Informatik - EIN/AUS - Bildung

Werner Hartmann

Pädagogische Hochschule Bern
CH-3000 Bern 9
werner.hartmann@phbern.ch

Abstract: Wer nicht lesen, schreiben, rechnen und mit Informationstechnologien umgehen kann, steht heute wie ein Esel am Berg vor einem Fahrkartenautomat, bei der Online-Reservation eines Hotels und erst recht beim Ausfüllen der elektronischen Steuererklärung. Der hohe Stellenwert der Informationstechnologien in unserer Gesellschaft ist unbestritten. Dennoch gibt es keinen Konsens über den Stellenwert der informatischen Bildung. Trotz vieler Anläufe hat sich im Bildungskanon kein Schulfach Informatik etablieren können und generell wird der Computer im Unterricht nicht im erwarteten Maße als Hilfsmittel genutzt. Warum wird der Bildungswert der Informatik verkannt? Schaffen es die Informatiker nicht, den Bildungswert der Informatik genügend zu kommunizieren? Müssten die Inhalte und die Methoden des Informatikunterrichtes neu überdacht werden? Die nachfolgenden Gedanken sind ein Plädoyer für einen Informatikunterricht mit einem stärkeren Bezug zum Alltag der Lernenden, für eine vermehrte Gewichtung methodischer Aspekte in der Informatik-Didaktik und für eine verstärkte Zusammenarbeit mit den Lehrpersonen anderer Fächer.

1 Braucht es wirklich ein Schulfach Informatik?

Lesen, Schreiben und Rechnen sind heute allgemein anerkannte Kulturtechniken und Grundlage jeder Allgemeinbildung. Und natürlich wissen wir alle, dass es nicht genügt, in der Schule nur das Einmaleins zu üben oder Prozentsätze von irgendwelchen Zahlen ausrechnen zu können. Wer nie ein tieferes Verständnis für die Prozentrechnung erworben hat, wird sich als Tourist nichts ahnend über's Ohr hauen lassen, wenn der Kellner zuerst 5% für das Gedeck, auf dem resultierenden Total 