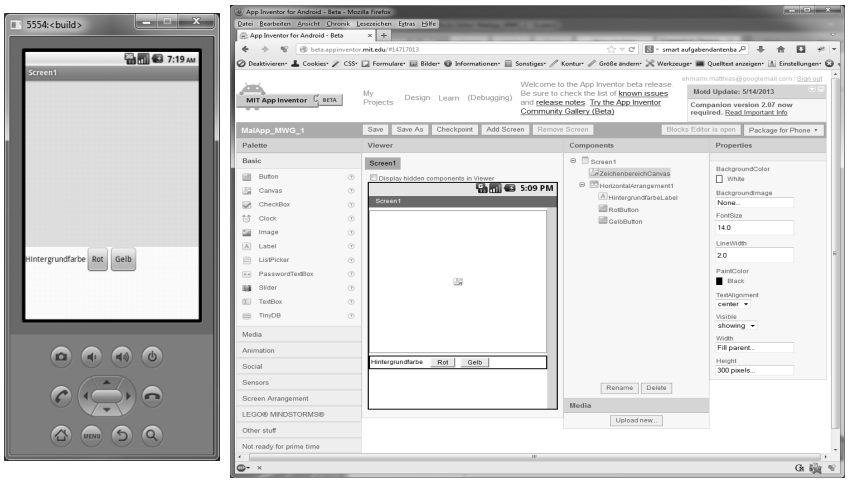


Abbildung 3: Oberfläche im Emulator und im Designer

Beim Erstellen der Benutzeroberfläche lernen die Schüler die Bedeutung der Attribute der verwendeten Oberflächenelemente kennen. Hierbei ist keine ausführliche Erläuterung nötig, die meisten Einstellungen sind selbsterklärend. Bei der Benennung der Oberflächenobjekte sollte auf treffende Namen geachtet werden. Dabei hat es sich als sinnvoll erwiesen, zu einem individuellen Teil auch noch den Klassennamen hinzuzufügen (etwa RotButton, GelbButton). Damit wird später das Auffinden und Identifizieren der Objekte bei der Programmierung erleichtert.

Bei der Erstellung der Programmlogik im Blocks Editor zeigt sich die intuitive Umsetzung durch die Ereignissteuerung. So ist die von Schülern selbst gefundene Versprachlichung des Ablaufs „Wenn RotButton angeklickt wird, soll der Hintergrund

[10] Der Kontext des Beispiels ist angelehnt an das PaintPot Projekt aus dem App Inventor Tutorial http://beta.appinventor.mit.edu/learn/tutorials/paintpot/paintpot-part1.html und erweitert diesen.

des ZeichenbereichCanvas rot werden. Wenn GelbButton angeklickt wird, soll der Hintergrund des ZeichenbereichCanvas gelb werden.“ direkt umsetzbar. Das Setzen der Hintergrundfarbe schafft Anknüpfungspunkte an Vorwissen der Schüler aus der Objektorientierung. Sie begegenen der Punktnotation und Methodenaufrufen wieder.[11]

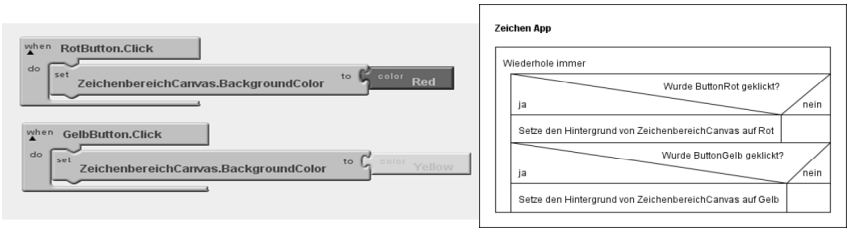


Abbildung 4: Ereignissteuerung im App Inventor, Struktogramm einer sequentiellen Umsetzung

Hierbei wird auch der Vorteil der Ereignissteuerung und die damit einhergehende parallele Verarbeitung beim App Inventor deutlich. Die beiden Ereignisse stehen grundsätzlich in keiner Beziehung zueinander. Ohne Ereignissteuerung würde zur Umsetzung eine unendliche Wiederholung mit enthaltenen Verzweigungen verwendet. Diese Umsetzung ist vom Denken der Schüler wesentlich weiter entfernt als die vom App Inventor unterstützte Lösung. Hier ist ersichtlich, dass die Modellierung natürlich auch von dem verwendeten Werkzeug zur Umsetzung abhängig ist.

Meilenstein 2

Das Canvas-Objekt für den Zeichenbereich besitzt Event-Handler für verschiedene Touch-Events. Mit dem Dragged-Event in Verbindung mit der Methode DrawCircle des Canvas-Objekts kann das Zeichnen realisiert werden. Zusätzlich bietet sich hier die Möglichkeit, die Pinselstärke mit einem Schieberegler zu wählen.

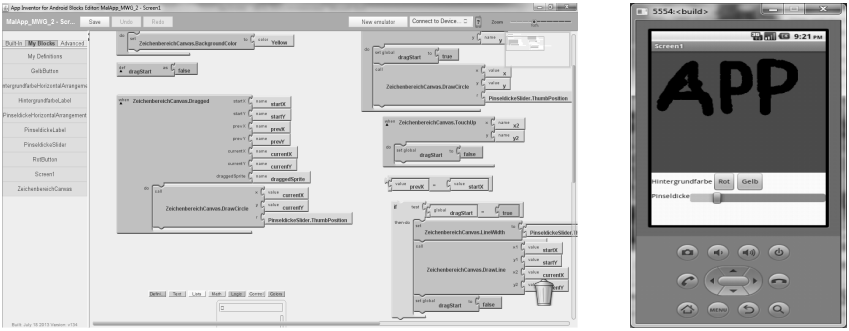


Abbildung 5: Zeichnen auf dem Canvas-Objekt mit variabler Pinseldicke

[11] Im Fall des bayerischen Gymnasiums wurden diese Inhalte in der 6. Jahrgangsstufe behandelt: http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/index.php?StoryID=26433

Der Event-Handler und die Methode zum Zeichnen haben in diesem Fall mehrere Parameter. Auch hier kann auf Vorwissen zurückgegriffen werden.Fehler: Referenz nicht gefunden An dieser Stelle bietet es sich an, noch genauer zu thematisieren, dass Event-Handler Methoden sind, die automatisch beim Eintreten des Ereignisses aufgerufen werden. Dadurch wird die Verwendung der Parameter auch beim Aufruf der Methode DrawCircle deutlicher.

Die gefundene Lösung ist nicht ganz befriedigend. Das Dragged-Ereignis wird erst nach dem Zurücklegen einer gewissen Strecke des Fingers auf dem Bildschirm aktiv. So wird auch erst „verspätet“ gezeichnet. Je nach Kenntnisstand der Schüler lässt sich dieses Problem lösen. Dazu ist die Verwendung von Variablen nötig, um nachträglich den Pfad vom Anfagspunkt des TouchDown-Events bis zum Beginn des Drag-Events zu zeichnen.

Meilenstein 3

Die Auswahl der Hintergrundfarben über einzelne Buttons in der Oberfläche ist nicht sonderlich sinnvoll. Viel geschickter kann dies mit einer Auswahlliste, einem sog. ListPicker-Objekt, erreicht werden. Dabei lernen die Schüler eine einfache Listenstruktur kennen. Das Setzen der Hintergrundfarbe wird durch Auswerten des gewählten Listeneintrags in einer Verzweigungsstruktur im AfterPicking-Event-Handler vorgenommen. Das Vorgehen lässt sich auch auf die Auswahl einer Zeichenfarbe anwenden. Bei der Durchführung in dem erwähnten Profilkurs in der 8. Jahrgangsstufe kamen die Schüler selbst auf die Idee noch die Hintergrundfarbe des Auswahlelements auf die gewählte Farbe zu setzen.

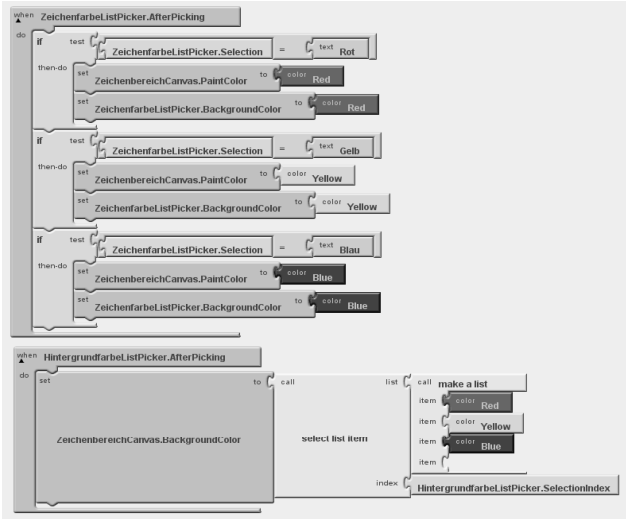


Abbildung 6: Auswertung einer Listenauswahl mit Verzweigungen und mit Listenoperationen

Für umfangreiche Listen wird die manuelle Erstellung einer Auswahlstruktur schnell zu aufwändig und unübersichtlich. Je nach Kenntnisstand der Schüler und Zielsetzung können hier weitere Listenoperationen thematisiert werden, die die Lösung eleganter machen.

Erweiterungen

Das Thema bietet einige Erweiterungsmöglichkeiten: Numerische Anzeige der Pinseldicke in einem Textfeld, Festlegen der Transparenz bei der Zeichenfarbe mit einem Schieberegler, Export der Zeichnung als Grafik. Dabei kommen viele der schon kennengelernten Strukturen wieder vor und sorgen für weitere Vertiefung.

3.2 Kugelspiellabyrinth App

Beim vorherigen Beispiel bestehen keine großen Unterschiede bezüglich der Ausführung im Emulator oder auf einem Android-Gerät. Das folgende Projekt verwendet den Lagesensor eines Mobilgeräts zur Steuerung einer Kugel durch ein Labyrinth. Eine Variante für den Emulator durch Ziehen an der Kugel ist möglich.Fehler: Referenz nicht gefunden

In einem ersten Schritt kann zunächst das Bewegen der Kugel mit dem Lagesensor umgesetzt werden. Bei der Kugel handelt es sich um ein Objekt der Klasse Ball, das in ein Canvas-Objekt eingefügt wird. Die Steuerung ist durch den Event-Handler OrientationChanged des Lagesensors möglich. Die Parameter pitch und roll repräsentieren die vertikale bzw. horizontale Neigung des Geräts.

Die Labyrinth-Grafik kann in einem beliebigen Grafikprogramm oder sogar mit der Zeichen App erstellt werden. Sie wird als Hintergrund für das Canvas-Objekt verwendet.

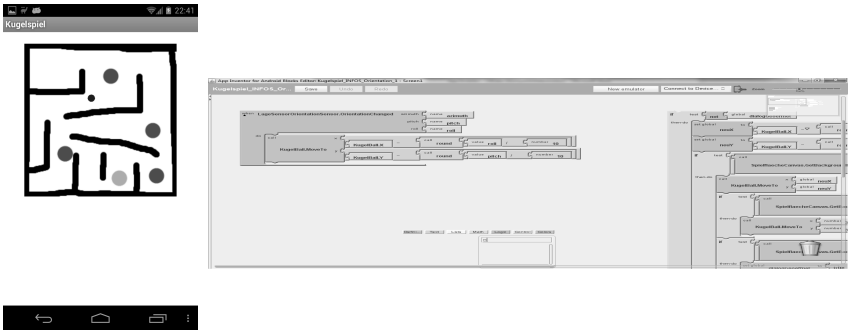


Abbildung 7: Ansicht der Kugelspiellabyrinth App, Steuerung der Kugel durch den Lagesensor

Der weitere Ausbau des Beispiels besteht darin, die Farbe hinter der Kugel auszuwerten. Je nach Farbwert wird sich die Kugel bewegen, an den Labyrinthwänden hängenbleiben,

in den roten Löchern verschwinden und an einem festzulegenden Startpunkt wieder erscheinen oder zum Spielende im grünen Loch landen. Zur Umsetzung ist es wegen der unterschiedlichen Alternativen sinnvoll, zunächst in einem Struktogramm den Gesamtablauf festzulegen und dann die Umsetzung anzugehen.

3.3 Erfahrungen

In der bisherigen Arbeit mit Schülern hat sich gezeigt, dass der Kontext App Entwicklung eine große Motivation ausübt. Förderlich ist hier auch die Verwendung realer Geräte. Die Schüler hatten die Möglichkeit eigene Smartphones bzw. Tablets in den Unterricht mitzubringen.[12] Die englischsprachige Umgebung hat nur anfänglich kleinere Probleme gemacht. Die Schüler waren recht schnell an den Umgang gewohnt und es gab nur noch bei neuen Puzzleteilen gelegentlich Nachfragen, die hauptsächlich die Bedeutung von Parametern von Event-Handlern betrafen. Die Schüler zeigten eine große Selbstständigkeit bei der Arbeit. Sie haben viele nötige Programmelemente selbst gefunden. Dies wurde auch in einem einführenden Workshop für Schüler der 8. Klasse in einem Profilkurs deutlich. Hier haben die Schüler zusätzliche Features bei der Zeichen App eigenständig geplant und umgesetzt und waren über zweieinhalb Stunden konzentriert bei der Sache.

Der eingangs erwähnte Problemlöseprozess verändert seine Struktur etwas. Wurde ein Struktogrammen bei der Verwendung textorientierter Programmierumgebungen als sichtbarer Abschluss der Modellierungsphase gewertet, erübrigt sich bei der Verwendung des App Inventors das manuelle Zeichnen meistens. Das bedeutet aber nicht, dass die Modellierung auf der Strecke bleibt bzw. ein „Drauflosprogrammieren“ gefördert werden soll. Ein Überlegungs- und Strukturierungsprozess findet natürlich statt, allerdings entsteht ein aus den Ergebnissen resultierendes Programm, das einem Struktogramm gleicht. Der Fokus liegt auf der Strukturierung von Abläufen, die Programmierung tritt eher in den Hintergrund.

Literaturverzeichnis

[MFS12] Medienpädagogischer Forschungsverband Südwest (Hrsg.): JIM 2012 – Jugend, Information, (Multi-)Media – Basisstudie zum Medienumgang 12- bis 19-Jähriger in Deutschland, 2012, Stuttgart.

[Mo11] Modrow, E.: Visuelle Programmierung – oder: Was lernt man aus Syntaxfehlern?, In: Thomas, M.: .: Informatik in Bildung und Beruf – 14. GI-Fachtagung „Informatik und Schule – INFOS 2011“, Lecture Notes in Informatics, P-189 GI, 2011, Münster.

[Sc09] Schatz, B.: Scratch, die bessere Alternative zu Robot Karol? Ein Vergleich in der siebten Jahrgangsstufe in Theorie und Praxis – Schriftliche Hausarbeit zur Ersten Staatsprüfung, Universität Bayreuth, 2009.

[Wol11] Wolber, D.: App Inventor and Real-World Motivation, SIGCSE'11, 2011, Dallas.

[12] Bei der großen Vielfalt von mitgebrachten Geräten ist die Installation eines ADB-Treiber Komplettpakets sinnvoll: http://download.clockworkmod.com/test/UniversalAdbDriverSetup6.msi

Agiler Projektunterricht in der Schulinformatik

Timo Göttel

Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg
tgoettel@acm.org

Ralf Romeike

Universität Potsdam
August-Bebel-Str. 89
14482 Potsdam
romeike@cs.uni-potsdam.de

Abstract: Gängige Prozessmodelle für Schulsoftwareprojekte orientieren sich bisher vor allem am Wasserfallmodell. In der professionellen Softwareentwicklung werden inzwischen agile Methoden erfolgversprechend eingesetzt, um Probleme, die sich aus dem Wasserfallmodell ergeben, zu umgehen. Dieser Beitrag beschreibt leicht umsetzbare Praktiken für die Gestaltung von Informatikunterrichtsprojekten, die sich an den agilen Methoden orientieren.

1 Agile Methoden in der Praxis

In der professionellen Softwareentwicklung hat sich in den letzten Jahren herausgestellt, dass sequentielle Vorgehensmodelle wie das Wasserfallmodel häufig zu geringer Softwarequalität, größeren Verzögerungen bei der Auslieferung und wenig Kundennähe führen (vgl. [LB03]). Ähnliche Probleme treten auch in Softwareprojekten im Informatikunterricht auf, die sich ebenfalls meist am Wasserfallmodell orientieren. So wird häufig von Problemen berichtet, die sich in unvollendeten Projekten, schlechter Zeiteinteilung und geringer Motivation für Tests bzw. die Dokumentation widerspiegeln (vgl. z. B. [Ha06] und [Ko92]). In der professionellen Softwareentwicklung werden zunehmend agile Methoden als Lösungsansatz gewählt, da sie ein flexibles Projektmanagement versprechen, das sich auf Interaktion und kurze iterative Entwicklungsphasen festlegt. Agile Methoden entsprechend des agilen Manifests[1] beruhen auf Werten und Praktiken, die auch für den Schuleinsatz angebracht erscheinen, die bekanntesten Umsetzungen sind Scrum [SB01] und Extreme Programming [BA04]. Auf der WiPSCE 2012 stellten wir ein auf agilen Methoden basierendes Modell für die Projektmethode im Informatikunterricht vor [RG12], welches im Folgenden skizziert sowie in seinen praxisrelevanten Ausprägungen beschrieben wird. Das Modell stellt so für die Unterrichtspraxis einen leicht einsetzbaren Satz an Praktiken bereit, der es erlaubt, im Unterricht agile kommunikationsfördernde Prozesse einzusetzen, die schnelle Projekterfolge ermöglichen. Gleichzeitig besteht durch die Referenzen auf aktuelle Vorgehensmodelle der Softwareentwicklung (hier sei [PM08] als vertiefende Lektüre zu Softwareprojekten empfohlen), die dynamische Prozesse und soziale Interaktion hervorheben, die Hoffnung, ein attraktives Bild professioneller Softwareentwicklung und damit der Informatik, zu vermitteln.

[1] http://agilemanifesto.org/iso/de/, zuletzt besucht am 23.08.2013

Das agile Manifest auf den Schulkontext übertragen, fokussiert sich demnach darauf...

1. Schüler und ihre Interaktionen zu unterstützen
2. schnelle Erfolge und funktionsfähige Software zu ermöglichen
3. gemeinsam auf Ziele hinzuarbeiten statt Vorgaben abzuarbeiten
4. Projekte am Lernfortschritt auszurichten statt festgelegten Plänen zu folgen.

Das folgende Kapitel stellt die einzelnen Praktiken vor, gibt praktische Anleitungen zum Einsatz im Informatikunterricht und präsentiert Vorlagen für nötige Arbeitsmaterialien.

2 Agiles Vorgehen für Softwareprojekte im Informatikunterricht

Das agile Vorgehensmodell für den Informatikunterricht sieht mehrere iterativ wiederkehrende Projektphasen vor (*Vorbereitung*, *Ideenfindung*, *User Stories*, *Planning Poker*, *Tasks*, *Miniprojekt*), die im Einzelnen nachfolgend erläutert werden und im Überblick in Abb. 1 dargestellt sind. Die Darstellung bezieht sich auf die Prozesse und Praktiken der agilen Methoden. Zusätzliche wichtige pädagogische Aspekte, wie z. B. die Aktivierung von Schülerinnen und Schülern oder der Unterstützung von Aushandlungsprozessen, werden nicht berücksichtigt.

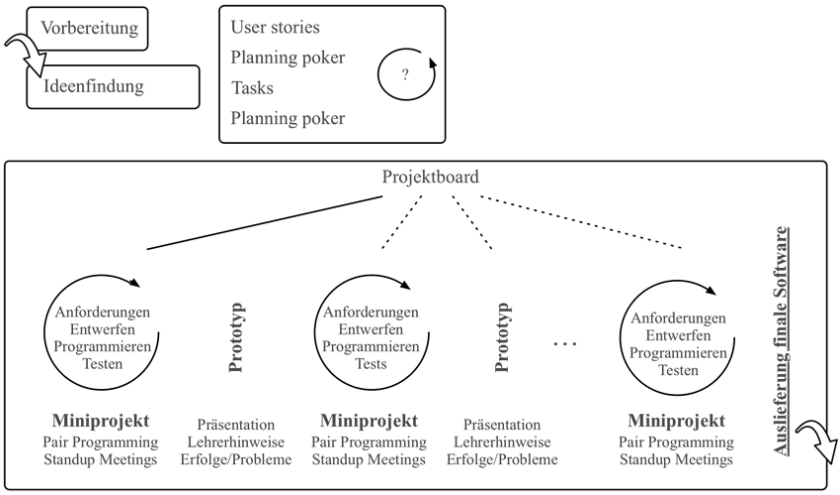


Abbildung 1: Agiles Vorgehensmodell für den Informatikunterricht

Das Vorgehen bzw. die Praktiken beinhalten klare Anweisungen, die von Schülerinnen und Schülern leicht zu befolgen sind. Trotzdem ist es empfehlenswert, die einzelnen

Praktiken zu üben bevor sie in einem vollwertigen Projektkontext eingesetzt werden bzw. den Prozess so anzuleiten, dass die Praktiken aufeinander aufbauend verwendet werden und sich so der Projektgedanke entfaltet.

2.1 Vorbereitung

Eine gute Vorbereitung stellt sicher, dass die erforderlichen Kompetenzen und Fähigkeiten der Schülerinnen und Schüler für Softwareprojekte vorhanden sind oder während des Projekts erarbeitet werden können (z. B. durch ergänzende Unterrichtsmaterialien). Darüber hinaus muss gewährleistet werden, dass eine stabile Infrastruktur vorhanden ist (Programmierumgebung, Netzwerkzugriff, Plattform zur Kommunikation und Medienaustausch).

2.2 Ideenfindung

Der nächste Schritt ist die Ideenfindung. Dabei sollten sich die Schülerinnen und Schüler durch eine ausgewählte und vorher einstudierte Methodik (z. B. Brainstorming, Interviews, Beobachtungen) sich über die Ziele für das Projekt Gedanken machen. Dabei wird darauf geachtet, dass besonders die Sicht des Kunden in den Vordergrund gestellt wird. Hierzu können Schülerinnen und Schüler bestimmt werden, die als Kunden auftreten und versuchen, den Entwicklern ihre Wünsche und Standpunkte zu vermitteln. In einigen Fällen können sich die Schülerinnen und Schüler hiervon überfordert fühlen. Dann ist es hilfreich, wenn auf vorbereitete Karteikarten mit Kundenwünschen oder Zielen zurückgegriffen werden kann.

2.3 User Stories

User Stories beschreiben in kurzen und prägnanten Sätzen einzelne geplante Tätigkeiten oder Ereignisse, die der Endnutzer später ausführen oder erleben kann. Die *User Stories* werden aus der Sicht des Benutzers geschrieben und sollen auch von Nicht-Entwicklern verstanden werden. Zusätzlich werden Prioritäten für die einzelnen User Stories angeben. Diese werden von 10 (sehr wichtig) bis 50 (eher unwichtig) in 10er-Schritten unterteilt. Ist die Priorität auf 10 gesetzt, bedeutet dies, dass die *User Story* bei Abgabe des Projektes auf jeden Fall implementiert sein soll, während eine 50 bedeutet, dass diese *User Story* auch in einer späteren Version nachgeliefert werden könnte.

Zur Entwicklung von *User Stories* empfehlen sich Rollenspiele, um den Schülerinnen und Schülern eine die Analyse aus Sicht des Nutzers zu vereinfachen. Die Regeln hierfür sind denkbar einfach: In der Rolle des Computers wird auf die Wünsche des Nutzers reagiert. Der Nutzer muss dazu dem Computer instruieren, was er durchführen möchte. Die Reaktion des Computers verbalisiert die Handlungen bzw. Folgen und beschreibt, was zu sehen ist. Häufig führt dies dazu, dass der Nutzer seine Anforderungen und Wünsche neu bzw. anders spezifizieren muss, damit der Computer dies im Sinne des Nutzers richtig ausführt. So entstehen im Zwiegespräch Computer – Nutzer Skizzen, die zu elementaren Funktionen führen, die man dann als *User Story* festhalten kann.

Während in der professionellen Softwareentwicklung die Prioritäten durch den Kunden vergeben werden, empfiehlt es sich, im Schulkontext dem Entwicklerteam diese Aufgabe zu überlassen.

Eine *User Story* wird auf einer Karteikarte festgehalten und besitzt einen Titel, eine Beschreibung sowie einen Platzhalter für Priorität und Schätzwert für Arbeitsaufwand (siehe Abb. 2).

Eine *User Story*...

- beschreibt eine Aufgabe, welche bewältigt werden muss
- beschreibt die Sicht des Nutzers
- ist kurz, enthält nicht mehr als 3 Sätze
- benutzt keine Fachausdrücke
- legt keine Werkzeuge oder Technologien fest

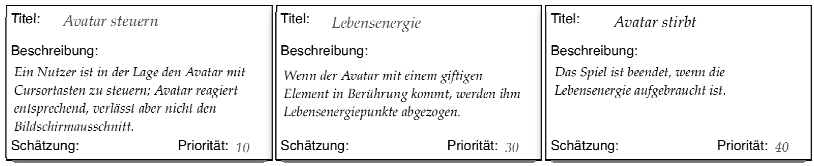


Abbildung 2: User Stories

2.4 *Planning Poker*

Planning Poker ist eine spielerische Methode, den Teilnehmern zu helfen, den benötigten Zeitaufwand für *User Stories* und *Tasks* (siehe 2.5) einzuschätzen und definiert dazu eine nachvollziehbare und klar strukturierte Vorgehensweise. Jeder Teilnehmer erhält einen Satz Karten, die mögliche Zeitaufwände darstellen. Außerdem gibt es Sonderkarten, wie z. B. „Benötige eine Pause“ (Pause), „Bereits erledigt“ (0 min.), „Nicht umsetzbar“ (2400 min.) oder „Fehlende Informationen“ (???). Ein möglicher Kartensatz für den Informatikunterricht ist in Abb. 3 dargestellt, PDF-Druckvorlagen stehen zur Verfügung[2].

Jede Spielrunde des *Planning Poker* ist einer *User Story* bzw. einem *Tasks* gewidmet. Alle Teilnehmer legen eine verdeckte Karte auf den Tisch, welche angibt, wie viel Zeit sie für die Bearbeitung der Aufgabe als notwendig erachten. Anschließend werden alle Karten eingesammelt und der Durchschnittswert berechnet. Gibt es große Ausreißer, so wird nach den Gründen gefragt und diskutiert. So werden bspw. diejenigen mit dem größten und kleinen Schätzwert aufgefordert, ihre Einschätzung zu begründen. Dies soll mögliche Probleme oder Missverständnisse beim Bearbeiten dieser Aufgabe beheben sowie die Kommunikation in der Gruppe anregen. Ist der Zeitaufwand bestimmt, wird

[2] http://informatikdidaktik.de/agil

dieser auf die Karte der *User Story* geschrieben. Ergab sich keine Einigung, wird die *User Story* oder der *Task* zurückgestellt und zu einem späteren Zeitpunkt nochmals betrachtet. Schülerinnen und Schüler können durch die fehlende Praxiserfahrung anfangs noch recht schlecht einschätzen, welche Probleme auftauchen könnten oder wie umfangreich eine Aufgabe wirklich ist. Durch das *Planning Poker* wird Runde für Runde ein Lernprozess gefördert, der alle Schüler herausfordert und sie dazu bringt, mehr auf die Details zu achten, sowie ihre Einschätzung verteidigen und begründen zu können. Daher ist davon auszugehen, dass zunächst zurückgelegte *User Stories* bzw. *Tasks* zu einem späteren Zeitpunkt besser eingeschätzt werden können.

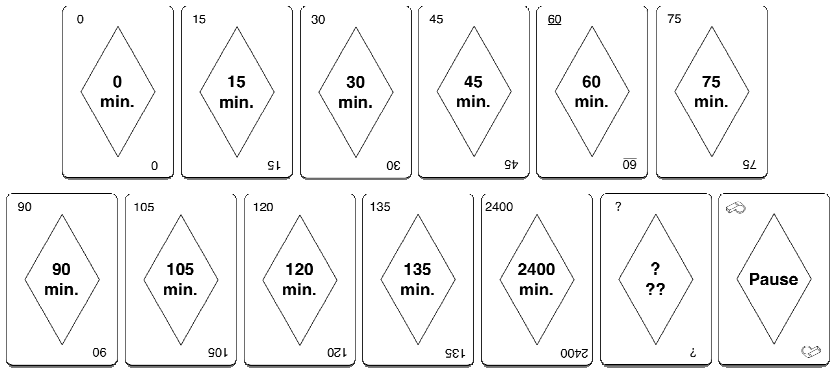


Abbildung 3: Kartensatz für das *Planning Poker* im Informatikunterricht.

Hierbei kann es nötig sein, dass Prozesse zur Einigung geleitet und organisatorische Abläufe überwacht bzw. eingehalten werden müssen. Für das *Planning Poker* wird daher ein Teamleiter benötigt. Wünschenswert ist es hierbei, dass in diesem Schritt eine Schülerin oder ein Schüler die Organisation und Moderation übernimmt. Diese Aufgabe kann reihum wiedervergeben werden, sodass letztlich alle Teilnehmer im Laufe des Projekts einmal in dieser Position tätig waren.

2.4 *Tasks*

Nachdem das *Planning Poker* zu den *User Stories* abgeschlossen ist, werden die einzelnen *User Stories* in *Tasks* aufgeteilt, die programmiertechnisch gelöst werden müssen, damit eine *User Story* umgesetzt werden kann. Eine *User Story* ist demnach eine Sammlung von *Tasks*. Die *Tasks* sind knapp verfasste Arbeitspakete, die von einem Paar (siehe 2.5, Programmieren im Paar) zu implementieren sind. Demzufolge müssen die Schülerinnen und Schüler nun die Perspektive des Entwicklers einnehmen. Hierbei ist durch die Lehrkraft darauf zu achten, dass erfahrene Programmierer und unsichere bzw. unerfahrene Teammitglieder in den Gruppenprozess durch Diskussionen einbezogen werden. So können z. B. erfahrene Mitglieder gebeten werden, Sachverhalte nochmals zu erklären oder Argumente zu nennen, die ihre Aussagen stützen. Hier kann es auch helfen, wenn

für typische *User Stories* entsprechende *Tasks* präsentiert werden (vgl. Abb. 2 und 4 für die *User Story* „Lebensenergie“).

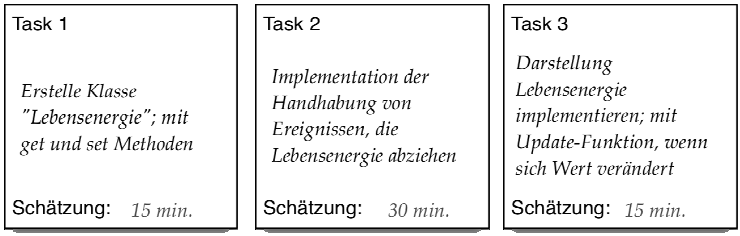


Abbildung 4: *Tasks* für *User Story* Lebensenergie.

2.5 Miniprojekt

Ein Miniprojekt stellt eine Iteration dar, deren Dauer (Anzahl Einheiten) vorher festgelegt wird und einen funktionierenden Prototypen zum Ziel hat. Anhand der Schätzwerte für den Zeitaufwand werden alle *User Stories* inklusive ihrer *Tasks* (geordnet nach Priorität), die zeitlich in der anstehenden Iteration möglich sind, ausgewählt und am Projektboard (siehe Abb. 5) befestigt.

Am Projektboard wird der Fortschritt des Projekts visualisiert: Die Paare nehmen sich einen *Task*, den sie bearbeiten wollen und verschieben diesen in das Feld „In Bearbeitung“. Sobald ein *Tasks* implementiert wurde, wird dieser in das Feld „Fertig“ verschoben und das entsprechende Team kann entweder einen neuen *Task* in Bearbeitung nehmen oder unter Umständen anderen Paaren Hilfe anbieten, falls am Projektboard ersichtlich wird, dass ein *Task* schon sehr lange in Bearbeitung ist.

Zusätzlich ist auf dem Projektboard ein *Burn Down Chart* angezeichnet, das die Gesamtzeit des Projekts darstellt und die noch zu erledigenden Arbeitspakete visualisiert: Nach jeder Einheit wird dort die Summe der noch offenen *Tasks* abgetragen. So entsteht eine Kurve, die sich bei guter Planung und guter Teamarbeit bis zum Ende des Projekts bei Null einfindet.

Zu Beginn jeder Einheit eines Miniprojekts wird ein *Standup-Meeting* abgehalten. Dies dient dazu, dass jeder Schüler des Teams kurz und prägnant abgeschlossene Aufgaben mitteilt, über Probleme und Lösungswege berichtet und die anstehenden Aufgaben der aktuellen Einheit nennt. Um eine Situation zu schaffen, die dazu anregt, sich kurz zu fassen, wird dieses Meeting im Stehen vor dem Projektboard abgehalten und sollte nicht länger als fünf bis zehn Minuten dauern.

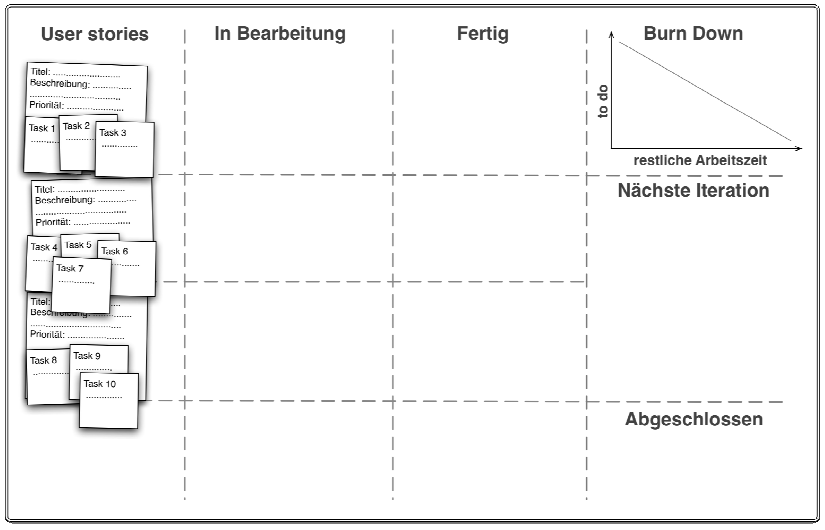


Abbildung 5: Projektboard für ein Miniprojekt (Iteration).

Sämtliche Implementationen werden nach den Regeln der agilen Praktik für das Programmieren im Paar durchgeführt. Hierzu übernimmt ein Partner, der *Driver*, die Tastatur und Maus und verfasst den Quelltext. Dabei ist er angehalten, sämtliche Entscheidungen und Absichten seinem Partner, dem *Navigator*, mündlich mitzuteilen. Dieser ist dafür verantwortlich, dass mögliche Missinterpretationen angesprochen, Quelltext hinterfragt und elegantere Lösungen in Betracht gezogen werden. Gleichzeitig sorgt der *Navigator* dafür, dass das große Ganze nicht aus den Augen verloren wird.

Jede Entwicklungsphase besteht aus vier Elementen, die von den Paaren umgesetzt werden, um einen *Task* zu implementieren: Zunächst müssen Anforderungen erhoben werden, dann werden Lösungswege entworfen, programmiert und getestet.

Ein Miniprojekt (eine Iteration), endet mit einer Präsentation des Prototyps vor allen Schülerinnen und Schülern und der Lehrkraft. Erfolge und Probleme sollen von den Präsentierenden benannt werden und sind auch von der Lehrkraft aufzugreifen. In dieser Situation können auch nötige Hinweise und Lerneinheiten untergebracht werden.

3 Zusammenfassung

Agiler Projektunterricht sieht eine Abfolge von Schritten vor, die sich im Einzelnen leicht erproben lassen und durch ihren repetitiven Charakter schnell erlernt werden und in ein Prozesswissen übergehen. So ist es möglich, agile Softwareprojekte im Schulunterricht umzusetzen, da sich Routinen entwickeln von denen sowohl Anfänger als auch Schülerinnen und Schüler mit Programmiervorerfahrung profitieren.

Literaturverzeichnis

[BA04] Beck, K.; Andres, C.: Extreme Programming Explained: Embrace Change (2nd Edition). 2004. Addison-Wesley.

[Ha06] Hartmann, W.; Näf, M.; Reichert, R.: *Informatikunterricht planen und durchführen.* 2006. Springer.

[Ko92] Koerber, B. 1992. Die Angst des Lehrers vorm Projektunterricht. *LOG IN 12,* 5/6, 3.

[LB03] Larman, C.; Basili, V.R.: Iterative and Incremental Development: A Brief History. Computer 36, S. 47-56. 2003.

[PM08] Pilone, D.; Miles, R.: *Softwareentwicklung von Kopf bis Fuß: Ein Buch zum Mitmachen und Verstehen.* 2008. O'Reilly Media, Inc.

[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, Hamburg 2012; S. 48-57; ACM.

[SB01] Schwaber, K.; Beedle, M.: Agile Software Development with Scrum. 2001. Prentice Hall.

E-Learning in der Schule – die Rolle des Lehrers als E-Teacher

Sven Hofmann,
Steffen Friedrich, Andrea Lißner, Sindy Riebeck, Michael Rudolph

Fakultät Informatik
AG „Didaktik der Informatik / Lehrerbildung“
Nöthnitzer Straße 46
01062 Dresden
Sven.Hofmann@tu-dresden.de

Abstract: Im Rahmen zweier Projekte der TU Dresden wurden E-Learning-Szenarien für Schülerinnen und Schüler Allgemeinbildender und Beruflicher Gymnasien Sachsens entwickelt und an ausgewählten Schulen erprobt, welche an der Schnittstelle vom Gymnasium zum Anfangsstudium angelegt sind. Über 1300 Schülerinnen und Schüler der Sekundarstufe II waren im Projekt „UnIbELT“ von 2009 bis 2012 aktiv. Das Nachfolgeprojekt „KoSEL“ bringt seit 2013 u.a. eine Erweiterung der Zielgruppe auf Schüler der 10. Klassen und auf Berufliche Gymnasien mit sich. Hierbei kommen neue Methoden zur Kompetenzentwicklung durch den Einsatz medialer Unterstützungssysteme wie 3D-Welten und virtuelle Klassenräume sowie neuer Wege in der Lernerfolgskontrolle durch E-Portfolios zur Anwendung. Das Einsatzszenarium dieser E-Learning-Kurse umfasst die drei Rollen des Lernenden (Schüler), des Kursbetreuers (Lehrer) sowie des Tutors (Mitarbeiter der TU Dresden). In beiden Projekten wurden bis dato mehr als 100 Kurse an den Schulen erprobt. Daraus sind auch Erfahrungen erwachsen, welche Rolle die Lehrerinnen und Lehrer an den Schulen wahrnehmen - was sie als Kursbetreuer und als Kursautoren leisten können.

1 E-Learning in der Schule

Schülerinnen und Schüler, die nach dem Abitur ein Studium beginnen, treffen auf eine Hochschullandschaft, in der Lernplattformen weitgehend etabliert sind. Die Notwendigkeit, das eigene Studium selbständig zu organisieren und dazu webbasierte Portale zu nutzen sowie Lernprozesse eigenverantwortlich zu gestalten, konfrontiert die Studienanfänger mit Herausforderungen, denen sie teilweise unzureichend gegenüber stehen.

„Sowohl für den Einstieg in ein Hochschulstudium als auch für die weitere erfolgreiche Bewältigung der Studienanforderungen sind Fähigkeiten und Kompetenzen unverzichtbar, die von den Studierenden bereits vor Studienbeginn in der Schule und in

Einführungskursen erworben werden müssen. Dazu zählen vor allem grundlegende Kenntnisse in Abhängigkeit vom jeweils studierten Fach, aber unter anderem auch Fähigkeiten, das Studium in bestimmtem Maße selbständig organisieren zu können." [He10]

In den Schulen sind Lehr-Lern-Szenarien, welche die zukünftigen Abiturientinnen und Abiturienten auf diese besondere Situation des Übergangs von der Schule zum Studium vorbereiten, noch im Erprobungsstatus. Da didaktische Szenarien für eine lernzielorientierte Nutzung selbstbestimmter und selbstreflektierter Lernmethoden erst noch zu entwickeln sind, kann die Schule dem Anspruch der Lernenden auf Selbständigkeit, Selbstverantwortung und Selbstbestimmung nicht vollumfänglich durch eine veränderte Informations- und Lernkultur gerecht werden.

So schätzen 44% der Studienabbrecher ihre in der Schule erhaltene Vorbereitung auf ein Studium als unzureichend ein, selbst 32% der Absolventen eines Studiums kommen zu dieser Einschätzung.

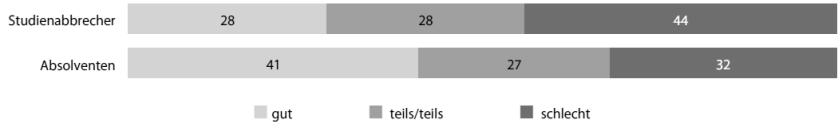


Abbildung 1: Vorbereitung in der Schule auf das Studium aus Sicht der Studienabbrecher und Absolventen [He10]

Die Entwicklung der Lernkompetenz als Fähigkeit der Lernenden, ihre Lernvorgänge selbständig zu planen, zu strukturieren, zu kontrollieren und zu reflektieren, rückt gegenwärtig immer stärker in den Fokus des Unterrichtsprozesses.

„Ziel der Entwicklung von Lernkompetenz ist es, dass Schüler ihre eigenen Lernvoraussetzungen realistisch einschätzen können und in der Lage sind, individuell geeignete Techniken situationsgerecht zu nutzen. [...] Für eine nachhaltige Wirksamkeit muss der Lernprozess selbst zum Unterrichtsgegenstand werden. Gebunden an Fachinhalte sollte ein Teil der Unterrichtszeit dem Lernen des Lernens gewidmet sein." [SK11]

Die Umsetzung dieser Zielstellungen wird insbesondere durch Aspekte sozialer Kompetenzen zur Kommunikation und Kollaboration zwischen Lehrendem und den Lernenden determiniert. Elemente wie Gruppenlernen, Diskussionsforen aber auch Portfolios zur Selbstreflektion des eigenen Lernerfolges unterstützen die Entwicklung der Sozial- und der Handlungskompetenzen der zukünftigen Studierenden. Hier liegen die Potenziale des E-Learning, mit dem es gelingt, handlungsorientierte Lehr-Lern-Prozesse sozial zu flankieren und einen Wechsel zwischen individueller und sozialer Lerntätigkeit zu organisieren, so wie dies im Studium alltäglich praktiziert wird.

Bisher ist es nicht gelungen, die Potenziale des E-Learning in den Schulen für den erfolgreichen Transfer des Unterrichts vom Prozess der Wissensvermittlung zum Prozess der Kompetenzentwicklung zu nutzen. Noch immer werden elektronische Medien vorrangig für die Internetrecherche und die Präsentation von Inhalten und Lernergebnissen genutzt.

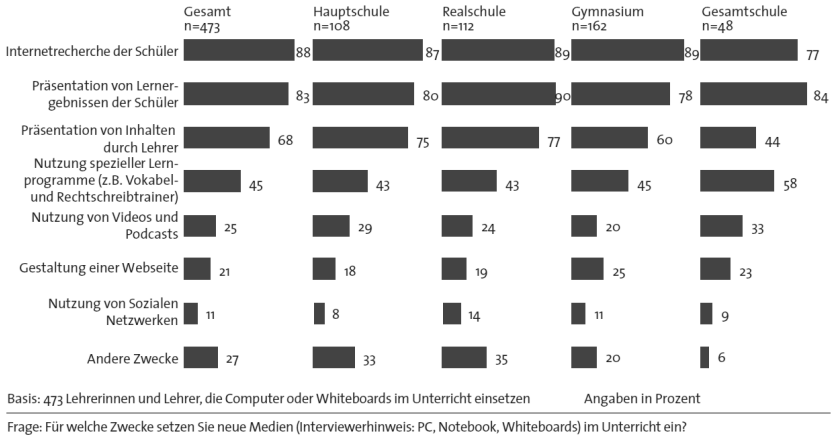


Abbildung 2: Einsatzzwecke elektronischer Medien - nach Schulart [BI11]

Die Ursachen dafür, dass webbasiertes Lernen beim Einsatz elektronischer Medien nur marginal Anwendung findet, sind vielschichtig. Unter anderem können der Ausstattungsgrad der Schulen mit IKT sowie die Lehrkompetenzen der Lehrenden den Nutzungsgrad webbasierter Lehr-Lern-Szenarien beeinflussen.

Für die Umsetzung einfacher Lehr-Lern-Szenarien genügen ein stabiler Internetzugang und die Ausstattung des entsprechenden Arbeitsraumes mit einer ausreichenden Anzahl Computerarbeitsplätze. Diesen Anforderungen werden die Schulen weitgehend gerecht, bereits im Schuljahr 2007/08 verfügten 99,2% der Schulen Deutschlands über im Unterricht einsetzbare Computer. Die durchschnittliche Anzahl der Schüler pro Computer ist auf unter 10 gesunken. 99,2% der deutschen Schulen verfügen über einen Internetzugang, 90% der Computer in der Schule sind vernetzt. [Se08]
Somit kann davon ausgegangen werden, dass der Ausstattungsgrad der Schulen webbasiertes Lernen ermöglicht und nur in Einzelfällen z.B. in ländlichen Gebieten die Netzanbindung zu technischen Problemen führen kann.

Die Einstellung der Lehrenden gegenüber elektronischen Medien hat sich in den letzten Jahren zum Positiven gewandelt. Im Jahr 2006 waren noch 47% der Lehrerinnen und Lehrer der Meinung, dass die Nutzung von IKT im Unterricht zu keinen signifikanten Vorteilen für die Schüler führt. [KH06] Mittlerweile ist sich die deutliche Mehrheit der Lehrer über den positiven Nutzen von PC- und Interneteinsatz in der Schule einig. Mehr

als 50% der Lehrenden aller Altersstufen stimmten den Fragen zum Nutzen von PC und Internet im Unterricht „voll und ganz“ bzw. „eher“ zu.

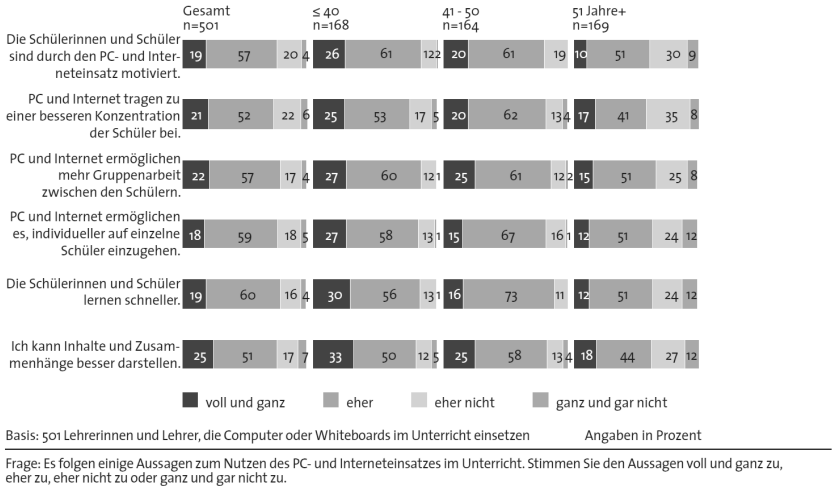


Abbildung 3: Aussagen der Lehrenden zum Nutzen von PC und Internet [BI11]

Mehr als 80% der Lehrerinnen und Lehrer sind der Meinung, dass die Rahmenbedingungen für den Einsatz elektronischer Medien zu verbessern sind. Dies betrifft insbesondere die Notwendigkeit zur Anpassung der Lehrpläne, die mangelhafte Quantität und Qualität der vorhandenen Lernmaterialien aber auch eine Modernisierung der Lehramtsstudiengänge, die von 84% der Lehrenden gefordert wird. [BI11]

„Die Ausweitung von Akzeptanz und Nutzung der neuen Medien ist allerdings kein Selbstläufer. Deshalb sind noch erhebliche Anstrengungen notwendig, u.a. in der Lehrerfortbildung. Entscheidend für die künftige Nutzung von digitalen Medien in der Schule sind überzeugende Inhalte und Konzepte für den schulischen Einsatz.“ [MM08]

Die Tatsache, dass 76% der Jugendlichen Social-Media-Plattformen täglich oder mehrmals wöchentlich aufsuchen und 49% der Schülerinnen und Schüler Computer und Internet mehrmals pro Woche nutzen um zu Hause für die Schule zu arbeiten bzw. zu lernen, macht das Potenzial für webbasiertes Lehren und Lernen deutlich. [MF12] Dies setzt jedoch voraus, dass die Lehrerinnen und Lehrer selbst über ausreichende Kompetenzen verfügen, um solche Szenarien didaktisch sinnvoll zu planen und methodisch zielführend umzusetzen.

„Students’ use of ICT for learning during lessons is related to teachers’ confidence level in their own ICT competences, their opinion about the relevance of ICT use for T&L and their access to ICT at school.” [EU13]

In den Projekten „UnIbELT[1]“ und „KoSEL[2]“ bestand und besteht u.a. die Aufgabe, E-Learning-Szenarien zu entwickeln und in den Schulen zu erproben, die Schülerinnen und Schüler auf den Übergang von der Schule zur Hochschule vorbereiten und ihnen Möglichkeiten zur Studienorientierung und zur Studienvorbereitung bieten. In der Phase der Umsetzung dieser Kurserprobung nehmen die Lehrerinnen und Lehrer die Rolle des Kursbetreuers ein.

2 Lehrerinnen und Lehrer als Kursbetreuer

Die E-Learning-Szenarien sind einerseits aus didaktischer Sicht so angelegt, dass die Lehrerinnen und Lehrer mit ihrer pädagogischen Erfahrung den Schülern als Coach zur Seite stehen und ihnen Impulse geben können aber auch zeitlich längere Phasen des selbstbestimmten Lernens vorgesehen sind. Der didaktische Anspruch an die Kursbetreuer/-innen ist hierbei so gestaltet, dass keine zusätzlichen Weiterbildungen für die Lehrenden erforderlich sind. Vielmehr bedeutet die Mitwirkung an den Projekten die Chance, über einen niederschwelligen Einstieg eigene Erfahrungen mit der Methode des E-Learning in Zusammenarbeit mit der AG „Didaktik der Informatik / Lehrerbildung“ der Technischen Universität Dresden zu sammeln.
Andererseits wurde die technische Umsetzung der E-Learning-Kurse auf einer Lernplattform so gestaltet, dass die Betreuung der Schülerinnen und Schüler keine erhöhten Herausforderungen an die Medienkompetenz für die Lehrenden darstellt.

Die in den genannten Projekten produzierten Kurse sind auf die Zielgruppe der Schülerinnen und Schüler der Jahrgangsstufen 10 bis 12 Allgemeinbildender und Beruflicher Gymnasien in Sachsen ausgerichtet. Mitwirkende Schulen ordnen die Durchführung der E-Learning-Kurse in ihr Konzept zur Studien- und Berufsorientierung ein, so dass sich an vielen der 29 mitwirkenden Schulen inzwischen eine fest etablierte Methodik der Kursdurchführung entwickelt hat.

[1] Übergang Schule-Hochschule mit Unterstützung Internetbasierter E-Learning-Tools
[2] Kompetenzentwicklung und Studienorientierung mit E-Learning

Zur administrativen Vorbereitung und zur Betreuung der Schüler während der Kursbearbeitung wurden in den meisten Fällen Lehrerinnen und Lehrer des betreffenden Gymnasiums gewonnen. Die Abbildung zeigt die Rollenverteilung und die Kommunikationskanäle, die während eines Kursdurchlaufes genutzt wurden.

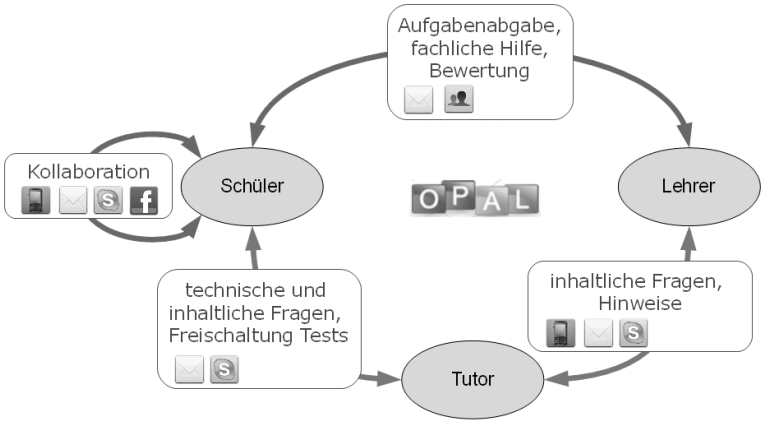


Abbildung 4: Rollenverteilung und Kommunikationskanäle während der Kursdurchführung

Die Erfahrung aus mehr als 100 Kursdurchläufen an den Schulen hat gezeigt, dass es vor allem bei der erstmaligen, selbstständigen Kursbearbeitung für die Lernenden vorteilhaft ist, wenn der Kursbetreuer die Schülerinnen und Schüler selbst unterrichtet oder wenigstens gut kennt. Kursteilnehmer haben anfangs immer wieder Probleme mit der Situation, die Kommunikation möglichst ausschließlich webbasiert zu realisieren und miteinander kollaborativ unter Nutzung von Social-Media – Anwendungen zu arbeiten. Hier ist eine entsprechende Führung der Lernenden durch den Kursbetreuer notwendig, die pädagogisches Geschick und Einfühlungsvermögen in die Situation der Schüler voraussetzt. Lehrerinnen und Lehrer sind mit ihrer pädagogischen Ausbildung besonders gut zur Erfüllung dieses Anspruchs geeignet.

Während der Kursbearbeitung können die teilnehmenden Schülerinnen und Schüler ihren Lernerfolg durch automatisch bewertete Tests, durch produzierte Artefakte eines E-Portfolios oder online einzureichende Aufgabenlösungen reflektieren. Die Aufgaben des Kursbetreuers bestehen hierbei in der Bewertung der eingereichten Dokumente unter Nutzung von Lösungsbildern, Punktevorschlägen etc. Die Lernplattform OPAL[3] bietet hierfür leicht bedienbare Bewertungswerkzeuge an, so dass eine kurze Einweisung genügt um die Lehrerinnen und Lehrer mit der Handhabung der Werkzeuge, der Punktevergabe, der Freischaltung von Tests und der Erteilung von Feedbacks vertraut zu machen. Das Bewertungswerkzeug bietet durch verschiedene Ansichtsmodi Möglichkeiten, sowohl alle erbrachten und noch offenen Leistungen eines Lernenden zu analysieren als auch die Ergebnisse aller Schülerinnen und Schüler in einem bestimmten

[3] Online Plattform für Akademisches Lernen

Kurselement anzuzeigen. Dadurch können die Lernenden individuell und differenziert geführt - sowie problematische Kurselemente erkannt werden.

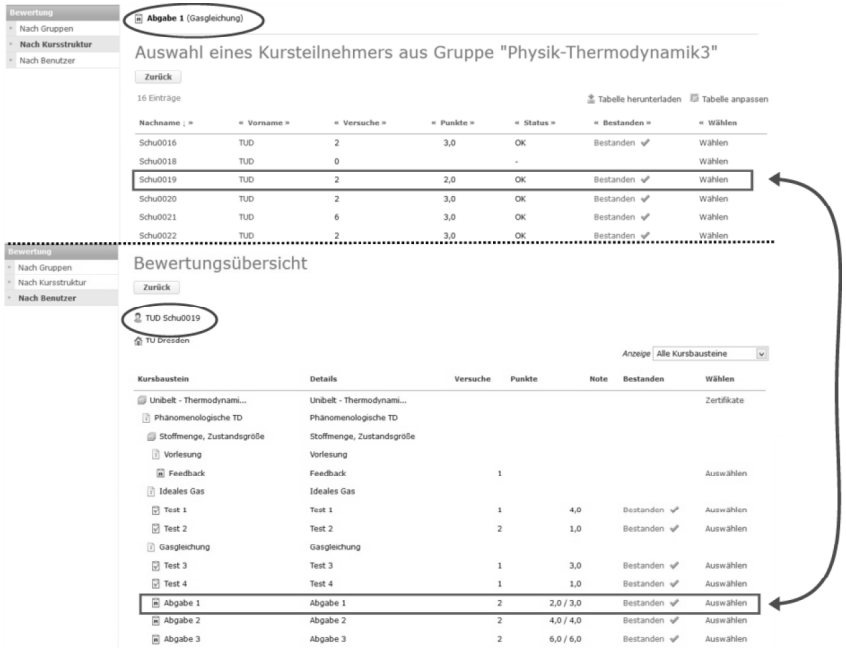


Abbildung 5: Darstellung beider Sichten des Bewertungswerkzeuges

Die besondere pädagogische Herausforderung während der Kursdurchführung besteht für die Lehrerinnen und Lehrer darin, einerseits den Lernprozess der Schülerinnen und Schüler weitgehend selbständig ablaufen zu lassen und dabei nicht zu stark zu führen. Das Setzen konkreter Termine für die Einreichung von Aufgaben, das Erteilen von „Hausaufgaben“ zum Kurs etc. würde hier der Intension des Projekts widersprechen, die Selbstlernkompetenz der Schüler zu entwickeln.
Andererseits ist es aber notwendig, auf Lernende mit Schwierigkeiten beratend zuzugehen, sie zu motivieren und ihnen neue Impulse zu verleihen. Dazu ist die Begleitung durch die betreuenden Lehrer/-innen unerlässlich, die hier bezüglich ihrer Fähigkeiten zur Binnendifferenzierung und zur Schaffung einer Balance zwischen pädagogischer Führung und zurückhaltender Beobachtung des Lernprozesses gefordert sind.

3 Der Lehrer als Kursautor

Die erfolgreiche Entwicklung von E-Learning-Szenarien, die den didaktischen Mehrwert gegenüber klassischen Lehr-Lern-Szenarien nutzen und zu einem positiven Lernerfolg führen, ist an organisatorische und personelle Bedingungen geknüpft. Es ist davon auszugehen, dass für eine Stunde reale Lernzeit in einem E-Learning-Kurs ein zeitlicher Autoringaufwand von mindestens 20 Stunden zu kalkulieren ist. [CM10]

Das Prozedere der Kurserstellung soll möglichst geringe Anforderungen an die Fähigkeiten und Fertigkeiten der Kursautoren im Umgang mit informatischen Werkzeugen stellen. Die weitaus größere Bedeutung kommt den didaktischen Fähigkeiten der Autoren und deren Fachwissen zu. Die Erfahrungen aus mehr als 25 Kursentwicklungen in den Projekten „UnIbELT“ und „KoSEL“ lassen die Ableitung eines Anforderungsprofils zur Entwicklung von E-Learning-Szenarien für Schülerinnen und Schüler zu:

- Die Kursautoren müssen über sicheres Fachwissen zur Thematik des Kurses verfügen.
- Die Kurve des Anforderungsniveaus soll einerseits an das durchschnittliche Niveau der Schulbildung anknüpfen, welches den Autoren bekannt sein muss. Gemäß den Zielstellungen der o.g. Projekte zur Studienvorbereitung soll die Leistungskurve andererseits auf einer Niveaustufe enden, die mit den Anforderungen an einen Studierenden im Grundstudium vergleichbar ist. Dazu ist Wissen über die an den Hochschulen gängigen Lehrmethoden und die entsprechenden Lerninhalte notwendig.
- Zur zielgruppen-adäquaten Entwicklung von E-Learning-Szenarien sind Erfahrungen in der didaktisch-methodischen Aufbereitung der Lerninhalte und entsprechende Lehrkompetenzen unerlässlich.
- Die Kursautoren müssen über ein ausreichendes Zeitbudget verfügen, um unter Anwendung des o.g. Lernzeit-Autoring-Verhältnisses von 1:20 den Kurs in einer vertretbaren Zeitspanne entwickeln und testen zu können.

Es ist davon auszugehen, dass Lehrerinnen und Lehrer sowohl über das nötige Fachwissen als auch über das didaktische Knowhow verfügen, um die Lernkurve des jeweiligen Kurses zielgruppenadäquat zu gestalten. Daher sind es eher organisatorische Aspekte, die das Kursautoring durch Lehrende in Frage stellen. In den Projekten „UnIbELT“ und „KoSEL“ sind Lernszenarien erstellt worden, die für eine Lernzeit von 10 bis 15 Stunden konzipiert sind. Das Verhältnis zwischen Lernzeit zu Autoringzeit von 1:20 hat sich bestätigt, so dass pro Kursentwicklung mindestens 200 Arbeitsstunden zu veranschlagen sind. Damit sind Lehrerinnen und Lehrer zur Entwicklung solcher Kurse kaum mehr geeignet, da dieser zeitliche Aufwand neben den eigentlichen beruflichen Aufgaben nicht in einem vertretbaren Zeitraum zu leisten ist. Dennoch ist es vorstellbar, dass Lehrende kleinere E-Learning-Szenarien mit entsprechend geringerem Aufwand entwickeln, sofern die technischen Hilfsmittel die benötigte Unterstützung leisten.

Zur Kursentwicklung wurden in den Projekten deshalb examinierte Lehramts-Studierende favorisiert, die über das nötige Zeitbudget aber auch die entsprechende fachliche und didaktische Ausbildung verfügen.

Der auf der Lernplattform OPAL abgelegte Content der Kurse wird durch xml-Konstrukte repräsentiert. Weder den Lehrern noch den Lehramtsstudierenden sollte es zugemutet werden, zur Content-Erstellung xml-Kenntnisse nachzuweisen oder alternativ die meist in ihrer Funktionalität beschränkten, internen Editoren der Lernplattform zu nutzen. Hier bestand die Aufgabe, den eingangs geforderten niederschwelligen Einstieg in die Kursentwicklung mit geeigneten Werkzeugen zu organisieren. Im Rahmen des Projekts „UnIbELT“ wurde deshalb ein Verfahren entwickelt, mit dem zunächst die Kursinhalte in einem Textsatz-System (hier OpenOffice-Writer) erstellt werden. Dazu erhalten die Kursautorinnen und –autoren eine vorgefertigte Dokumentvorlage, in der alle benötigten und in allen Kursen einheitlich anzuwendenden Formate definiert und abgelegt sind. Erst nach Abschluss der inhaltlichen Gestaltung werden die Dokumente unter Nutzung einer Extension in das xml-Format exportiert. Der Quellcode bindet passende CSS-Stylesheets ein, so dass nach Ablegen der Dateien auf der Lernplattform eine nahezu originalgetreue Anzeige als Webseite erfolgt.

Fazit

Es darf konstatiert werden, dass Lehrerinnen und Lehrer der Methode des E-Learning offen gegenüberstehen und die Potenziale dieser Lernmethode erkannt haben. Der Einsatz webbasierter Lernszenarien kann in den Schulen forciert werden, wenn es gelingt, den Lehrenden einen niederschwelligen Einstieg zu bieten und vorgefertigte Anwendungsbeispiele bereitzustellen. Durch Nutzung dieser existierenden Szenarien vertiefen die Lehrenden ihre Kompetenzen in der pädagogischen Betreuung webbasierter Lehr-Lernprozesse, in der Gestaltung kompetenzorientierter Lehre und der Organisation kollaborativen Lernens.

Die Erfahrungen aus den beiden Projekten „UnIbELT“ und „KoSEL“ zeigen, dass aus der Nutzung existierender E-Learning-Szenarien der Anspruch der Lehrenden erwächst, eigene Szenarien zu entwickeln und mit den Schülern zu erproben. Hierfür bedarf es einer angemessenen technischen, aber auch der ökonomischen Unterstützung. Der existierende Workflow vom einfachen OpenOffice-Dokument hin zum webbasierten Lernszenarium – entwickelt und getestet an mehr als 25 Kursentwicklungen in den o.g. Projekten – steht zur Nachnutzung bereit und darf als probates Mittel angesehen werden, Lehrerinnen und Lehrern den Weg in die Methode des E-Learning nicht nur als Anwender sondern auch als Produzenten zu ebnen.

Literaturverzeichnis

[BI11] BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.: Schule 2.0. Eine repräsentative Untersuchung zum Einsatz elektronischer Medien an Schulen aus Lehrersicht. BITKOM, Berlin, 2011.

[CM10] Clauß, M. et al.: Towards a Framework for Developing Standardized E-Learning Modules - A Report on Methods and Tools in a Distributed Content Production Project. In: (Cordeiro, J. et al.): CSEDU 2010. Proceeding of the 2nd International Conference on Computer Supported Education, Volume 1, Valencia, Spain, 7.-10.4.2010.

[EU13] Europäische Kommission: Survey of Schools: ICT in Education. Final Study Report. Brüssel, 2013.

[He10] Heublein, U. et.al.: Ursachen des Studienabbruches in Bachelor- und herkömmlichen Studiengängen. HIS Hochschul-Informations-System GmbH, Hannover, 2010.

[KH06] Korte, W.B.; Hüsung, T.: Benchmarking Access and Use of ICT in European Schools 2006. empirica Gesellschaft für Kommunikations- und Technologieforschung mbH, Bonn, 2006.

[MF12] Medienpädagogischer Forschungsverbund Südwest: KIM-Studie 2012. Stuttgart, 2013.

[MM08] Institut für Medien- und Kompetenzforschung: Digitale Schule – wie Lehrer Angebote im Internet nutzen. Essen, 2008; http://www.dlr.de/Portaldata/45/Resources/dokumente/bildungsforschung/MMB_Veroeffentlichung_Lehrer_Online_20080505_final.pdf.

[Se08] Sekretariat der Ständigen Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland: Dataset - IT-Ausstattung der Schulen. Bonn, 2008.

[SK11] Sächsisches Staatsministerium für Kultus und Sport: Lehrplan Gymnasium. Saxoprint GmbH, Dresden, 2011.

Web und Mobile Apps Programmieren mit Dart

Marco Jakob

Kalaidos Fachhochschule Schweiz
majakob@gmx.ch

Abstract: Bisher war es kaum realistisch, im Anfängerunterricht mobile oder webbasierte Applikationen zu entwickeln. Die Programmiersprache Dart bietet neue Möglichkeiten, wie solche Applikationen viel einfacher programmiert werden können. Zudem zeigt sich, dass Dart in einigen Bereichen sogar besser geeignet ist in die objektorientierte Programmierung einzuführen als herkömmliche Programmiersprachen.

1 Einleitung

Heute sind vor allem *mobile und webbasierte Anwendungen aktuell*. Immer weniger werden Desktop Applikationen, d.h. fest installierte Programme, verwendet. Viele Nutzer bewegen sich fast ausschliesslich im Webbrowser oder auf dem Mobiltelefon.

Es ist deshalb sehr motivierend, wenn Lernende im Unterricht mobile oder webbasierte Anwendungen selber entwickeln können. Zudem können sie solche Anwendungen auf einfache Weise mit ihren Freunden teilen.

Mit den gängigen Ansätzen, wie in den letzten Jahren Programmieren unterrichtet wurde, ist es aber *kaum oder nur sehr beschränkt möglich*, mobile oder webbasierte Anwendungen zu entwickeln.

Im Folgenden wird zuerst dargelegt, weshalb es mit herkömmlichen Möglichkeiten schwierig ist, mobile oder webbasierte Applikationen im Anfängerunterricht zu programmieren. Anschliessend wird die neuere Programmiersprache *Dart* vorgestellt und untersucht, in wiefern sich diese für den Unterricht eignet.

2 Problematik mobiler Anwendungen

Android und Apples iOS sind die gängigsten Plattformen für mobile Anwendungen. Der Einsatz dieser Plattformen im Unterricht ist mit Schwierigkeiten verbunden, die hier kurz erwähnt werden.

iOS-Apps. Die Entwicklung von iOS-Applikationen erfolgt in einer eigenen Sprache namens Objective-C speziell für die iOS-Plattform. Dies bedeutet, dass die Programme auch nur auf iPhones bzw. iPads laufen. Zum Entwickeln von iOS-Anwendungen benötigt man

(bevorzugt) ein Mac-Betriebssystem. So ein System dürfte nicht überall zur Verfügung stehen. Zudem ist es sehr umständlich bis unmöglich für eine Schule, die eigenen Programme über den App Store auf anderen iOS-Geräten zu installieren.

Android-Apps. Obwohl Android-Applikationen in Java programmiert werden, stellt sich die gleiche Problematik wie bei iOS-Applikationen: Die Programme laufen nur auf Geräten mit einem Android-Betriebssystem. Der Besitz eines Android-Gerätes kann aber nicht vorausgesetzt werden. So werden Lernende ausgeschlossen, die ihr Programm auf einem iOS-Gerät, einem Windows Phone oder auf dem Desktop ausführen möchten.

3 Problematik webbasierter Anwendungen

Webbasierte Anwendungen haben den Vorteil, dass sie *sowohl auf Desktopbrowsern* (Internet Explorer, Firefox, Chrome, Safari) *als auch auf mobilen Browsern* laufen können. Zur Entwicklung von Webanwendungen gibt es konvenzionell zahlreiche Möglichkeiten, die aber alle ihre Tücken haben, wenn sie im Unterricht eingesetzt werden sollen.

PHP, ASP.net etc. Diese sogenannten Skriptsprachen werden häufig verwendet, um dynamische Webanwendungen zu erstellen. Dabei werden auf einem Webserver HTML-Seiten generiert, welche dann im Browser angezeigt werden können. Die so notwendige Kommunikation zwischen dem Server und dem Browser auf dem Client bringt zusätzliche Komplexität in das Erlernen einer Programmiersprache. Ausserdem eignen sich Skriptsprachen nicht besonders gut, um moderne, objektorientierte Programmierkonzepte zu unterrichten.

Python, Java, C++, Visual Basic etc. Mit all diesen Sprachen lassen sich objektorientierte Programmierkonzepte vermitteln. Der Weg zu einer Webapplikation ist jedoch sehr aufwändig und anspruchsvoll. All diese Programmiersprachen laufen nicht direkt im Browser. Somit muss, wie bei den Skriptsprachen, eine Serverapplikation geschrieben werden, welche HTML-Seiten generiert. Und da diese Sprachen nicht für die Webprogrammierung entwickelt wurden, ist die Komplexität noch höher als bei den oben genannten Skriptsprachen.

JavaScript. JavaScript ist die einzige Programmiersprache, welche von allen gängigen Browsern unterstützt wird. Es ist somit möglich, damit ganze Webanwendungen zu schreiben, welche direkt in Desktopbrowsern und mobilen Browsern laufen können. Leider ist aber JavaScript für den Unterricht weniger geeignet. JavaScript ist eine Programmiersprache mit schwierigen Konzepten und vielen Ausnahmefällen. Diese Hindernisse sind (nicht nur) für Anfänger sehr frustrierend.

4 Dart

Google hat eine neue Sprache namens Dart[1] entwickelt.

Dart ist *für das Web gemacht* und lässt sich direkt in JavaScript übersetzen. Somit laufen Dart-Anwendungen ohne zusätzliche Installation in allen modernen Desktopbrowsern und mobilen Browsern.

Abbildung 1: Dart-Programme laufen auf allen modernen Plattformen

Dart ist eine *objektorientierte Sprache*, welche sich sehr gut für den Unterricht eignet. Sie ist stark an die bekanntesten Programmiersprachen wie Java und C++ angelehnt, ist aber konsistenter und eleganter. Das macht die Sprache einfach zu lernen für Umsteiger und viel einfacher für Programmieranfänger (siehe Code-Beispiele weiter unten).

Fast genauso wichtig wie die Programmiersprache selbst ist das, was rund um die Sprache zur Verfügung steht:

Dart Editor ist eine komfortable Entwicklungsumgebung für Dart. Der Editor enthält professionelle Programmierhilfen wie Code-Vervollständigung, Umbenennen von Variablen und Debugging. Trotzdem ist der Dart Editor sehr einfach und übersichtlich gehalten, d.h. er enthält keine unnötigen Elemente, die Anfänger abschrecken könnten.

Der Dart Editor läuft *ohne Installation* auf Windows, Linux und Mac. Da keine Installation nötig ist, kann er zum Beispiel auch von einem USB-Stick gestartet werden.

Viele *nützliche Bibliotheken* sind bei Dart schon eingebaut. Zum Beispiel für mathematische Berechnungen, für Interaktionen mit HTML, zum Speichern in Dateien und Datenbanken, für Kryptografie etc. Zusätzliche Bibliotheken werden nach Bedarf automatisch von einem zentralen Server heruntergeladen.

Es gibt drei Möglichkeiten, Dart-Programme auszuführen: In der Kommandozeile, als Ja-

[1] Siehe http://www.dartlang.org

vaScript in allen modernen Browsern oder in Chromium, einem Google Browser mit Dart Virtual Machine.

Man kann auch *Server-Applikationen* in Dart programmieren. So ist es möglich, eine gesamte Client-Server-Applikation in der gleichen Sprache zu verfassen. Der Server wird in der Kommandozeile gestartet und von einem Browser Client angesprochen. Mit kaum einer anderen Programmiersprache ist dies möglich.

5 Dart Code-Beispiele

Anhand von Code-Beispielen soll gezeigt werden, wie einfach es ist, mit Dart zu programmieren. Wer bereits eine Programmiersprache kennt, dem wird Dart sehr vertraut vorkommen. Dies ermöglicht den Umstieg auf diese Sprache innert weniger Stunden.

Dort, wo herkömmliche Sprachen unnötige Stolpersteine beinhalten, bietet Dart elegante Lösungen. Um dies aufzuzeigen, werden bei einigen Beispielen Parallelen zu Java gezogen.

5.1 Hello World

Mit diesem Klassiker zeigt sich, wie aufwändig es ist, ein erstes Programm in einer Sprache zum laufen zu bringen. Ein minimales Programm in Dart sieht wie folgt aus:

```
main() {
  print("Hello, World!");
}
```

Angenommen, das Programm oben wurde unter *helloworld.dart* gespeichert. Nun kann es von der Kommandozeile wie folgt gestartet werden:

```
dart helloworld.dart
```

Kommentar. Obwohl in Dart Klassen häufig verwendet werden, gibt es die Möglichkeit, Funktionen ohne Klassen zu definieren. In Java ist dies nicht möglich. Dort würde ein entsprechendes Programm wie folgt aussehen:

```
// Java
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

Der Anfänger ist mit einem solchen Programm schnell überfordert. Neben den Schlüsselbegriffen *public, class, static* werden fortgeschrittene Sprachelemente wie Arrays und ein statischer Aufruf verwendet. Um dieses Java *Hello, World* zu verstehen, braucht es bereits etliches an Programmierkenntnissen.

Ein weiterer Schritt, der bei Dart wegfällt, ist das Kompilieren. Kompilieren ist für die Dart Virtual Machine[2] (VM) nicht nötig, da die VM direkt den Quellcode ausführt.

5.2 Funktionen und Variablen

```
// Funktion definieren.
printNumber(int number) {
  // Nummer auf Konsole ausgeben.
  print("Die Nummer ist $number.");
}

main() {
  // Variable deklarieren und initialisieren.
  var number = 42;
  // Funktion aufrufen.
  printNumber(number);
}
```

Kommentar. Typen von Variablen, Parametern und Rückgabewerten können optional angegeben werden. Man könnte also bei *var number* den Typ genauer angeben und *int number* schreiben. Damit würde man vom Dart Editor gewarnt, wenn man etwas anderes als einen Integer in diese Variable speichern möchte. Deshalb ist es oft sinnvoll, den Typ anzugeben. Für eine Einführung in die Thematik der Variablen ist es aber praktisch, wenn man zuerst ohne Typen arbeiten kann. Diese Möglichkeit hat man bei den meisten Programmiersprachen nicht.

5.3 Integer

```
// String in einen int umwandeln.
int i = int.parse("5");

// int in einen String umwandeln. Da 22 ein Objekt ist,
// kann man Funktionen darauf aufrufen.
String s = 22.toString();
```

[2] Gründe, warum Dart VM keine Bytecode VM ist: http://www.dartlang.org/articles/why-not-bytecode/

Kommentar. In Dart gibt es keine Unterscheidung zwischen elementaren Datentypen und Objekten (wie etwa in Java). Alles, was in eine Variable gespeichert werden kann, ist ein Objekt. Das ist für das Verständnis viel einfacher!

Ausserdem kann ein int eine beliebige Grösse haben. Man braucht sich also nicht darum zu kümmern, ob man noch im Zahlenbereich drin ist oder wie in Java zwischen int und long unterscheiden.

5.4 Klassen

```
import 'dart:math';

class Point {
  num x;
  num y;

  // Konstruktor (kurze Schreibweise)
  Point(this.x, this.y);

  num distanceTo(Point other) {
    var dx = x - other.x;
    var dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
  }
}

main() {
  var p = new Point(2, 3);
  var q = new Point(3, 4);

  // String Interpolation
  print('Distanz von p zu q = ${p.distanceTo(q)}');
}
```

Kommentar. Definition einer Klasse und das Erzeugen von Objekten funktionieren analog zu Java. Einzig der Konstuktor mit den Zuweisungen zu den Variablen ist eine verkürzte Schreibweise. Optional könnte man den Konstruktor auch gleich schreiben wie in Java:

```
// Konstruktor (lange Schreibweise)
Point(num x, num y) {
  this.x = x;
  this.y = y;
}
```

5.5 Interaktion mit dem Browser

```
import 'dart:html';

main() {
  // HTML Knopf erstellen.
  var button = new ButtonElement()
  ..text = 'Bestellen'
  ..classes.add('wichtig')
  // Beim Klicken die Funktion handleOnClick aufrufen.
  ..onClick.listen(handleOnClick);

  // Knopf in HTML einfuegen.
  query('#bestellung').children.add(button);
}

void handleOnClick(MouseEvent event) {
  window.alert('Danke!');
}
```

Kommentar. Graphische Benutzeroberflächen werden in Dart für den Browser geschrieben. Nach dem Übersetzen des Dart-Codes in JavaScript können alle modernen Browser das Dart-Programm anzeigen. In Java würde man typischerweise eine Benutzeroberfläche mit Swing oder JavaFX programmieren. Im Gegensatz zu Dart lassen sich diese aber nicht direkt im Browser ausführen.

6 Fazit

Mit Dart gibt es erstmals eine realistische Möglichkeit, mobile und webbasierte Applikationen im Anfängerunterricht zu programmieren. Zudem zeigt sich, dass Dart in einigen Bereichen sogar besser geeignet ist in die objektorientierte Programmierung einzuführen als herkömmliche Sprachen. Dies ist möglich, da Dart eine neuere Sprache ist und deshalb auf jahrzentelanger Erfahrung mit Sprachen wie Java, C++, Visual Basic, etc. aufbauen konnte.

7 Weitere Details

Unterrichtsmaterial und weitere Informationen zum Programmieren mit Dart finden Sie auf der folgenden Webseite:

- http://edu.makery.ch (unter *Projects*, *Learn Dart*)

Das „Flipped Classroom“-Konzept

Alexandra Kück

Kurt-Körber-Gymnasium
Niendorfer Gehege 260
22527 Hamburg
alex.kueck@gmx.net

Abstract: „To flip“ – aus dem Englischen kommend bedeutet ins Deutsche übersetzt „umdrehen, umkehren, wenden“. „Flipping the Classroom“ oder auch „Reversing the Classroom“ ist ein Unterrichtsprinzip, bei dem die Aufgaben, die im herkömmlichen Unterricht zu Hause erledigt werden (Hausaufgaben) mit denjenigen, die sonst im Unterricht bearbeitet werden (Inhalte vermitteln) – getauscht („flipping“, „reversing“) werden.

Das heißt also, dass konkrete Aufgaben im Unterricht bearbeitet werden (anstatt sie als Hausaufgaben alleine zu Hause zu bearbeiten), wenn der Lehrer und andere Mitschüler bereit sind, bei Problemen zur Seite zu stehen.

Inhalte, die sonst über Lehrvorträge oder andere Methoden vermittelt werden, werden nun in Screencasts (Videos, bei denen lediglich der Bildschirminhalt incl. Ton aufgezeichnet wird. Es ist keine externe Kamera nötig, die Screencasts werden mittels einer speziellen Software direkt auf dem Rechner oder Tablet PC aufgezeichnet) festgehalten, welche der Schüler in Ruhe zu Hause anschauen und bearbeiten kann. Er kann das so oft er es mag und braucht tun. Er kann den Vortrag auch anhalten, um Informationen aus dem Video erst einmal zu verarbeiten oder sich in Ruhe Notizen machen zu können.

Die Zeit, die der Lehrer in der Unterrichtsstunde für einen Lehrvortrag verwenden würde, kann er nun einsetzen, um Schüler individuell bei ihren Verständnisproblemen und Fragen zu beraten. Die Schüler ihrerseits können die Unterrichtszeit nutzen, um Aufgaben oder Texte zu den neuen Themen zu bearbeiten. Es geht also während des Unterrichts nicht mehr zentral darum, neue Inhalte zu „erlernen“ bzw. zu vermitteln, es geht darum, die zu Hause studierten Inhalte nun in der Unterrichtsstunde auf konkrete Aufgabenstellungen anzuwenden („flipping the classroom“).
So definiert sich die reine Form des „Flipped Classroom“-Konzeptes, wie es in den USA ursprünglich für Studenten entwickelt wurde.

Das Konzept wurde abgeändert und durch neue Komponenten erweitert, da es sich aus praktischer Erfahrung heraus bewährt hat, an einigen Stellen auf die Schüler anders einzugehen als es z.B. bei Studenten nötig gewesen wäre, für die das Konzept ursprünglich entwickelt wurde.

Diese Änderungen waren auch nötig, da das Prinzip aus den USA kommt und nicht eins zu eins auf das deutsche Schulsystem übertragen werden kann. Z.B. ist die Fächerzahl in den USA viel geringer. Fächer werden dort sehr intensiv, dafür aber auch nur halbjährlich unterrichtet.

1 Erweiterungen des "Flipped Classroom"-Konzeptes

2.1 Arbeiten mit einem Skript

Ein Skript ist eine der Lerneinheit entsprechende Zusammenstellung mit Hinweisen zum Vorgehen, allgemeinen, wichtigen Informationen oder auch Definitionen zu dem behandelten Thema, Verweisen auf Materialien, die die Inhalte des behandelten Themas erklären, dazu passenden Aufgaben (bzw. Verweisen auf Arbeitsblätter) und den dabei zu erreichenden Kompetenzen. Mit Hilfe eines Skriptes ist ein Schüler in der Lage, sich ein neues Thema, mit Hilfe unterschiedlichster Materialien, eigenständig zu erarbeiten. Es kann sich um eine einfache Worddatei handeln – je nachdem, welche Ausstattung die Schüler zur Verfügung haben, digital oder ausgedruckt.
Aufgaben, die ein Schüler normalerweise über verschiedene Arbeitsblätter vom Lehrer zur Verfügung gestellt bekommt, um ein gerade behandeltes Thema zu bearbeiten und zu zeigen, dass er in der Lage ist, sein zu dem Thema erworbenes Wissen anzuwenden, werden in das Skript integriert.

Es ist sinnvoll, das zu behandelnde Thema in einzelne, logisch abgeschlossene Module aufzuteilen. Man kann dem Schüler so eine Struktur für den Aufbau des Themas vorgeben. Diese Module zusammen ergeben dann das gesamte Skript. Z.B. könnte man ein Skript zum Thema Bruchrechnung in die Module „Brüche addieren/subtrahieren", „Brüche multiplizieren" und „Brüche dividieren" aufteilen.

Zu jedem Modul werden jeweils die am Ende zu erreichenden Kompetenzen angegeben, damit der Schüler einschätzen kann, ob er eine solche Einheit wirklich „gemeistert" hat. Das bedeutet natürlich auch, dass der Schüler Stück für Stück lernt, diese Eigenverantwortung bzgl. seines Lernprozesses zu übernehmen.

Der Vorteil, alles in einem Skript zusammenzufassen, besteht darin, dass der Schüler eine Art **Leitfaden** bzw. eine Struktur an die Hand bekommt, der bzw. die ihn durch das Thema führt, der Schüler aber trotzdem eigenverantwortlich seinen Lernprozess steuert.

Das Skript besteht im Wesentlichen aus:

- Verweisen auf **Materialien**, die das Thema erklären
    - Hinweisen auf die die Module erklärenden **Screencasts.**
    - **eigenen Erklärungen** des Lehrers zu den einzelnen Modulen (bzw. bei längeren Erklärungen aus den Verweisen darauf).
    - **Verweisen** auf Erklärungen in ausgesuchten Websites.
    - **Verweisen** auf Bücher, Zeitschriften etc.
    - Hinweisen auf passende **Audio-Dateien**
- **Aufgaben** in jedem Modul**,** um Inhalte zu vertiefen oder anzuwenden.
- **Kompetenzen**, die in jedem Modul zu erreichen sind.

2.2 Skript mit Projektaufgabe

Ein Skript, so wie es oben beschrieben wurde, ermöglicht eigenverantwortliches Lernen – vor allem in Bezug auf

- **Lerntempo**: jeder Schüler kann in seinem Tempo weiterarbeiten, muss am Ende dann einen „Mindestlevel“ geschafft haben. Schnellere Schüler bekommen zusätzliche Aufgaben oder auch zusätzliche Projekte.
- **Lerntyp:** freie Material-Wahl: Text, Zeitschrift, Buch, Website, Screencast, Audio-Datei.

Wie ein Schüler ein Thema bearbeitet (bzgl. Reihenfolge und Intensität) wird aber ganz strukturiert vorgegeben.
Für schwächere Schüler ist es durchaus sinnvoll, mit einem solch detaillierten Skript zu arbeiten, da man ihnen eine Struktur der zu bearbeitenden Inhalte vorgibt. Den Schülern, die in der Lage dazu wären, nimmt es aber eventuell die Kreativität, den Forscherdrang und die Herausforderung sich selbst mit neuen Themen auseinander zu setzen. Für solche Schüler kann es sinnvoll sein, eine projektbezogene Aufgabenstellung zu geben und dazu nur die zu erreichenden Kompetenzen anzugeben. Eventuell kann man noch erklärende Screencasts oder weitere Hinweise auf begleitende Materialien (Internetseiten, Zeitschriften, Bücher, etc.) dazu anbieten.

Die schwächeren Schüler wären mit so einem Skript aber extrem überfordert. Daher bietet man am besten beide Möglichkeiten an. Auf diese Art und Weise können sich die Schüler, die sich das zutrauen, erst mal an die schwierigere Variante herantrauen und dann auf das stärker strukturierte Skript mit detaillierten Beschreibungen und Aufgaben wechseln, wenn sie merken, dass sie nicht klarkommen (oder auch anders herum).

Ein Skript mit Projektaufgabe setzt sich aus folgenden Komponenten zusammen **(hinzugekommen sind die Punkte 1,2 und 4c)**

1. Einer einführenden **Projektaufgabe**.
2. **Allen Kompetenzen**, die am Ende des gesamten Skriptes durch die Bearbeitung des Projektes erreicht werden sollen.
3. Einer Auflistung **aller Materialien** bzw. Hinweisen darauf, wo Materialien zu finden sind, um sich Informationen einzuholen, die hilfreich sind, um das Projekt bearbeiten zu können. Z.B.
   - Erklärenden Screencasts
   - eigenen Erklärungen zu den jeweiligen Projektaufgaben
   - Verweisen auf Erklärungen in ausgesuchten Websites
   - Hinweisen auf passende Bücher, Zeitschriften oder weiteren schriftlichen Materialien.
   - Audio-Dateien
4. Nach Modulen aufgeteilt:
   a. Materialien (Screencasts, Infotexte, Zeitschriften, Audiodateien, etc.)
   b. Aufgaben, um Inhalte, die in den Materialien gegeben wurden, anzuwenden.
   c. Aufgaben, die so gegliedert sind, dass sie Stück für Stück zu der Lösung des Gesamt-Projektes, welches gleich zu Beginn gegeben wurde, hinführen. Diese Projekt-Aufgaben sollten jeweils passend zu den behandelten Inhalten des Moduls formuliert sein, aber immer im Hinblick auf die Lösung des Gesamt-Projektes.
   d. Kompetenzen, die in jedem Modul zu erreichen sind.

2.3 Arbeiten mit Schüler-Blogs zur Lernreflexion

In den Schüler-Blogs können die Schüler Ihre eigenen Lernwege und -strategien festhalten und reflektieren. Es bietet sich vor allem an, wenn die Schüler eigenständig mit einem Skript oder sogar an Projekten arbeiten und kann auch auf Papier geführt

werden. Es ist sehr hilfreich, den Schülern zu vermitteln, dass der Blog sie darin unterstützen soll, eine „forschende“ Haltung zu den behandelten Themen oder Projekten einzunehmen. Über die unten angegebenen Fragestellungen kann man den Schülern klarmachen, dass sich ein guter Forscher immer Notizen zu seinen Untersuchungen macht, um später eventuell darauf zurückzukommen, damit weiter zu arbeiten oder auch neue Erkenntnisse, die man meinte gewonnen zu haben, zu überarbeiten. Diese „forschende“ Grundhaltung ist es ja im Endeffekt, die man von den Schülern haben möchte.

Natürlich kann ein Blog die Schüler auch ganz unabhängig vom Skript in Ihren Lernprozessen begleiten.

Z.B. über folgende Aspekte sollte der Blog „haushalten“:

- **Woran habe ich heute geforscht** (nur die **neuen** Erkenntnisse, die ich daraus gewonnen habe, werden hier festgehalten)?
- Was hat **gut geklappt**?
- Wobei hatte ich **Fragen/Probleme** (genau formulieren, wie das Problem/die Frage lautet)?
- Welche wichtigen **Merkregeln, Aussagen, Definitionen** oder gar „Vorgehensweisen“, mit der man bestimmte Aufgabentypen lösen kann, sie herausgefunden haben.

Ein Blog enthält also

- die R**eflexion des eigenen Lernprozesses** und
- wichtige **Ergebnisse, Merkregeln, Definitionen** oder Ähnliches (möglichst in eigenständig formulierten Texten oder Bemerkungen z.B. zu bearbeiteten Formeln oder Definitionen).

Er enthält jedoch nicht die konkrete ausführliche Bearbeitung der Aufgaben aus den Arbeitsblättern oder vom Skript. Diese werden entweder in separaten Seiten des Blogs, anderen elektronischen Dokumenten oder einer Mappe geführt.

Wichtig ist, dass der Schüler lernt, **eigene Formulierungen oder auch Skizzen von Zusammenhängen** im Blog festzuhalten. So kann er feststellen, ob er bestimmte Zusammenhänge wirklich verstanden hat. Auf der anderen Seite hat der Lehrer dadurch die Möglichkeit, falsche oder fehlerhafte Vorstellungen mit dem Schüler zusammen zu korrigieren.

2.4 Arbeiten mit Lehrer-Blogs, um Unterricht transparent zu halten

Unabhängig davon, ob man im Unterricht mit oder ohne Skript arbeitet, kann man den Unterrichtsverlauf in einem Blog, also einer Art Online-Tagebuch, transparent machen. Dort kann man

- Allgemeine Hinweise für die Schüler
- wichtige Termine (Lernkontrollen etc.)
- Materialien, z.B. Arbeitsblätter, Texte oder Hinweise auf Webseiten
- Ergänzungen zum Skript
- Videos und Aufgaben (falls Sie ohne ein Skript arbeiten)
- Unterrichtliche Inhalte
- Kompetenzen
- Lösungen und Ergebnisse

festhalten.

Sollte man Gefallen an dem „Flipped Classroom"-Konzept finden und den Schülern daher öfter erklärende Screencasts zur Verfügung stellen, arbeitet man dabei aber nicht mit einem Skript, so wird man schnell feststellen, dass einige Schüler den Überblick darüber verlieren, welche Screencasts sie zu welchen Themen oder Arbeitsblättern begleitend anschauen können.
Es bietet es sich bei dieser Art des Unterrichts an, einen begleitenden Lehrerblog zu führen, um den Unterricht transparent zuhalten, eine Materialsammlung zu erstellen und genau anzugeben, welche Materialien zu welchen Themen zur Verfügung gestellt wurden. Auch Videos, die man nachträglich überarbeitet hat, kann man hier angeben.

2 Konkrete Unterrichtsgestaltung mit Hilfe des erweiterten "Flipped Classroom"-Konzeptes

Das erweiterte „Flipped Classroom"-Konzept bietet hervorragende Möglichkeiten, den eigenen Unterricht in Bezug auf folgende Punkte neu zu gestalten:

- Öffnung
- Differenzierung
- Kompetenzorientierung
- Motivation
- Benotung

Im Folgenden wird ein möglicher Ablauf einer Unterrichtsstunde nach dem erweiterten „Flipped Classroom“-Konzept aufgezeigt. Natürlich kann man einzelne Komponenten auch einfach weglassen. Man muss z.B. keinen begleitenden Lehrer-Blog führen. Auch die Arbeit mit einem Skript ist nicht zwingend erforderlich.

Man könnte einfach anfangen, zu einem Arbeitsblatt begleitend erklärende Screencasts einzusetzen, die die Schüler zu Hause sehen sollen, um ihnen anschließend die Bearbeitung der Aufgaben im Unterricht zu erleichtern.
Wichtig ist, dass die Schüler anfangen, ihren Lernprozess selbst zu steuern. Und wenn es zunächst nur die Entscheidung ist, wie häufig sie ein erklärendes Video ansehen müssen, um in der Lage zu sein, die dazu passenden Aufgaben lösen zu können. Oder die Entscheidung, nach weiteren Screencasts zu fragen, weil bestimmte Zusammenhänge noch nicht klar genug sind, um eine Aufgabe bearbeiten zu können.
Der Lehrer auf der anderen Seite kommt mehr und mehr in die Rolle eines Beraters, der die Schüler darin unterstützt und berät, ihren Lernprozess zu gestalten. Er bereitet die Materialien und Inhalte so auf, dass die Schüler damit einfach arbeiten können (Screencasts, Arbeitsblätter und Skripte erstellen). Er übernimmt aber nicht die Verantwortung für den Lernprozess der Schüler; den tragen diese selbst.

Setzt man zusätzlich ein Skript ein, so ergeben sich neue Möglichkeiten, den Unterricht kompetenzorientiert zu gestalten, ihn zu öffnen und so auch die Eigenverantwortung der Schüler zu fördern.

Um die Schüler darin zu unterstützen, ihren Lernprozess eigenverantwortlich zu gestalten, helfen zusätzlich Schüler-Blogs, da der Schüler selbst Lernstrategien entwickeln und entdecken lernt. Zusätzlich hat der Lehrer – im Sinne seiner beratenden Tätigkeit - die Möglichkeit, Verständnisschwierigkeiten des Schülers zu entdecken und ihn dadurch bei der Korrektur der Inhalte zu unterstützen (vorausgesetzt, der Schüler hat im Blog in eigenen Worten formuliert und nicht die Gedanken anderer kopiert).
Die Kontrolle darüber sollte jedoch unbedingt auf Seiten des Schülers bleiben.

Es gibt eine Menge Unterrichtsszenarien, die sich ergeben, wenn man die einzelnen Komponenten, die hier vorgestellt wurden zusammensetzt.
Eine mögliche Variante könnte folgendermaßen aussehen:

Unterrichtsablauf

Vorbereitung für die Stunde

Die Schüler haben alle nötigen Materialien, so dass man eigentlich nichts vorbereiten muss.

Ein Schüler hat sich in der letzten Stunde ein zusätzliches, erklärendes Video zum Thema XYZ gewünscht. Ein anderer Schüler, der mit dem Skript bereits fertig ist, möchte unbedingt ein Projekt zum Thema ABC bearbeiten.

Das Video sollte bereits zu Hause vorbereitet worden sein und am besten schon auf dem eigenen YouTube Kanal (oder bei Vimeo oder auch draufhaberTV) online zur Verfügung stehen. In dem unterrichtsbegleitenden Lehrer-Blog ist das zusätzliche Video bereits vermerkt. Ebenso wurde dort angegeben, wie weit man im Skript sein müsste, um im Zeitplan zu liegen. Vielleicht ist auch schon die nächste Lernkontrolle und die dort abzuprüfenden Kompetenzen angegeben, damit die Schüler sich rechtzeitig darauf einstellen können und gegebenenfalls ihre Geschwindigkeit erhöhen können, wenn sie im Zeitplan „hinterher hängen"

Materialien zum Thema ABC, welches sich der Schüler, der schon fertig ist, gewünscht hatte, stehen ebenfalls im Blog bereit (falls Materialien vorhanden sind, ansonsten kann der Schüler natürlich eigenständig recherchieren).

Unterrichtsstunde –Start

Es wird kurz koordiniert, welche neuen Einträge im Lehrer-Blog wichtig sind (Materialien, Lernkontrolle, Zeitplan). Da man die Blogs, die die Schüler führen nicht alle auf einmal ansehen sollte (wegen des Arbeitsaufwandes), gibt man kurz an, welche Blogs man nach der Unterrichtsstunde (oder eventuell schon während der Stunde) zur Durchsicht haben möchte. Schüler, die eine Rückmeldung zu ihren Ergebnissen haben möchten, sollten immer die Möglichkeit haben, ihren Blog auch unaufgefordert abgeben zu können.

Unterrichtsstunde – Verlauf

Die Schüler arbeiten individuell an ihrem Skript. Eventuell schauen sie erklärende Videos während der Stunde, wenn sich dies im Verlauf der Skriptarbeit ergibt. Es haben sich schnell Teams gebildet, die gut miteinander arbeiten können – die Schüler können auch umher gehen, um Mitschüler, die schon weiter sind, um Rat zu fragen. Diese Vorgehensweise steigert natürlich auch die Sozialkompetenz der Schüler, indem sie sich gegenseitig helfen bzw. helfen lassen. Falls die Schüler Fragen haben, beantwortet man diese individuell im Zwiegespräch mit den Schülern.

Fragen, die von verschiedenen Schülern zum selben Thema gestellt werden, sollte man entweder in einem kurzen Lehrvortrag beantworten oder man bietet zur nächsten Stunde einen weiteren erklärenden Screencast an, den man zusätzlich anfertigt.

Eventuell bleibt in der Stunde Zeit, sich einzelne Blogeinträge der Schüler anzuschauen. Dann kann man direkt eine Rückmeldung geben und gegebenenfalls falsche Ergebnisse oder Konzepte direkt mit dem Schüler klären.

Unterrichtsstunde – Ende
Die Schüler können kurz berichten, was für Probleme sich ergeben haben, gegebenenfalls wird das dann über einen weiteren Screencast beantwortet. Wichtige Erkenntnisse oder auch Lösungsstrategien können ausgetauscht werden. Schüler, die direkt das Projekt bearbeiten, ohne dem Skript zu folgen, stellen Zwischenergebnisse vor oder geben Probleme an, die sich beim Bearbeiten der Projektaufgabe ergeben haben.

Unterrichtsstunde – Nachbereitung
Zu Hause oder auch in einer längeren Pause kann man den gewünschten Screencast zu dem Thema XYZ auf nehmen und ihn am besten direkt online zur Verfügung stellen. Kurz eine Notiz im Blog hinzufügen, dass das neue Material zur Verfügung steht und wo es zu finden ist (Videolink angeben).

Literaturverzeichnis

Bergmann, J. (kein Datum). *Flipped Classroom*. Von http://www.flippedclassroom.com/ abgerufen

Christian Spannagel, pädagogische Hochschule Heidelberg. (kein Datum). *Blog*. Von http://cspannagel.wordpress.com/?s=flipped abgerufen

Gerstein, J. (kein Datum). *User generated Educatoon*. Von http://usergeneratededucation.wordpress.com/2011/06/13/ abgerufen

Khan, S. (kein Datum). *Khans Academy*. Von http://www.khanacademy.org/ abgerufen

KMK. (6. 12 2012). *KMK Plenarsitzung*. Von http://www.kmk.org/presse-und-aktuelles/meldung/ergebnisse-der-plenarsitzung-der-kultusministerkonferenz-am-6-dezember-2012-in-bonn.html abgerufen

Sand, M. (kein Datum). *Sandbox*. Von http://digitalsandbox.weebly.com/flipped-classroom.html abgerufen

Hands-on-Informatik – Vermittlung objektorientierter Konzepte mit einem Funktionsmodell

StR Carsten Müller, Dr. Matthias Ehmann

Didaktik der Informatik
Universität Bayreuth
Universitätsstraße 30
95447 Bayreuth
carsten.mueller@uni-bayreuth.de
matthias.ehmann@uni-bayreuth.de

Abstract: Bei der Umsetzung von Konzepten der Objektorientierung wird häufig die Programmiersprache Java mit der Entwicklungsumgebung BlueJ eingesetzt. Dabei wird die Darstellung der aktuellen Objektzustände oft auf eine textbasierte Ausgabe auf der Konsole reduziert. Eine schönere, lebensnähere Darstellung mit Grafiken und Symbolen würde allerdings die Verwendung von Grafikbibliotheken erfordern, dabei rückt aber evtl. der wesentliche Lehrplaninhalt in den Hintergrund. Doch wie kann man objektorientierte Konzepte im Informatik-Unterricht für die Schüler "anschaulich" und "begreifbar" machen? Der Workshop zeigt, wie dies durch die Verwendung eines Funktionsmodells, das verschiedene Sensoren und Aktoren zur Verfügung stellt, erreicht werden kann.

1 Überblick

Zur Vermittlung objektorientierter Konzepte werden im Unterricht meist Themen aus der Erfahrungswelt der Schüler herangezogen, allerdings lassen rein softwarebasierte Beispiele, wie z. B. die Implementierung einer Ampelsteuerung in Java, keine „echte" Interaktion zwischen der Simulation und dem Schüler zu und erscheinen den Schülern oft nur als Mittel zum Zweck, um theoretische Inhalte zu verpacken. Die Darstellung der aktuellen Objektzustände wird dabei häufig auf eine textbasierte Ausgabe auf der Konsole reduziert. Eine schönere, lebensnähere Darstellung mit Grafiken und Symbolen würde die Verwendung von Grafikbibliotheken erfordern, dabei rückt aber oft der wesentliche Lehrplaninhalt in den Hintergrund.

Bei der Erstellung von (evtl. auch grafisch unterstützen) Simulationen ist eine „echte" Steuerung im Sinne von Interaktion meist auch nur eingeschränkt möglich, da hierfür nebenläufige Prozesse notwendig sind, um während der Ausführung eines Prozesses z. B. auf einen Tastendruck oder das Auslösen eines Sensors zu einer bestimmten Zeit zu reagieren. Die Vermittlung der Funktionsweise einer Steuerung reduziert sich in diesem Fall eher zu einer Ablaufprogrammierung, die dem Anspruch bei der Vermittlung von objektorientierten Konzepten nicht genügt. Gerade die Themenbereiche „Objekte und

ihre Beziehung“ oder „Generalisierung und Spezialisierung“ lassen sich auf diese Weise nur schwer vermitteln.

In diesem Workshop wird ein Hands-on-Projekt[1] vorgestellt, um objektorientierte Konzepte unter der Verwendung eines Funktionsmodells[2] zu vermitteln, ohne weitere Konzepte wie die Hardwareansteuerung in den Vordergrund zu stellen. Funktionsmodelle bieten den Schülern durch Aktoren (z. B. LED, Motor) die Möglichkeit, ein direktes Feedback auf ihre Programmierung zu geben, andererseits ermöglichen sie durch Sensoren (z. B. Taster, Helligkeitssensor) auch eine direkte Eingabe von Seiten der Schüler.

Viele Arbeiten zur Vermittlung von informatischen Inhalten beschäftigen sich bereits mit dem Einsatz von technischen Modellen wie LEGO Mindstorms Robotern [WB08] oder fischertechnik im Unterricht. Leider basieren diese häufig auf der Vermittlung der algorithmischen Grundstrukturen bzw. der Ablaufprogrammierung und sind oft nicht objektorientiert.

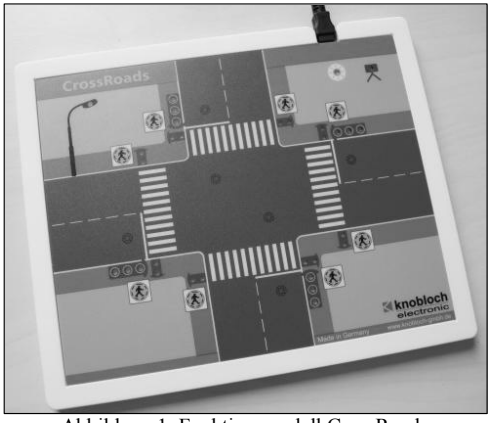


Abbildung 1: Funktionsmodell CrossRoads

Als exemplarisches Funktionsmodell kommt in diesem Workshop das Modell CrossRoads (siehe Abbildung 1, [KEPV]) einer Kreuzung zum Einsatz. Dieses Funktionsmodell stellt mit 30 LEDs (Lampen der Ampeln, einer Laterne sowie eines Blitzgeräts), acht Tastsensoren, sechs Magnetsensoren und einem Helligkeitssensor ein vielfältiges Angebot an Aktoren und Sensoren zur Verfügung. Es bildet die Kreuzung einer Haupt- und Nebenstraße nach, die an jeder Einmündung jeweils mit einer Straßenampel sowie zwei Fußgängerampeln mit entsprechenden Tastern zur Anforderung des Signals ausgestattet ist. Darüber hinaus bieten die digitalen Magnetsensoren in Kombination mit einem Automodell, das mit einem Magnet bestückt

[1] hands-on (engl.): aktiv, interaktiv, handlungsorientiert
[2] vereinfachte, maßstabgetreue Nachbildung eines oder mehrerer Geräte, die die wichtigsten Funktionen des Originals enthält

ist, die Möglichkeit, die „Position“ des Fahrzeugs zu ermitteln und darauf entsprechend zu reagieren.

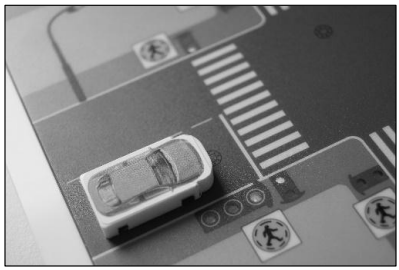


Abbildung 2: Magnetsensor (Schachtdeckel)

Außerdem steht ein analoger Helligkeitssensor zur Verfügung, dessen Werte z. B. dazu genutzt werden können, um die Ampelanlage in einen Tag- bzw. Nachtmodus zu versetzen. Bei der Auswahl dieses Funktionsmodells wurden mehrere Kriterien gegeneinander abgewogen: die Anschaulichkeit für die Schüler, die robuste Bauweise, das Angebot an Sensoren und Aktoren, die einfache Installation (Verbindung mit USB-Kabel, Treiberinstallation [FTKL]) und der Anschaffungspreis. Im Vergleich zu anderen Funktionsmodellen konnte CrossRoads eine ausgewogene Verteilung der oben genannten Kriterien erfüllen.

CrossRoads wurde ursprünglich für den Einsatz mit der Software ROBO Pro von fischertechnik [FRPS] entwickelt. Das dort verwendete ablauforientierte Konzept zur Implementierung von Algorithmen lässt keine objektorientierte Sichtweise zu.

Im ersten Teil unseres Hands-on-Projekt haben wir die technische Realisierung der Ansteuerung so angepasst, dass die Hardwareansteuerung in der Programmiersprache Java mit der Entwicklungsumgebung BlueJ [BlueJ] zur Verfügung steht, die häufig im Unterricht zur Implementierung von objektorientierten Konzepten eingesetzt wird. Ein weiteres Ziel war es, die Einbindung der externen Bibliotheken und der Treiber so zu gestalten, dass diese für die Schüler nicht sichtbar ist und somit keine weiteren Konzepte im Vordergrund stehen. Die dazu notwendigen DLL-Dateien und Anregungen zur Verwendung von CrossRoads mit Java wurden mit freundlicher Genehmigung von Herrn Ulrich Müller aus seinem ftComputing-Projekt [Müft] übernommen.

Als Voraussetzung für die Nutzung des Funktionsmodells mit BlueJ müssen neben den technischen Voraussetzungen (Treiberinstallation, 32-bit Java JDK) die von uns entwickelten JAR- und verwendeten DLL-Dateien in zwei Verzeichnisse des von den Schülern erstellten BlueJ-Projekts kopiert werden. Damit lässt sich die Hardwareansteuerung auch bei bereits von den Schülern erstellten BlueJ-Projekten problemlos „nachrüsten“.

2 Unterrichtskonzept

2.1 Inhaltliche Ausrichtung

Neben der Anpassung der Hardwareansteuerung wurde auch ein Unterrichtskonzept entwickelt, das von der Vermittlung der objektorientierten Grundbegriffe wie Objekt, Attribut, Methode usw. bis zur Einführung von Vererbung reicht.

Die inhaltliche Grundlage des Unterrichtskonzepts bildet der Lehrplan für die 10. Jahrgangsstufe am Naturwissenschaftlich-technologischen Gymnasium (NTG) in Bayern [BS03]. Die Grundbegriffe der Objektorientierung wie Objekt, Klasse, Attribut, Attributwert und Methode werden in der Jahrgangsstufe 6 im Fach Natur und Technik – Schwerpunkt Informatik eingeführt und in den Jahrgangsstufen 7 und 9 weiter vertieft. Ausgehend von der zustandsorientierten Sichtweise auf Objekte werden in der 10. Jahrgangsstufe Abläufe modelliert und anschließend in einer objektorientierten Programmiersprache (meist Java unter der Verwendung der Entwicklungsumgebung BlueJ) implementiert. Neben den Beziehungen von Objekten und deren Kommunikation reichen die Inhalte des bayerischen Lehrplans bis zum Thema Vererbung und schließen mit einem zehnstündigen Anwendungsbeispiel, bei dem die Schüler grundlegende Vorgehensweisen bei der Planung und Durchführung von Softwareprojekten erfahren sollen.

Als Modellierungsarten kommen Objekt- und Klassendiagramme, Zustandsübergangs- und Sequenzdiagramme sowie Struktogramme zum Einsatz. Die Inhalte lassen sich – auch auszugsweise – auf Inhalte der Lehrpläne anderer Bundesländer sowie in andere Schularten transferieren. Das Unterrichtskonzept kann begleitend während des Schuljahres als auch in einem mehrstündigen Projekt durchgeführt werden.

Das Hands-on-Projekt wurde so gestaltet, dass den Schülern nach Anschluss des CrossRoads-Boards via USB mit dem PC ein nicht sichtbares Objekt `hardware` der versteckten Klasse `Hardware` zur Verfügung steht. Als Dokumentation für die Schüler dient folgende Klassenkarte mit den entsprechenden Hinweisen zu den vordefinierten Methoden:

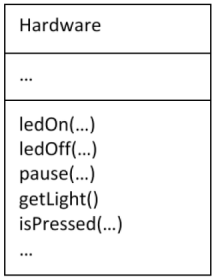


Abbildung 3: Klassenkarte der Klasse `Hardware`

Die Verwendung eines Funktionsmodells fordert – wie auch die reale Verkabelung und Ansteuerung einer Ampelanlage durch entsprechende Elektronik in einem Schaltungskasten – die Festlegung von Ein- und Ausgängen, an denen die Sensoren und Aktoren angeschlossen sind. Zur Erleichterung der Arbeit für die Schüler wurden die LEDs, Magnet- und Tastsensoren im Uhrzeigersinn durchnummeriert, außerdem steht den Schülern eine Schablone zur Verfügung, auf der sie schnell ablesen können, welche Nummer eines Ein- bzw. Ausgangs zu welchem Sensor bzw. Aktor gehört.

Die Hardware-Klasse stellt im Wesentlichen folgende Methoden zur Verfügung:

- `ledOn(ausgang)`: schaltet die LED am Ausgang `ausgang` ein.
- `ledOff(ausgang)`: schaltet die LED am Ausgang `ausgang` aus.
- `pause(ms)`: hält das Programm für `ms` Millisekunden an.
- `getLight()`: gibt den Wert des Helligkeitssensors zurück.
- `isPressed(eingang)`: gibt für den Taster mit der Nummer `eingang` einen Wahrheitswert zurück, ob der Taster aktuell gedrückt (`true`) bzw. nicht gedrückt (`false`) ist.
- `wasPressed(eingang)`: gibt für den Taster mit der Nummer `eingang` einen Wahrheitswert zurück, ob der Taster seit der letzten Abfrage mit `wasPressed(…)` gedrückt (`true`) bzw. nicht gedrückt (`false`) wurde. Nach der Abfrage eines Tasters wird sein Zustand auf `false` zurückgesetzt.
- `isContacted(eingang)`: gibt für den Magnetsensor mit der Nummer `eingang` einen Wahrheitswert zurück, ob der Sensor durch einen Magneten aktuell ausgelöst (`true`) bzw. nicht ausgelöst (`false`) ist.
- `loop(objekt, methodenname)`: startet die Methode mit dem Namen `methodenname` des Objekts `objekt` als parallelen Prozess und führt diese wiederholt aus.
- `listLoops()`: gibt die mit `loop(…)` gestarteten Prozessen mit ihrem Index und dem Methodennamen auf der Konsole aus.
- `stopLoop(methodenname)`: stoppt die wiederholte Ausführung der Methode mit dem Namen `methodenname`.
- `stopLoop(index)`: stoppt die wiederholte Ausführung der Methode mit dem Index `index`.
- `startClock()`: setzt die interne Stoppuhr auf Null zurück und startet die Stoppuhr.
- `stopClock()`: stoppt die interne Stoppuhr und gibt die verstrichene Zeit seit dem Stoppuhr-Start in Millisekunden zurück.

Gerade die loop-Methoden, die intern parallele Prozesse verwalten, bieten den Schülern der Mittelstufe die Möglichkeit, realitätsnahe Funktionen, wie die bedarfsgesteuerte Fußgängerampel, zu implementieren, ohne das theoretische Fundament bzw. die Implementierung von parallelen Prozessen in Java zu kennen.

2.2 Beispiele aus dem Unterrichtskonzept

Als Einstieg in die Arbeit mit dem Funktionsmodell bietet sich die Betrachtung einer realen Ampelanlage (z. B. auch als Foto) an, deren wesentliche Objekte identifiziert und mit einem Klassendiagramms modelliert werden sollen. Ein Ausschnitt eines möglichen Klassendiagramms könnte folgendermaßen aussehen:

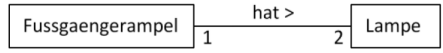


Abbildung 4: vereinfachtes Klassendiagramm einer Fußgängerampel

Ausgehend von den Straßen- und Fußgängerampeln (sowie später den Lampen der Laterne und des Blitzgerätes) wird zuerst die zustandsorientierte Sichtweise einer einzelnen Lampe genauer betrachtet:

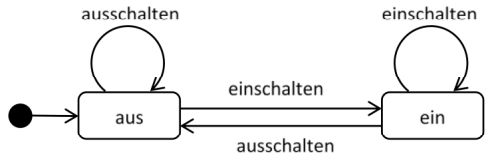


Abbildung 5: vollständiges Zustandsübergangsdiagramm einer Lampe

Eine unter diesem Aspekt erstellte Klassenkarte (ohne Datentypen und Sichtbarkeiten) könnte neben einem Attribut `zustand` vom Typ `String` die beiden Methoden `einschalten()` und `ausschalten()` beinhalten:

Abbildung 6: Klassenkarte der Klasse `Lampe`

Dies führt, zuerst ohne die Verwendung eines Funktionsmodells, beispielsweise zu folgender Implementierung in Java:

```
public class Lampe
{
    private String zustand;

    public Lampe() {
        zustand="aus";
    }

    public void einschalten() {
        zustand="ein"; // Zustandsänderung
```

```
    }

    public void ausschalten() {
        zustand="aus"; // Zustandsänderung
    }
}
```

Abbildung 7: Implementierung der Klasse Lampe in Java

Wie bereits oben beschrieben, muss zur Verwendung einer LED des Funktionsmodells die entsprechende Nummer des Ausgangs angegeben werden. Eingebunden in den zustandsorientierten Kontext ergibt sich daraus im Zustandsübergangsdiagramm eine Erweiterung um ausgelöste Aktionen:

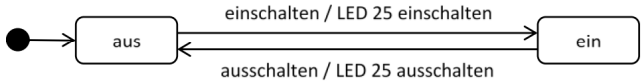


Abbildung 8: vereinfachtes Zustandsübergangsdiagramm mit ausgelösten Aktionen

Neben der Einführung eines Attributs `nummer` zur Verwaltung des zugehörigen Ausgangs müssen für die Ansteuerung des CrossRoads-Boards (symbolisiert durch das Objekt `hardware`) lediglich drei Zeilen ergänzt werden:

```
public class Lampe
{
    private String zustand;
    private int nummer;

    public Lampe(int nummer) {
        zustand="aus";
        this.nummer=nummer;
        hardware.ledOff(nummer);
    }

    public void einschalten() {
        if (zustand.equals("aus")) {
            zustand="ein"; // Zustandsänderung
            hardware.ledOn(nummer); // ausgelöste Aktion
        }
    }

    public void ausschalten() {
        if (zustand.equals("ein")) {
            zustand="aus"; // Zustandsänderung
            hardware.ledOff(nummer); // ausgelöste Aktion
        }
    }
}
```

Abbildung 9: Implementierung der Ansteuerung des Funktionsmodells

Zur Behandlung des Themenbereichs „Objekte und ihre Beziehungen“ bietet sich die Betrachtung der Ampeln an. Exemplarisch soll hier als Erweiterung der aufgezeigten Vorgehensweise die 1:3-Beziehung zwischen den Klassen `Strassenampel` und `Lampe` aufgezeigt werden:

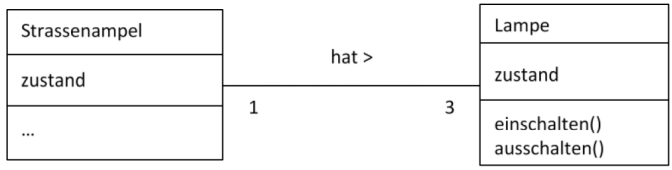


Abbildung 10: Modellierung einer 1:n-Beziehung

In diesem Fall ist die Vergabe einer internen Nummer für jede Straßenampel und die jeweilige Zuordnung der verwendeten Lampen notwendig. Dies stellt jedoch keine zusätzliche Information bzgl. der Hardwareansteuerung des Funktionsmodells dar, sondern dient lediglich zur Unterscheidung der einzelnen Elemente. Die Methode `weiterschalten()` führt dabei bei jedem Aufruf einen Schritt des Ampeldurchlaufs aus:

```
public class Strassenampel
{
    private Lampe rot;
    private Lampe gelb;
    private Lampe gruen;
    private String zustand;
    private int nummer;

    public Strassenampel(int nummer) {
        zustand="aus";
        this.nummer=nummer;
        switch (nummer) {
            case 1: { // Ampel links
                rot=new Lampe(3);
                gelb=new Lampe(4);
                gruen=new Lampe(5);
                break;
            }
            case 2: { // Ampel oben
                rot=new Lampe(10);
                gelb=new Lampe(11);
                gruen=new Lampe(12);
                break;
            }
            …
        }
    }
    …
    public void weiterschalten() {
        if (zustand.equals("aus")) {
```

```
            zustand="rot";
            rot.einschalten();
        }
        else {
            if (zustand.equals("rot")) {
                zustand="rot-gelb";
                gelb.einschalten();
            }
            else {
                if (zustand.equals("rot-gelb")) {
                    zustand="grün";
                    rot.ausschalten();
                    gelb.ausschalten();
                    gruen.einschalten();
                }
            …
        }
    }
}
```

Abbildung 11: Implementierung der Klasse `Strassenampel`

Zur vollständigen Modellierung bzw. Umsetzung einer Ampelsteuerung können noch weitere Klassen berücksichtig werden:

- Steuerungs-Klasse, die neben der Verwaltung der einzelnen Ampeln auch z. B. den Wert des Helligkeitssensors für den Tag- und Nachtmodus überwacht und die Blitzer-Steuerung übernimmt.
- Klassen zur Auswertung der Tast- und Magnetsensoren.
- Einführung einer abstrakten Klasse `Ampel` mit den Spezialisierungen `Fussgaengerampel` und `Strassenampel`, um den Themenbereich „Generalisierung und Spezialisierung“ abzudecken.

Vervollständigt man die Modellierung noch bzgl. der Tast- und Magnetsensoren, so könnte sich in BlueJ folgendes Projekt ergeben:

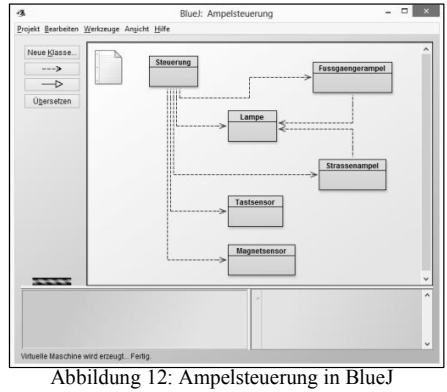


Abbildung 12: Ampelsteuerung in BlueJ

2.3 Weitere Anwendungsbeispiele

Das beschriebene Funktionsmodell kann neben der Verwendung als Kreuzungsmodell auch als Grundlage zur Umsetzung von Spielen eingesetzt werden, wenn man die aufgedruckte Kreuzung vernachlässigt.

Durch die in etwa kreisförmige Anordnung der LEDs und die Verwendung von Sensoren sind z. B. ein Roulette-Spiel oder einfache Geschicklichkeitsspiele – auch mit mehreren Spielern – umsetzbar. Diese Aufgaben bieten sich als Projekte an oder können auch Schülern an die Hand gegeben werden, die im Unterricht bei der Umsetzung der Ampelsteuerung schneller vorankommen und weitere Aufgaben benötigen.

2.4 Fazit

Das aufgezeigte Unterrichtskonzept wurde bereits an mehreren bayerischen Gymnasien in der Jahrgangsstufe 10 sowohl unterrichtsbegleitend als auch im Projektteil erfolgreich eingesetzt. Die eigenen Erfahrungen sowie die Rückmeldungen von Kollegen und Schülern deuten eindeutig darauf hin, dass gerade der Einsatz von enaktiven Modellen aus der Erfahrungswelt der Schüler neben der reinen Motivation durch den Einsatz des Modells und dessen Hands-on-Möglichkeiten auch ein tieferes Verständnis bei der Betrachtung von komplexen Beziehungen zwischen Objekten und ihrer Kommunikation fördert. Hier müssen fundierte wissenschaftliche Evaluationen den Erfolg noch bestätigen.

Literaturverzeichnis

[WB08] Wiesner, B, Brinda T.: Using Robots as Teaching Aids in Early Secondary Informatics Education. Proceedings of the Joint Open and Working IFIP Conference on ICT and Learning for the Net Generation, IFIP, 2008.

[KEPV] Knobloch Electronic Produktions- und Vertriebsgesellschaft mbH: CrossRoads. Erbes-Büdesheim.

[FTKL] fischertechnik: Treibersoftware KeLib für ROBO Interface Version 1.78a**,** http://www.fischertechnik.de/ResourceImage.aspx?raid=3125 (zuletzt aufgerufen am 28.07.2013). fischertechnik GmbH, Waldachtal.

[FRPS] fischertechnik: ROBO Pro Software, http://www.fischertechnik.de/desktopdefault.aspx/tabid-21/39_read-35/usetemplate-2_column_pano/ (zuletzt aufgerufen am 27.07.2013). fischertechnik GmbH, Waldachtal.

[BlueJ] Entwicklungsumgebung BlueJ für Java. http://www.bluej.org (zueletzt aufgerufen am 28.07.2013). La Trobe University, Australia, University of Kent at Canterbury, UK.

[Müft] Müller, U.: ftComputing, http://www.ftcommunity.de/ftComputingFinis/ (zuletzt aufgerufen am 27.07.2013), Paderborn, 2011.

[BS03] Bayerisches Staatsministerium für Unterricht und Kultus: Lehrplan für das Gymnasium in Bayern. Kastner AG, Wolnzach, 2003.

Smart und Reich durch App-Entwicklung

Programmieren in der Sekundarstufe mit App Inventor für Android-Smartphones

Ralf Romeike

Institut für Informatik / Didaktik der Informatik
Universität Potsdam
August-Bebel-Str. 89
14482 Potsdam
romeike@cs.uni-potsdam.de

Abstract: Mobiltelefone und Smartphones stellen für viele Jugendliche einen wichtigen Gegenstand im Alltag dar. Für viele Softwarefirmen besteht ein Großteil des Geschäfts inzwischen aus der Entwicklung mobiler Applikationen für Smartphones. Während das Entwickeln solcher Apps für Laien bisher noch zu komplex war, bietet App Inventor eine in ihrer Komplexität reduzierte Möglichkeit an, per Baustein-Programmierung Applikationen für Android-Smartphones zu entwickeln. Die Programmierumgebung orientiert sich dabei an Scratch und basiert auf der MIT Open Blocks Bibliothek. Applikationen können in Echtzeit auf einem angeschlossenen Smartphone oder dem mitgelieferten Emulator ausprobiert werden. Schnittstellen wie GPS, Beschleunigungssensor oder Webzugriff können einfach eingebunden werden. Selbst die Veröffentlichung der Applikationen im Android-App-Market ist möglich. Im Workshop werden Grundlagen, Möglichkeiten und Grenzen dieser Android-App-Entwicklung in der Schule besprochen sowie einfache Applikationen programmiert. Teilnehmer sind nach dem Workshop in der Lage, selbst Applikationen zu entwerfen und Unterrichtsideen mit App Inventor zu entwickeln. Der Einsatz von App Inventor in der Schule als Werkzeug verspricht smarte Schüler, die mit der richtigen Idee durchaus reich werden können - vielleicht nicht nur reich an Erfahrung....

1 Einleitung

Der Stellenwert des Programmierens im Informatikunterricht ist unbestritten. Aufgabe eines modernen Informatikunterrichts sollte es sein, die Schüler zu einem gestalterischen Umgang mit Informatiksystemen zu befähigen [Ro08a, Gu02, Re07, GI07]. Die technische Entwicklung ist inzwischen weiter fortgeschritten; Smartphones und Tablets werden nun mindestens die gleichen Möglichkeiten wie Computern zugeschrieben. Der PC als Informatiksystem hat an Bedeutung verloren [Ma03], [Sc03], [Sa06]. Vielmehr gibt es inzwischen diverse elektronische Kleingeräte wie Mobiltelefone, MP3-Player, Smartphones und mobile Spielkonsolen, die in der Alltagswelt zunehmend an Bedeutung gewinnen und den PC aus dem Fokus verdrängen (vergleiche Ubiquitous Computing, [We93]). Schulische Bildung darf sich keinesfalls gegenüber diesen Neuerungen ver-

schließen, sondern sollte vielmehr die damit verbundenen Themen aufgreifen und die Schülerinnen und Schüler mit ihrem Interesse an diesen Systemen dort abholen, wo sie stehen.

Laut der KIM-Studie 2006 umfasste die Verbreitung des Mobiltelefons „96 Prozent der Haushalte, in denen Kinder aufwachsen". In der JIM-Studie 2007 lag der Anteil des „Handy"-Gerätebesitzes von Jugendlichen bei 94% (95% der Mädchen/92% der Jungen). Aufgrund dieser Zahlen ist die Nutzung mobiler und allgegenwärtiger Informatiksysteme als Lernwerkzeug mit entsprechendem Content im schulischen Unterricht zu überdenken und mittels Pilotversuchen zu eruieren [CHH08]. Solches Mobiles Lernen wird als spontan, persönlich, tragbar und situativ beschrieben [KH05]. Potentiell wird die Verbreitung mobiler Lernszenarien steigen, da viele Eltern Smartphones und Tablets besitzen und diese ihren Kindern gern für eine sinnvolle Nutzung zur Verfügung stellen. Dies spiegelt sich unter anderem am Erfolg von Bildungsanwendungen in den App-Markets der verschiedenen Plattformen wider. Ebenso macht Kindern der Umgang mit solchen Medien Spaß.

Moderne Programmiersprachen wie App Inventor ermöglichen nun auch Anfängern, anspruchsvolle Anwendungen zu entwickeln. Durch die reduzierte Komplexität bei mobilen Anwendungen für Smartphones und Tablets eignen sich Entwicklungsumgebungen für diese Geräte in besonderer Weise, sinnvoll reduziert Konzepte und Anwendungen der Informatik zu verdeutlichen und zu vermitteln.

2 Ansätze zum Informatikunterricht mit Mobiltelefonen

Die Verbreitung von Mobiltelefonen als mobile und vor allem persönlich allzeit begleitende Informatiksysteme führte in der fachdidaktischen Diskussion zu Überlegungen, diese sowohl als Unterrichtsmittel als auch als Unterrichtsgegenstand im Informatikunterricht zu verankern. So widmete sich ein ganzes LOG IN Heft (Nr. 145 (2007)) dem Thema, Mobilkommunikation und dabei unter anderem der Frage: „Gibt es einen mobilkommunikationszentrierten Ansatz für die Schulinformatik"? Humbert wurde gar zitiert mit der Prognose: „Ich bin davon überzeugt, dass in zehn Jahren an den Schulen Informatik nur noch mit Mobiltelefonen erteilt wird". Im Gegensatz zu diesen Überlegungen, Inhalte und Werkzeuge der Schulinformatik durch Mobiltelefone zu ersetzen, versucht der vorgestellte Workshop mit App Inventor das Motivationspotential zu nutzen, welches sich aus der Entwicklung von Smartphone-Apps zur Gestaltung des eigenen Smartphones für die Schülerinnen und Schüler ergibt. Programmiert wird, auch aufgrund der ergonomischen Vorteile, am großen Computerbildschirm.

3 App Inventor

App Inventor ist eine graphische Softwareentwicklungsumgebung für Android-Anwendungen (vgl. Abb. 1). Ähnlich wie in Scratch oder Alice ist es bei App Inventor nicht möglich, Syntaxfehler zu machen.

Abb. 1.: Scratch (o.l.) im Vergleich mit App Inventor: Blocks Editor (u.l.), App Inventor Designer mit Komponenten (u.r.) und Emulator (o.r.)

Programmierte Anwendungen können einfach auf ein angeschlossenes Android-Smartphone oder Android-Tablet übertragen werden oder in einem im Paket inkludierten Emulator ausgeführt werden. Insbesondere die Einbindung von typischen Smartphone-Anwendungen (GPS, Maps, Kamera etc.) wird durch einfache Bausteine unterstützt.

App Inventor entspringt Google Labs. Die Idee war, durch einen leichteren Zugang zur Programmierung eigener Applikationen die Bindung vor allem junger Menschen an Mobiltelefone mit dem Android-Betriebssystem zu erhöhen. Der Zugriff auf App Inventor wurde dann aber zunächst mit Beendung von Google Labs Ende 2011 eingestellt. Im Weiteren wurden allerdings die zugrunde liegenden Dateien unter Open Source Lizenz veröffentlicht sowie die Überführung zum MIT Center for Mobile Learning unterstützt. Seit März 2012 ist App Inventor dort wieder öffentlich zugänglich.[1]

[1] Zur erreichen unter http://appinventor.mit.edu/.

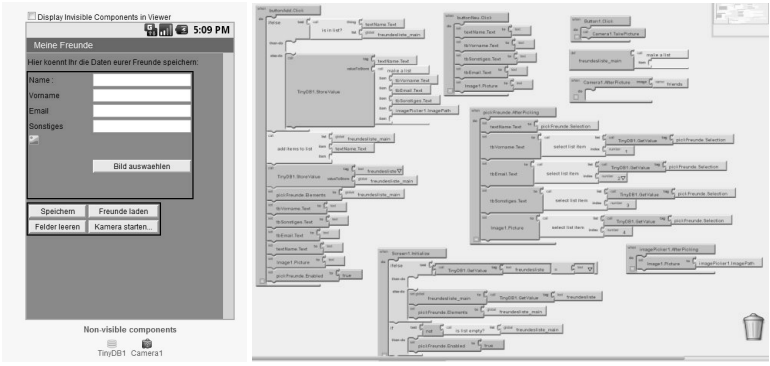


Abb. 2: Beispiel: Konzeption und Gestaltung der App „Meine Freunde“
mit App Inventor unter Nutzung einer Datenbank als Vorbereitung auf Phase 2.

App Inventor verwendet die Open-Blocks Java-Bibliothek zur Erstellung von grafisch-basierten Programmiersprachen des MIT, welche auch in z.B. StarLogo TNG verwendet wird.

Der Compiler übersetzt die grafischen Blöcke für die Implementierung auf Android. Genutzt wird die Entwicklungsumgebung Kawa und eine Kawa spezifische Abwandlung der Programmiersprache Scheme. Daraus ergibt sich leider, dass keine Möglichkeit existiert, Quellcode einzusehen, da kein Java-Code generiert wird.

4 Vergleich Scratch / App Inventor

Wie in Scratch wird mit App Inventor objektbasiert programmiert. Objekte in App-Inventor werden als Komponenten bezeichnet. App Inventor-Komponenten der Animation-Palette, genauso wie Scratch-Objekte, haben u. a. die in Tab. 1 dargestellten Attribute.

Scratch	**App Inventor**
• X-Position, Y-Position	• X,Y,Z (Z = Ebene)
• Größe	• Width/Heigth
• Kostüm	• Picture

Tab. 1: Attribute im Vergleich.

Die Attributwerte können jeweils in der GUI oder dynamisch per Programmierbaustein verändert werden:

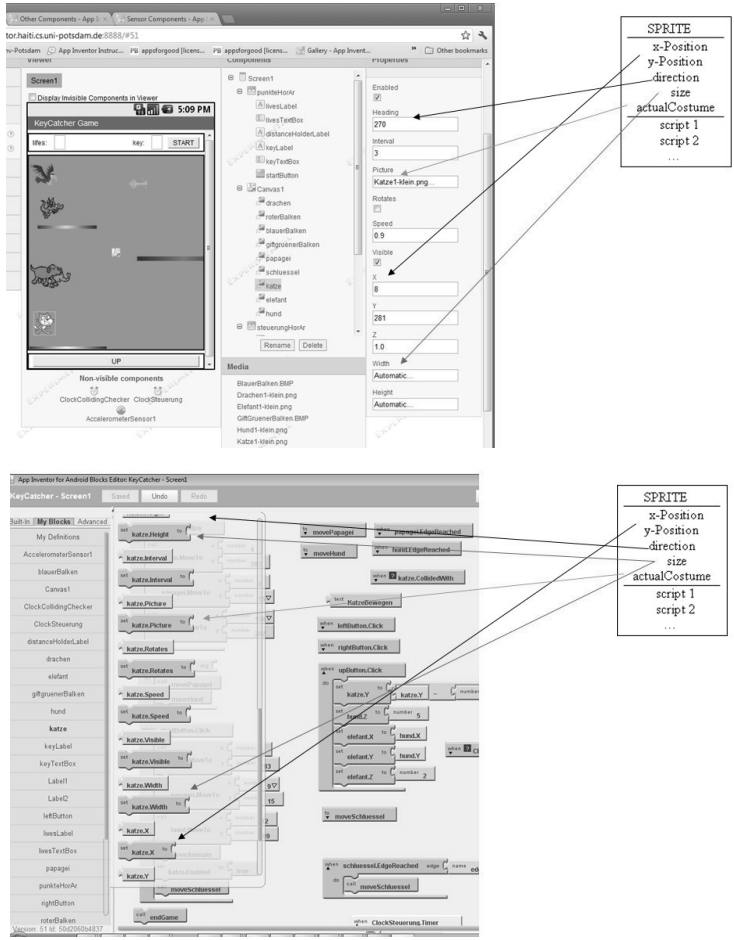


Abb. 3: Änderung von Attributwerten in App Inventor.

App Inventor, ebenso wie Scratch, ist eine eventbasierte Programmiersprache und verwendet hierfür die Punktnotation. Beispiele für Events sind screen.Initilize, hund.Touched, hase.Dragged, kugel.CollidedWith.

Daten können verarbeitet werden mit Variablen (polymorph), Prozeduren (mit und ohne Rückgabewert) und Listen.

5 Beispielprojekte:

Projekt 1: Hasenjagd

Ziel des Projekts ist es, die Grundfunktionen von App Inventor kennenzulernen und zugleich ein kleines Spiel zu entwickeln. Zuerst werden im App Designer eine Canvas und ein ImageSprite hinzugefügt, benannt und dem ImageSprite ein Hasenbild zugeordnet. Als nächstes kann im Blocks Editor das Programm entwickelt werden. Bei Berührung des Hasen soll diesem eine neue Position unter Verwendung einer Zufallszahl zugewiesen werden (Hase.X, Hase.Y). Als nächstes werden Sound und Vibration hinzugefügt, wobei zu beachten ist, dass erst nach Hinzufügen der Komponente Sound, die entsprechenden Blocks sichtbar werden. Schließlich wird mit der Komponente Clock ein Timer hinzugefügt, mit welcher im Timer-Takt Zufallsposition erzeugt werden. Ergänzend kann das Zählen von Punkten und einer Punkteanzeige hinzufügt werden (Variable deklarieren, Punkte auf Label-Komponenten anzeigen).

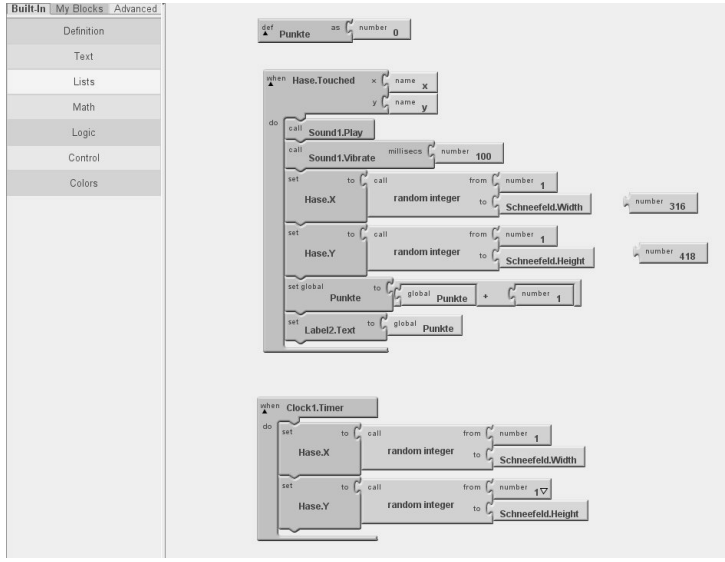


Abb. 4: App Hasenjagd im Blocks Editor.

Projekt 2: Kleine Datenbanken

In diesem Projekt soll der Umgang mit Daten mit den Komponenten TinyDB und TinyWebDB ausprobiert und dabei Grundlagen der persistenten Datenspeicherung mit App Inventor erlernt werden. Schülerinnen und Schüler kennen den Vorteil persistenter Daten aus Spielen: In Highscores werden die besten Spielstände gespeichert und mit dem aktuellen Ergebnis verglichen. Mit App Inventor ist dies möglich im Speicher des Smartphones per Speicherung in einer TinyDB oder in einer Cloud (per Internet) mit der Komponente TinyWebDB. Für die Nutzung der TinyWebDB empfiehlt sich die Einrichtung einer TinyWebDB-App bei Google AppSpot. Eine Beschreibung hierfür findet sich unter http://appinventorapi.com/program-an-api-python/. Als Voreinstellung ist zwar ein Testservice vorgegeben, der aber nur eine limitierte Anzahl von Einträgen zulässt und möglicherweise bald eingestellt wird (http://appinvtinywebdb.appspot.com/).

Die Verwendung der TinyWebDB ist denkbar einfach: *StoreValue* speichert einen Wert (numerisch, alphanumerisch, Listen) und einem gegebenen Schlüssel. *GetValue* ruft den unter einem bestimmten Schlüssel gespeicherten Wert ab.

Highscore	3
engList	["","one","two","three","four","five"]
deuList	["","eins","zwei","drei","vier","fuenf"]
Eins	One

Tab. 2: Beispiele für Daten in der TinyWebDB.

Sprachausgabe und Spracherkennung

Dank des in Android implementierten Sprachsystems ist es mit App Inventor einfach, gesprochene Sprache zu erkennen und Texte vorlesen zu lassen. Hierzu ist allenfalls die Vorgabe der Sprache notwendig, die sich witziger Weise auch dazu nutzen lässt, dem Sprachcomputer einen ausländischen Akzent zu verpassen. Ein einfaches Sprachdialogsystem ließe sich hiermit in wenigen Schritten verwirklichen.

Accelerometer-Sensor

Einem Smartphone fehlen, gegenüber traditionellen Computern, Tasten, mit denen Spiele und Applikationen gesteuert werden können. Natürlich lassen sich diese virtuell als Button auf dem Bildschirm des Smartphones darstellen und verwenden. Mehr Spaß kann es allerdings bedeuten, die eingebauten Sensoren des Smartphones zu verwenden und z. B. ein Spiel durch die Bewegung des Smartphones zu steuern. Hierdurch ergibt sich nicht nur ein ganz neuer Spielspaß für traditionelle Spiele, die Möglichkeit, Bewegungen zu erfassen und in der Spiellogik umzusetzen kann auch die Erfindung eigener neuer Spiele und anderer Apps anregen.

Kompass/Orientation-Sensor

Mit der Komponente *Orientation-Sensor* erhält man Orientierungsdaten des Smartphones zum Auswerten: Wie wird das Gerät gehalten? Welche Seite ist oben? Wo ist Norden? Ob Kompass oder entdeckendes Lernen im Physikunterricht - die Sensoren lassen sich für vielfältige Apps verwenden.

Webzugriff

Die wohl mächtigste Komponente stellt „Web" dar, mit welcher Funktionen für HTTP GET und POST zur Verfügung gestellt werden. Somit lassen sich nicht nur Internetseiten aus einer App heraus abfragen sondern z. B. auch Webservices einbinden, um im Web verteilte Daten zu verwenden.

6 Fazit

App Inventor bietet eine gute erste Möglichkeit, ohne detaillierte Programmierkenntnisse, Apps für Android Smartphones und Tablets zu entwickeln. Dabei kommt der Arbeit mit App Inventor zugute, dass keine Syntaxfehler gemacht werden können und Änderungen am Programm in Echtzeit auf dem Emulator oder angeschlossenen Smartphone betrachtet werden können. Für Schüler und Studenten ist die einfache Einbindung typischer Smartphonekomponenten und -sensoren sehr motivierend. Die Vielfältigkeit der Komponenten ermöglicht es zudem, ohne technischen Overhead Grundlagen moderner Informatiksysteme, z. B. der einfachen Datenspeicherung in der Cloud, Webservices oder Geolokalisierung im Unterricht praktisch zu thematisieren. Einschränkungen ergeben sich aus dem, im Vergleich zu Scratch, nicht ganz intuitiven User Interface sowie dem Fehlen der Möglichkeit, Klassen zu programmieren.

Erfahrungen bestätigen, dass die App-Entwicklung bei den Schülerinnen und Schülern sehr gut ankommt. Schnell entwickeln diese ihre eigenen Ideen und versuchen, diese umzusetzen.

Literaturverzeichnis

[CHH08] Ralf Carrie, Matthias Heming, Ludger Humbert: „Mobile Programming" auf Mobiltelefonen. IFFase (08) ,Hamm, Arnsberg 2008.

[GI07] Gesellschaft für Informatik e.V.: Grundsätze und Standards für die Informatik in der Schule. LOG IN Verlag, Berlin 2008.

[Gu02] Guzdial, M.; Soloway, E.: Teaching the Nintendo generation to program. In Commun. ACM 45(4), 2002; S. 17-21.

[KH05] Kukulska-Hulme, A. et al: Mobile learning : a handbook for educators and trainers, London, 2005.

[Ma03] Friedmann Mattern: „Allgenwärtiges Rechnen", LOG IN Heft Nr. 125, 2003

[Re07] Resnick, M.: Sowing the Seeds for a More Creative Society. Proc. Learning & Leading with Technology, International Society for Technology in Education (ISTE), 2007.

[Ro08] R. Romeike: Kreativität im Informatikunterricht. Dissertation, Universität Potsdam, 2008.

[Sa06] Martin Sauter: „Grundkurs Mobile Kommunikationssysteme. Von UMTS, GSM und GPRS zu Wireless LAN und Bluetooth Piconetzen“, Vieweg Verlag 2006

[Sc03] Jochen Schiller: „Mobilkommunikation“, Pearson Education, München, 2003, ISBN 3-8273-7060-4

[We93] Mark Weiser: „Some Computer Science Problems in Ubiquitous Computing”, Communications of the ACM, July 1993. (reprinted as „Ubiquitous Computing”. Nikkei Electronics; December 6, 1993; pp. 137-143.)

Kann man RSA (noch) vertrauen? Eine Unterrichtsskizze zur Behandlung der Sicherheit von RSA in der Sek I

Helmut Witten
Brandenburgische Str. 23
10707 Berlin
helmut@witten-berlin.de

Andreas Gramm
Gymnasium Tiergarten
Altonaer Str. 26
10555 Berlin
gramm@gymnasium-tiergarten.de

Abstract Wir skizzieren zunächst, wie das RSA-Verfahren als wichtigstes asymmetrisches Verschlüsselungsverfahren bereits in der Sek I behandelt werden kann. Dann gehen wir auf den Shared-Primes-Bug ein, der seit Beginn des Jahres 2012 in der internationalen Presse zu aufgeregten Befürchtungen führte, RSA sei nicht mehr sicher.[1] Leicht kann gezeigt werden, dass es sich um einen Implementierungsfehler handelt. Um diesen Fehler zu verstehen, genügt es, den Euklidischen Algorithmus nachzuvollziehen und anzuwenden. Damit kann auch in der Sek I an einem aktuellen Beispiel ein elementares Verständnis für die Komplexität von Algorithmen und das für die Sicherheit von RSA zentrale Faktorisierungsproblem angebahnt werden.

1 Einleitung

RSA, das wichtigste und für den E-Commerce unverzichtbare asymmetrische Kryptosystem, galt in der fachdidaktischen Diskussion vielfach als zu schwierig für die Behandlung in der Sek I. Wir wollen in Abschnitt 2 und 3 Wege aufzeigen, dies dennoch zu tun, ausführlich ist dies in [WEGH11] und [WEGH12] geschehen. Das zentrale Hilfsmittel dabei ist das mehrfach ausgezeichnete Programm CrypTool[2], mit dem wir den Algorithmus sozusagen verstecken. Der verbleibende mathematische Anteil reduziert sich damit auf das Minimum, das man zum Verständnis der asymmetrischen RSA-Verschlüsselung benötigt: Primzahlen, Semiprimzahlen sowie die (Un-)Möglichkeit der Faktorisierung.

Hier werden wir den Fokus auf die Sicherheit von RSA legen und Möglichkeiten zeigen, wie sich die Lernenden selbstständig von der Bedeutung einer ausreichenden Schlüssellänge überzeugen. Allerdings liefert die Schlüssellänge nur eine notwendige, aber keineswegs hinreichende Voraussetzung für die Sicherheit der RSA-Verschlüsselung. Bei dem „shared primes bug“ (Abschnitt 4) werden die Primzahlen nicht wirklich zufällig ausgewählt, mehrfach verwendete Primzahlen machen die entsprechenden Moduln leicht angreifbar.

[1] Vgl. dazu New York Times [Ma12] bzw. in Deutschland ZEIT [St12]. Zur Entwarnung durch Krypto-Experten siehe [Ka12], [He12], [ESS12], [HDWH12].
[2] http://www.cryptool.org/de/

Für diesen Angriff benötigt man lediglich den Euklidischen Algorithmus zur Ermittlung der gemeinsam genutzten Primzahlen. Eine ausführliche Sachanalyse zum Thema Sicherheit von RSA mit Vertiefungsmöglichkeiten für die Sek II kann unter [WS12] nachgelesen werden.

2 Wie funktioniert die asymmetrische Kryptographie?

Die grundlegende Idee für die asymmetrische Kryptographie wurde von Whitfield Diffie und Martin Hellman bereits vor 35 Jahren veröffentlicht [DH78]. Der Hauptgedanke ist dabei, dass man nicht den gleichen Schlüssel zum Ver- und Entschlüsseln nutzt, sondern zwei: einen öffentlichen Schlüssel zum Verschlüsseln (den public key) und einen geheimen Schlüssel (den private key) zum Entschlüsseln. Damit ist es nicht mehr notwendig, geheime Schlüssel zu verteilen; es genügt vielmehr, die öffentlichen Schlüssel in einer Art Telefonbuch zu einzutragen. Bob kann dort nachschlagen, wie der Schlüssel von Alice lautet, seine Nachricht damit verschlüsseln, die dann Alice und nur Alice mit ihrem geheimen Schlüssel wieder entschlüsseln kann.

Mit der asymmetrischen Kryptographie wurde aber zugleich ein weiteres, zentrales Problem gelöst: Wie kann Alice sicher sein, dass die angeblich von Bob stammende Nachricht wirklich von ihm ist und nicht von der bösen Eve, da jede/r den öffentlichen Schlüssel von Alice nachschlagen und einfach behaupten kann, dass er/sie Bob sei?

Die Lösung heißt Signatur oder digitale Unterschrift: Bob erzeugt einen „Fingerabdruck" seiner Botschaft, den sog. Hashwert, eine Zahl, die charakteristisch für seine Nachricht ist. Diesen verschlüsselt Bob mit seinem geheimen Schlüssel und verbindet dies mit seiner mit Alice öffentlichem Schlüssel verschlüsselten Botschaft. Alice kann dann mit Bobs öffentlichem Schlüssel nachprüfen, dass wirklich er es war, der unterschrieben hat und anschließend die Nachricht selber mit ihrem geheimen Schlüssel wieder entschlüsseln. So kann sie nachprüfen, ob die Botschaft von Bob vertraulich und authentisch übermittelt werden konnte.

In dieser vereinfachten Übersicht fehlt noch ein Hinweis auf die erforderliche Schlüssel-Infrastruktur PKI[3]. Eine solche PKI stellt sicher, dass der öffentliche Schlüssel von Alice auch wirklich von ihr stammt. Die PKI kann streng hierarchisch organisiert sein („Trustcenter"). Ein Gegenmodell dazu ist das „web of trust", das im Zusammenhang mit der Software PGP entwickelt wurde (http://de.wikipedia.org/wiki/Web_of_Trust).

Zur Einführung in die asymmetrische Kryptographie verwenden wir das E-Learning-Programm „Nachrichtenverschlüsselung und Digitales Signieren", das vom Institut für Telematik[4] entwickelt wurde. Diese interaktive Einführung kann von den Lernenden

[3] public key infrastructure, vgl. http://de.wikipedia.org/wiki/Public-Key-Infrastruktur

[4] Das Institut für Telematik e. V. mit dem Gründungsdirektor Christoph Meinel war 1996-2002 in Trier unter Betreuung der Fraunhofer-Gesellschaft mit Fragestellungen aus dem Gebiet der Internet- und Web-Technologien befasst. Prof. Meinel ist inzwischen Direktor des Hasso-Plattner-Instituts an der Universität Potsdam und hat freundlicherweise gestattet, das an seinem Institut entwickelte Programm auf dem Cryptoportal für Lehrer (https://www.cryptoportal.org/) allen Interessierten zur Verfügung zu stellen.

verwendet werden, um sich die Begriffe Integrität, Authentizität, Verbindlichkeit und Vertraulichkeit ganz ohne Zahlen und Mathematik selbstständig zu erarbeiten (Bild 1). [5]

Alternativ zu diesem Programm kann auch die Animation von Andreas Gramm verwendet werden, die wir in [WEGH12], S. 83ff. ausführlich beschrieben haben. Diese Animation realisiert die asymmetrische Verschlüsselung im Hintergrund mit dem Krypto-System RSA, wobei kleine Zahlen verwendet werden, die sich auch für die händische Simulation von RSA eignen. Wir empfehlen, die Entscheidung, welche der beiden Animationen verwendet werden soll, von dem beabsichtigten Unterrichtsgang abhängig zu machen. Wenn auch der eigentliche RSA-Algorithmus erarbeitet werden soll bietet sich die Animation von Andreas Gramm an, wenn dagegen nur das Prinzip der asymmetrischen Verschlüsselung und ggf. die Sicherheit von RSA im Vordergrund stehen sollen, ist im Sinn der Konzentration auf das Wesentliche das Programm des ehemaligen Instituts für Telematik vorzuziehen.

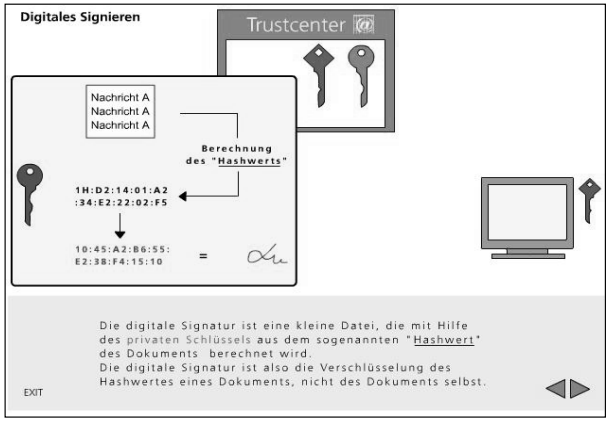


Bild 1: Bildschirmfoto „Digitales Signieren“ aus der Animation des Instituts für Telematik

Für das Durcharbeiten dieses Selbstlernprogramms des Instituts für Telematik werden je nach Leistungsfähigkeit der Lerngruppe 20-40 Min. benötigt.

3 Die Realisierung asymmetrischer Kryptographie mit RSA

Um ein Schlüssel-Paar (öffentlich und privat) erstellen zu können, benötigt man eine Einweg-Funktion mit Hintertür (auch „Falltürfunktion“ genannt). Eine Einwegfunktion muss einfach zu berechnen sein, für die Bestimmung der Umkehrfunktion benötigt man aber zusätzliche Informationen, man muss gewissermaßen die Hintertür kennen. Ohne diese Kenntnis ist es praktisch unmöglich, den inversen Schlüssel zu ermitteln.

Beim RSA-Verfahren[6] wird die Falltürfunktion durch sog. „harte“ Semiprimzahlen realisiert. Semiprimzahlen sind jeweils ein Produkt aus genau zwei Primzahlen. Die Multipli-

[5] Alternativ können die Prozesse einer PKI (Schlüssel generieren, Zertifikat beantragen, Request prüfen, und genehmigen, Zertifikat erstellen, widerrufen und sperren) auch mit dem PKI-Plugin in der Lernsoftware JCrypTool (JCT) durchgespielt werden: Menü Visualisierungen → Public-Key-Infrastruktur.
Die Java-Version von CrypTool erhält man unter: http://www.cryptool.org/de/jct-downloads-de/jct-downloads-weekly-de

kation auch sehr großer Zahlen ist ein algorithmisch einfaches Problem, während die Faktorisierung eines Produktes von zwei annähernd gleich großen Primzahlen i. a. ein schweres Problem ist [SW10]. Wenn dieses Produkt besonders schwer zu faktorisieren ist, nennt man die entsprechende Semiprimzahl „hart“. Die Hintertür ist dabei schlicht die Kenntnis der Primfaktoren.

Im Unterricht gilt es zunächst, die Begriffe „Primzahl“ und „Semiprimzahl“ zu erarbeiten. Darüber hinaus sollte bei den Lernenden eine Idee entstehen, wie man im Prinzip Primzahlen finden und Nicht-Primzahlen in die Primfaktoren zerlegen kann. Das Verfahren der Wahl hierzu ist das Sieb des Eratosthenes. Die Lernenden können sich diese Begriffe und das Sieb des Eratosthenes mit einem Arbeitsbogen erarbeiten. Er findet sich z. B. in [WEGH12] mit weiteren Hinweisen zu den Aufgaben auf S. 86f. Dort wird auch auf die animierte Einführung des Siebs von Hans-Bernhard Meyer verwiesen, die das „Sieben“ dynamisch veranschaulicht[7]. Mit einer kleinen Erweiterung des Siebs (man merkt sich die Zahl, mit der die jeweilige Nicht-Primzahl ausgesiebt wurde) eignet sich das Verfahren auch zur Faktorisierung kleinerer zusammengesetzter Zahlen.

Auf dem Arbeitsbogen wird kurz die RSA-Challenge erwähnt. Dies ist eine Sammlung von harten Semiprimzahlen, die erst zu einem geringeren Teil faktorisiert wurden[8]. Der aktuelle Faktorisierungs-Rekord vom 12.12.2009 ist die Zahl RSA-768 mit 232 Dezimalstellen bzw. 768 Bit Länge. Mit dieser (inzwischen beendeten) Challenge wollte die Firma RSA verdeutlichen, wie sicher RSA ist. Tatsächlich ist die übliche Länge von RSA-Schlüsseln z. Zt. 1024 Bit und bereits seit einiger Zeit wird empfohlen, Schlüssel mit 2048 Bit Länge zu verwenden. Bislang ist es noch niemandem gelungen, eine harte Semiprimzahl mit 1024 Bit Länge zu faktorisieren, eine Faktorisierung eines 2048-Bit-Schlüssels liegt daher noch in weiter Ferne.

Es gibt in der Literatur und im Netz sehr viele Quellen, die das RSA-Verfahren anschaulich erklären[9]. Ein wirkliches Verständnis erfordert aber einige Kenntnisse aus der elementaren Zahlentheorie (z. B. den Satz von Euler-Fermat zum Beweis der Korrektheit der RSA-Verschlüsselung), die nicht Schulstoff sind und eine gewisse Routine im modularen Rechnen voraussetzen. Nach unseren Erfahrungen kann dies in der Oberstufe in vertretbarer Zeit erarbeitet werden. Für die Sek I schlagen wir einen anderen Weg vor: Der RSA-Algorithmus wird in der Software CrypTool „versteckt“, damit man Zeit gewinnt, sich der wichtigen Frage nach der Sicherheit von RSA zu widmen. Dieses Problem ist sicherlich für alle Bildungsgänge interessant und sollte daher bereits in der Sek I thematisiert werden.

Hierfür hat sich die RSA-Demo aus CrypTool 1[10] hervorragend bewährt: Nach Installation und Start des Programms lautet der Aufruf
`Einzelverfahren → RSA-Kryptosystem → RSA-Demo`… Arbeitsbögen zum Ver- und Entschlüsseln mit RSA sind in [WEGH12], S. 88f abgedruckt. In der RSA-Demo

[6] Siehe [RSA78] bzw. http://de.wikipedia.org/wiki/RSA-Kryptosystem
[7] http://www.hbmeyer.de/eratosib.htm
[8] Siehe http://en.wikipedia.org/wiki/RSA_Factoring_Challenge und [WS12] sowie die dort zitierte Literatur.
[9] Siehe z. B. http://www.cryptool.org/images/ct1/presentations/RSA/RSA-Flash-de/player.html
[10] http://www.cryptool.org/de/cryptool1

sind verschiedene Verfahren zur Faktorisierung eingebunden. So kann sehr anschaulich vermittelt werden, dass die Faktorisierung des Moduls **n** gleichbedeutend mit dem Brechen der RSA-Verschlüsselung ist; der Arbeitsbogen „RSA knacken mit CrypTool“ findet sich in der eben genannten Quelle auf S. 90. Die Lernenden erhalten die Möglichkeit, durch eigene Experimente herauszufinden, wie lang ein RSA-Schlüssel sein muss, damit er mit der aktuell zur Verfügung stehenden Technik (PC mit CrypTool 1 im Jahr 2013) nicht faktorisiert werden kann. Diese Zahl liegt mit 315 Bit deutlich unter der heute üblichen Länge von 1024 Bit, von der neuesten Empfehlung, 2048 Bit zu verwenden ganz zu schweigen. Wie wir bereits in der Einleitung geschrieben haben, ist eine ausreichende Schlüssellänge nur eine notwendige, aber leider keine hinreichende Voraussetzung für die Sicherheit von RSA.

4 Recycling von Primzahlen – nein danke!

Bei der sicheren Kommunikation im Internet werden für die RSA-Moduln in Aber-Milliarden von Transaktionen riesige Mengen von Primzahlen benötigt, die sich alle unterscheiden müssen. Denn wenn Primzahlen mehrfach verwendet werden, ergeben sich Sicherheitslücken, die Anfang des Jahres 2012 für den eingangs erwähnten großen Wirbel in der internationalen Presse gesorgt haben. Diese Lücke gilt es jetzt genauer zu untersuchen und zu verstehen.

Wenn wir die für die Zukunft empfohlenen 2048 Bit-Schlüssel nehmen, benötigt man für einen Schlüssel zwei 1024 Bit-Primzahlen als Faktoren der Semiprimzahl. Wie viele Primzahlen mit 1024 Bit gibt es?

Diese Frage lässt sich mit dem Primzahlsatz von Gauß beantworten: Es gibt mindestens $1{,}262449 \cdot 10^{305}$ Primzahlen der Länge 1024 Bit (s. [WS10], S. 98ff.). Die Zahl der Atome im gesamten Universum wird auf weniger als 10^{80} geschätzt, die Zahl der Elementarteilchen auf 10^{87}. Somit ist es physikalisch unmöglich, eine Liste aller 1024 Bit-Primzahlen zu erstellen. Diese Riesenzahl hat zu der falschen Annahme geführt, dass es bei einer zufälligen Auswahl einer Primzahl praktisch unmöglich sei, zweimal die gleiche Zahl zu wählen. Erst 2012 haben Arjen Klaas Lenstra und seine Mitstreiter sowie unabhängig davon Bernhard Esslinger und Nadia Heninger untersucht, ob wirklich unterschiedliche Primzahlen verwendet werden.[11] Aber wie kann man herausfinden, dass in zwei unterschiedlichen Semiprimzahlen der gleiche Primfaktor versteckt ist? Dazu könnte man jede dieser Zahlen faktorisieren, aber das ist ja – wie wir oben ausgeführt haben – bislang praktisch unmöglich. Der geniale Gedanke der Forscher war es, nicht zwei Semiprimzahlen **einzeln** zu zerlegen, sondern den größten gemeinsamen Teiler (ggT) von **zwei** Moduln zu bestimmen. Ist dieser 1, so ist alles gut, andernfalls hat man einen gemeinsam genutzten Primfaktor („shared prime“) gefunden; damit sind **beide** Moduln gebrochen.

[11] Siehe [LHA12], [ESS12]. Zu den Hintergründen siehe auch [WS12], zu neueren Untersuchungen siehe [HDWH12] bzw. https://factorable.net/.

Für diese Untersuchung wurde der Euklidische Algorithmus verwendet, seit 2½ Jahrtausenden bekannt und gern genutzt – auch im Informatik-Unterricht, um z. B. eine iterative mit einer rekursiven Implementierung zu vergleichen. Bei der Untersuchung des Shared-Primes-Bug spielt das extrem unterschiedliche Laufzeitverhalten der verwendeten Algorithmen eine zentrale Rolle: Während an eine Faktorisierung einer Semiprimzahl in der heute verwendeten Größenordnung nicht zu denken ist, erfolgt die Untersuchung zweier Moduln auf gemeinsame Primfaktoren mit Hilfe des Euklidischen Algorithmus unglaublich schnell. In [ESS12] werden Python- und Shell-Skripte zur Verfügung gestellt, die in ungefähr 13 Stunden 1 Million Moduln auf gemeinsam genutzte Primzahlen prüfen. Das erfordert fast 1 Billion Überprüfungen[12]! Wenn noch mehr Moduln getestet werden müssen, kann man vom CrypTool-Projekt auch eine noch schnellere C++-Version beziehen.

5 Euklid Reloaded

In [WS06] wurde der Euklidische Algorithmus vorgestellt, allerdings mit einer etwas anderen Zielsetzung: Es ging hauptsächlich um den Erweiterten Euklidischen Algorithmus (EEA), der dazu dient, die modulare Inverse zu berechnen. Im Kontext RSA wird der EEA zur Schlüsselgenerierung bzw. bei bekannter Primfaktorzerlegung des Moduls zur Berechnung des geheimen Schlüssels benötigt. Für die Sek I überlassen wir dies wiederum der Software CrypTool. Auch von dem Verständnis her stellt der EEA deutlich höhere Anforderungen als der (einfache) Euklidische Algorithmus, den wir im Folgenden noch einmal kurz im Hinblick auf den Einsatz in der Sek I vorstellen wollen.

Der Euklidische Algorithmus ist der älteste bekannte nicht-triviale Algorithmus. Er findet sich in den „Elementen“ (Buch VII, Proposition 1 und 2, Veröffentlichung um 300 v. Chr.). Das Verfahren wurde jedoch wahrscheinlich nicht von Euklid erfunden, sondern war schon bis zu 200 Jahre früher bekannt.[13]

Euklid nannte das Verfahren „Wechselwegnahme“ und formulierte es als geometrisches Problem: „Wenn CD aber AB nicht misst, und man nimmt bei AB bzw. CD abwechselnd immer das Kleinere vom Größeren weg, dann muss (schließlich) eine Zahl übrig bleiben, die die vorangehende misst“.[14] Diese Wechselwegnahme wird schrittweise durch das Programm **`ggT-step.exe`** von Cornelia Niederdrenk-Felgner visualisiert, das vor vielen Jahren mit dem DIFF-Studienbrief „Algorithmen der elementaren Zahlentheorie“ verteilt wurde[15]. Die folgenden Bildschirmfotos von **`ggT-step`** zeigen, dass

[12] Die Schätzung 1 Billion ergibt sich aus der folgenden groben Näherung $(10^6)^2 = 10^{12}$. Es sind bei n Schlüsseln aber nicht n^2, sondern genau $\binom{n}{2} = \frac{n(n-1)}{2}$ Vergleiche erforderlich – also deutlich weniger, aber doch von der gleichen Größenordnung. Durch den Aufbau von Produkt- und Restebäumen kann die Anzahl der Vergleiche weiter verringert werden.

[13] Vgl. auch Wikipedia, Stichwort „Euklidischer Algorithmus“

[14] In der Geometrie kann es vorkommen, dass die Wechselwegnahme nicht terminiert – wenn nämlich die Streckenlängen in einem irrationalen Verhältnis stehen, z. B. bei den Kanten und Diagonalen eines Quadrats http://de.wikipedia.org/wiki/Inkommensurabilit%C3%A4t_(Mathematik). Uns interessieren im Zusammenhang mit RSA nur natürliche Zahlen – da terminiert der EA immer, spätestens mit 1.

[15] Frau Niederdrenk-Felgner hat als Programmautorin das DOS-Programm **`ggT-step.exe`** freundlicherweise zur Nutzung für den Unterricht und für Fortbildungen zur Verfügung gestellt, es steht im Cryptoportal für Lehrer (https://www.cryptoportal.org/) mit einer Installationsanleitung und einem

das Programm das Prinzip anschaulich und dynamisch darstellt. Beim Start sind die Zahlen 48 und 18 voreingestellt, die einfach auch mit anderen Werten überschrieben werden können.

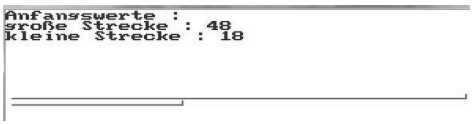


Bild 2: Bildschirmdarstellung von ggT-step (Teil1)

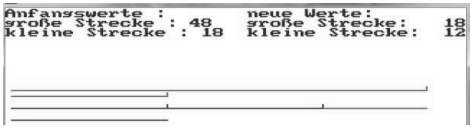


Bild 3: Bildschirmdarstellung von ggT-step (Teil2).
Die kleine Strecke mit der Länge 18 wird solange es geht von der großen Strecke weggenommen. Dann wird die kleine Strecke zur neuen großen Strecke und der Rest (12) wird die neue kleine Strecke.

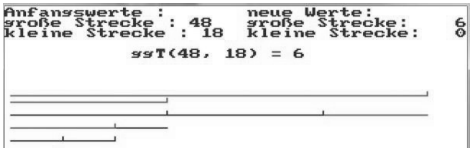


Bild 4: Bildschirmdarstellung von ggT-step (Teil3). Dieser Vorgang wird so lange wiederholt, bis die neue kleine Strecke zu Null geworden ist.

Die Schülerinnen und Schüler arbeiten selbstständig mit dem Programm unter Anleitung eines Arbeitsbogens, den wir im Folgenden in Auszügen abdrucken.[16]

Arbeitsaufträge

1. Gib jeweils die in der Tabelle angegebenen Zahlenpaare[17] in das Programm ein: (48,18), (23,17), (85,51).

2. Lass das Programm schrittweise ablaufen und trage die Zwischenergebnisse in die Tabelle ein.

3. Formuliere den Algorithmus in einzelnen Arbeitsanweisungen und teste mit Deinem Nachbarn, ob die Anweisungen funktionieren.

Arbeitsbogen zum Herunterladen zur Verfügung. Dieses Programm läuft unter dem Betriebssystem DOS, das von moderneren Windows-Versionen u. U. nicht mehr unterstützt wird (Bis WinXP konnte das Programm direkt von der Kommandozeile gestartet werden). Eifrige Programmierer haben schon vor einiger Zeit mit der DOSBox (http://de.wikipedia.org/wiki/DOSBox) Abhilfe geschaffen, die unter allen gängigen Betriebssystemen lauffähig ist. Für Windows-Systeme gibt es außerdem ein benutzerfreundliches Frontend, das die Konfiguration erleichtert und in einem Paket mit der DOSBox heruntergeladen und einfach installiert werden kann (http://dfendreloaded.sourceforge.net/). Ab September 2013 ist diese Visualisierung auch in dem Plattform-unabhängigen JCT (JCrypTool RC7) und in dem Vektorgrafik-basierten CT2 (Download unter http://www.cryptool.org/de/ct2-download-de) enthalten.

[16] Wenn genügend Zeit zur Verfügung steht, können sich die Lernenden zusätzlich mit dem 2-Personen-Spiel „Euclid“ zur Einübung der Wechselwegnahme beim Euklidischen Algorithmus beschäftigen; Siehe [CD69].

[17] Aus Platzgründen sind hier die Lösungen bereits kursiv eingetragen. Ebenfalls aus Platzgründen sind die weiteren Zahlenpaare des Arbeitsbogens hier weggelassen.

groß	klein	
48	**18**	
18	*12*	
12	*6*	
6	*0*	

ggT(48,18) = 6

groß	klein	
23	**17**	
17	*6*	
6	*5*	
5	*1*	
1	*0*	

ggT(23,17) = 1

groß	klein	
85	**51**	
51	*34*	
34	*17*	
17	*0*	

ggT(85,51) = 17

An dieser Stelle muss in einem Unterrichtsgespräch zunächst geklärt werden, dass hier die Division mit Rest zu Grunde liegt, wobei die Division als fortgesetzte Subtraktion realisiert wird. Aufschlussreich sind die Zahlenketten, die sich bei den drei Durchläufen ergeben:

48 > 18 > 12 > 6 > 0 mit dem ggT **6**;
23 > 17 > 6 > 5 > 1 > 0 mit dem ggT **1**;
85 > 51 > 34 > 17 > 0 mit dem ggT **17**.

Man sieht, dass alle Glieder der Kette Vielfache des jeweiligen ggT sind. Alle Ketten sind streng monoton fallend und enden bei Null, das letzte, von Null verschiedene Glied gibt den ggT an.

Diese Ketten entstehen durch fortgesetzte Division mit Rest:
$a = q \cdot b + r$ mit $a > b > r \geq 0$ sowie $q \geq 0$. Aus dieser Definitions-Gleichung ergibt sich, dass eine Zahl t, die a und b teilt, auch r teilen muss, weil man t ausklammern kann, wenn man die Gleichung nach r auflöst. Umgekehrt teilt jede Zahl, die b und r teilt, auch a. Somit pflanzt sich die Teiler-Beziehung durch die ganze Kette fort. Die Maximalität des ggT wird plausibel, wenn man sich vor Augen führt, dass bei der Abarbeitung der Wechselwegnahme die Zahlen nach jedem Durchlauf immer kleiner werden und man den ggT erhält, wenn die Division zum ersten Mal aufgeht, der Algorithmus also den größtmöglichen gemeinsamen Teiler liefert.

Bei Aufgabe 3 geht es um die anspruchsvollste Fragestellung auf diesem Arbeitsbogen: Wie kann man aus dem beobachteten Vorgehen der Wechselwegnahme einen Algorithmus formulieren, der von einer Maschine ausgeführt werden kann? Möglicherweise beschreiben die Schüler zunächst die fortgesetzte Subtraktion. Man wird dann in einem zweiten Schritt herausarbeiten, dass es eigentlich nur auf den verbleibenden Rest ankommt und man sich das mehrmalige Subtrahieren sparen kann, wenn man eine Funktion zur Verfügung hat, die den Divisionsrest direkt liefert.[18]

Wir verwenden der Einfachheit halber die Skript-Sprache Python[19], weil sie es als interpretierende Sprache gestattet, Programmfragmente oder Funktionen sehr einfach interaktiv zu testen. Für die kryptologischen Anwendungen kommt hinzu, dass sie über eine

[18] Das ist in allen gängigen Programmiersprachen der Fall, entweder **`mod`** bei den Sprachen aus der Pascal-Familie oder der Prozentoperator `%` für die meisten anderen Sprachen mit C-ähnlicher Syntax.
[19] http://www.python.org , Version 2.x

eingebaute Langzahlarithmetik verfügt – sehr praktisch für eine RSA-Implementierung! Darüber hinaus ist Python 2.x weitgehend mit dem Open Source Computer Algebra-System SAGE kompatibel (http://www.sagemath.org/).

Wir erläutern kurz die imperative Lösung[20]. Der Arbeitsbogen enthält für jedes Zahlenpaar drei Spalten, weil wir drei Variablen benötigen werden. Was wird in der dritten Spalte stehen? Nach der Logik des Algorithmus kann in dieser Spalte eigentlich nur der Rest stehen, der sich bei der Division der Zahl in der ersten Spalte durch die Zahl in der zweiten Spalte ergibt. Dann fällt im nächsten Iterationsschritt die Zahl in der ersten Spalte fort und die beiden anderen Zahlen werden eine Zeile nach unten und eine Spalte nach links verschoben.

Wenn man dies in Python notiert und die Variablen in den drei Spalten a_alt, a_mitte und a_neu tauft, ergibt sich die folgende Python-Funktion:

```
def ggT (a,b):
    "Berechnung des ggT mit dem euklidischen Algorithmus."
    a_alt = a
    a_mitte = b
    while a_mitte <> 0:
        a_neu = a_alt % a_mitte
        a_alt = a_mitte
        a_mitte = a_neu
    return a_alt
```

Wir testen diese Python-Funktion mit realistischen Moduln n_1 und n_2 der Länge 1024 Bit, die wir so erzeugt haben, dass sie eine gemeinsame Primzahl verwenden:[21]

```
>>> n1 =
34732254778223769415055150461891551268786606723240696518819899162877746574
62925646684517235136643203584799050871730775354924656268001511597190959262
69818293732323467262365610616945310947166315144208138858916410025740671044
33064095784975954648834566806136726396322929684907977758226027669561230325
2640679480791L[22]

>>> n2 =
30516265235959074574047098774694808751925662383066789206189923155656828206
10871876852448688095271763396707376999488148216714194044073715864732486985
18701768671474180839002033680407365173398040282481329067008227535239946744
85463265001080431495843549385618221601538369661608261369575149152148945324
3616765868141L
```

`ggT(n1,n2)` liefert praktisch unmittelbar nach dem Aufruf den in den beiden Moduln versteckten gemeinsamen 512 Bit-Primfaktor:

```
13931663684190457165277107136142136655125628604926985024762644970078828945
97189059737179677894601445675840091942392072338387993798345511563520837037
4785889L
```

[20] Eleganter ist natürlich die rekursive Variante im funktionalen Programmierstil (s. [WS06], S. 52), aber für die Sek I weniger geeignet.
[21] Was man ihnen allerdings nicht ansieht! Vgl. [WS12], S. 51
[22] Mit dem L am Ende der Zahl zeigt der Python-Interpreter an, dass es sich um eine lange Integer-Zahl handelt.

Die beiden anderen Faktoren erhält man jeweils durch eine einfache Division. Wir haben also in für uns nicht messbarer Zeit die zwei 1024 Bit-Moduln n_1 und n_2 faktorisiert – ein Kunststück, das bislang mit keiner normalen harten Semiprimzahl mit einer Standard-Faktorisierung gelungen ist!

Die Untersuchungen der Kryptologie-Expertin Nadia Heninger und anderen ([He12], [HDWH12]) zeigen, dass solche schwachen Schlüssel durch Netzwerkgeräte wie Router und Firewalls mit schlechtem Zufallsgenerator[23] erzeugt wurden. Der letztgenannte Artikel enthält – ebenso wie [ESS12] – Empfehlungen, wie man diese Fehler vermeiden kann.

Danksagung: Wir danken Cornelia Niederdrenk-Felgner und Christoph Meinel für die Bereitstellung zentraler Tools für die Unterrichtsreihe; Bernhard Esslinger, Ralph-Hardo Schulz und Astrid Witten danken wir für wichtige Verbesserungsvorschläge zu diesem Artikel.

Literaturverzeichnis

[CD69] Cole, A. J.; Davie, A. J. T.: A Game Based on the Euclidean Algorithm and a Winning Strategy for It. In: The Mathematical Gazette, Vol. 53, No. 386 (Dec., 1969), S. 354-357. http://www.jstor.org/stable/3612461

[ESS12] Esslinger, B.; Simon, V.; Schneider, J.: RSA-Sicherheit in der Praxis – Fehler in der Anwendung des RSA-Algorithmus. In: <kes> – Die Zeitschrift für Informations-Sicherheit, 28. Jg. (2012), Heft 2, S. 22–27. http://www.cryptool.org/images/ctp/documents/kes_2012_RSA_Sicherheit.pdf

[DH78] Diffie, W.; Hellman, M. E.: New Directions in Cryptography. In: IEEE Transactions on Information Theory. 22. Jg. (1978), Nr. 6, S. 644–654. http://www.cs.jhu.edu/~rubin/courses/sp03/papers/diffie.hellman.pdf

[He12] Heninger, N.: New research: There's no need to panic over factorable keys – just mind your Ps and Qs. 15. Februar 2012. https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs/

[HDWH12] Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J. A.: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices https://factorable.net/weakkeys12.extended.pdf

[Ka12] Kaminsky, D.: Primal Fear – Demuddling The Broken Moduli Bug. 17. Februar 2012. http://dankaminsky.com/2012/02/17/primalfear/

[LHA12] Lenstra, A. K.; Hughes, J. P.; Augier, M.; Bos, J. W.; Kleinjung, Th.; Wachter, Chr.: Ron was wrong, Whit is right. Cryptology ePrint Archive – Report 2012/064. 14./17. Februar 2012. http://eprint.iacr.org/2012/064

[Ma12] Markoff, J.: Flaw Found in an Online Encryption Method. In: The New York Times, 14. Februar 2012. http://www.nytimes.com/2012/02/15/technology/researchers-find-flawin-an-online-encryption-method.html?_r=1

[23] Die gängigen Zufallsgeneratoren arbeiten intern deterministisch. Aus einer Startzahl (seed) wird eine pseudozufällige Zahlenfolge generiert. Man kann das auch beim Zufallsgenerator von CrypTool 1 beobachten: Bei jedem Aufruf nach dem Neustart des Programms wird bei vorgegebenem Wertebereich die gleiche Zahlenfolge erzeugt.

[RSA78] Rivest, R. L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. http://people.csail.mit.edu/rivest/Rsapaper.pdf

[St12] Strassmann, B.: Zufällige Sicherheit. In: DIE ZEIT, 67. Jg., Nr. 19 vom 3. Mai 2012, S. 37, und ZEIT-ONLINE – Datenschutz (05.05.2012). http://www.zeit.de/2012/19/N-zufaellige-Zahlenreihen/komplettansicht

[SW10] Schulz, R.-H.; Witten, H.: Zeit-Experimente zur Faktorisierung – Ein Beitrag zur Didaktik der Kryptologie. In: LOG IN, 30. Jg. (2010), Heft 166/167, S. 107–114. http://bscw.schule.de/pub/bscw.cgi/d864899/Schulz_Witten_Zeit-Experimente.pdf

[WEGH11] Witten, H.; Esslinger, B., Gramm, A., Hornung, M.: Asymmetrische Kryptographie für die Sek I – RSA (fast) ohne Mathematik? In: Weigend, M., Thomas, M., Otte, F.: Informatik mit Kopf, Herz und Hand – Praxisbeiträge zur INFOS 2011. Zentrum für Lehrerbildung, Münster 2011, S. 224-234. http://bscw.schule.de/pub/bscw.cgi/d938724/RSA_fast_ohne_Mathematik.pdf

[WEGH12] Witten, H.; Esslinger, B., Gramm, A., Hornung, M.: Kann man RSA vertrauen? Asymmetrische Kryptographie für die Sekundarstufe I. In: LOG IN Heft Nr. 172/173, 2011/2012, S. 79-92. http://bscw.schule.de/pub/bscw.cgi/d1024037/RSA_Sek1.pdf

[WS06] Witten, H., Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge Teil 2: RSA für große Zahlen In: LOG IN Heft Nr. 143, 2006, S. 50-58. http://bscw.schule.de/pub/bscw.cgi/d404410/RSA_u_Co_NF2.pdf

[WS10] Witten, H., Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge Teil 4: Gibt es genügend Primzahlen für RSA? In: LOG IN Heft Nr. 163/164, 2010, S. 97-103. http://bscw.schule.de/pub/bscw.cgi/d864891/RSA_u_Co_NF4.pdf

[WS12] Witten, H., Schulz, R.-H.: RSA & Co. in der Schule – Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. Neue Folge Teil 6: Das Faktorisierungsproblem oder: Wie sicher ist RSA? In: LOG IN Heft Nr. 172/173, 2011/2012, S. 59-69. http://bscw.schule.de/pub/bscw.cgi/d1024013/RSA_u_Co_NF6.pdf[

Messung der Pedagogical Content Knowledge (PCK): Schülerkognition

Laura Ohrndorf

Lehrstuhl für Didaktik der Informatik und E-Learning
Universität Siegen
Hölderlinstraße 3
57076 Siegen
laura.ohrndorf@uni-siegen.de

Abstract: Im Rahmen dieses Forschungsvorhabens soll die Schülerkognition von Lehrern modelliert und gemessen werden. Diese Kompetenz ist im Informatikunterricht von besonderer Bedeutung, da Informatiklehrer häufig mit sehr inhomogenen Lerngruppen konfrontiert sind, deren informatische Vorkenntnisse stark differieren. Dies erfordert eine besondere Beachtung eventuell herausgebildeter Präkonzepte, sowie eine differenzierte Analyse von Schülerfehlern.

1 Hintergrund

Shulman beschreibt in seiner Kategorisierung des Lehrerwissens das fachdidaktische Wissen (Pedagogical Content Knowledge, PCK) wie folgt: "It represents the blending of content and pedagogy into an understanding of how particular topics, problems, or issues are organized, represented and adapted to the diverse interests and abilities of learners." [Sh01].

Das wissenschaftliche Interesse an der Messung dieses Wissens hat in der fachdidaktischen Forschung in den letzten Jahren stark zugenommen. Verschiedene Projekte haben dieses Wissens modelliert und mit Hilfe von Testinstrumenten umgesetzt. In diesem Kontext findet auch die sogenannte Schülerkognition Beachtung, welche die Fähigkeit beschreibt, Schülermeinungen und -fehler zu erkennen, zu analysieren und ggf. zu korrigieren. Erfahrungen in der informatikdidaktischen Ausbildung von Lehrerinnen und Lehrern zeigen, dass häufig ein Verständnis für fehlerhafte Konzepte und Interpretationen von Schülerinnern und Schülern fehlt. Im Gegensatz zu verwandten Fachdidaktiken (z.B. Mathematik und Physik), findet diese Kompetenz in der Informatikdidaktik nur wenig Aufmerksamkeit.

2 Projektdurchführung

Das Forschungsvorhaben sieht die folgenden Arbeitsschritte vor:

1. Theoriebildung und Review

Die Untersuchung der Schülerkognition war Teil vieler fachdidaktischer Projekte und Studien wie z.B. COACTIV (Mathematik) [Kr08]. Diese haben zu vielfältigen Ergebnissen geführt, die auch in der Informatikdidaktik von großem Interesse sind. So zeigte sich u.a. in mehreren Studien, dass auch Studierende in fachwissenschaftlichen Studiengängen äquivalente oder sogar bessere Leistungen in fachdidaktischen Tests zeigen.

Aufgrund der kurzen Geschichte des Schulfachs Informatik und dem dadurch bedingten Fehlen an informatikdidaktischen Materialien und Konzepten nimmt die Ausbildung von Informatiklehrern zudem eine Sonderstellung ein.

2. Entwurf von Testitems

Basierend auf dem Testinstrument des informatikdidaktischen Projekts MoKoM, mit welchem die Kompetenz von Schülern gemessen wurde, wurden Aufgaben entworfen, die zur Messung der Schülerkognition eingesetzt werden können. Zur inhaltlichen Strukturierung wird auf eine im COACTIV Projekt eingesetzte Kategorisierung zurückgegriffen. Mit dem Bereich "PCK task" werden Testitems bezeichnet, die das Wissen über die Formulierung und damit verbundene Probleme von Aufgaben prüfen. "PCK student" bezieht sich auf die Reaktion und Denkweise von Schülerinnen und Schülern und umfasst damit auch Präkonzepte, die bereits vorab ausgebildet wurden. [OH01]

Abbildung 1 zeigt ein Testitem der Kategorie PCK student (Item ST01), welches zunächst eine Multiple-Choice-Aufgabe darstellt, in der die Schülerinnen und Schüler entscheiden sollen, welcher Algorithmus die Addition der Zahlen 0 bis n effektiver implementiert. Aufgabe des Testteilnehmers ist es nun, mögliche Gründe aufzuführen, die zu einer fehlerhaften Antwort des Schülers führen können.

Correct	Algorithm

	`Enter: n`
	
	`Set sum = 0`
	`Set i = 0`
	`Repeat from 0 to n`
	`  Set sum = sum + i`
	`  Set i = i + 1`
	
	`Return sum`

	`Enter: n`
	
	`Set sum = 0`
	`If n odd-numbered, then`
	`  Set sum = sum + n`
	`  Set n = n - 1`
	`Set sum = (n/2) * (n+1)`
	
	`Return sum`

Abbildung 1: Testitem PCK student ST01

3. (Vor-)erprobung

Im Rahmen einer Vorerprobung wird das Instrument im Wintersemester 2013/2014 in informatikdidaktischen Vorlesungen an drei Universitäten eingesetzt werden, um die Validität und Reliabilität zu überprüfen. Die Ergebnisse dieser Erhebung werden zur Überarbeitung und gegebenenfalls Neuerstellung von Testitems genutzt werden. Die tatsächliche Erhebung wird in Form eines Power-Tests im Jahr 2014 durchgeführt werden. Zielgruppe werden hier neben den Studierenden des Lehramts Informatik auch ausgebildete Lehrer sein. Die bearbeiteten Items werden anschließend durch zwei Experten-Ratings analysiert und bewertet.

4. Fazit

Die Ergebnisse werden eine grundlegende Vorstellung über die Ausbildung und Entwicklung der Schülerkognition geben. Von besonderem Interesse ist hier, wie diese mit weiteren Faktoren (z.B. der Vorerfahrung oder persönlichen Interessen) zusammenhängt.

Literaturverzeichnis

[Kr08] S. Krauss, J. Baumert, and W. Blum. Secondary mathematics teachers' pedagogical content knowledge and content knowledge: validation of the COACTIV constructs. Zdm, 40(5):873–892, Oct. 2008.

[Oh01] Ohrndorf, Laura; Schubert, Sigrid; Measurement of pedagogical content knowledge: students' knowledge and conceptions. In: Proc. of The 8th Workshop in Primary and Secondary Computing Education, 2013 (to appear).

[Sh87] L. S. Shulman. Knowledge and teaching: Foundations of the new reform. Harvard educational review, 1987.

Autorinnen und Autoren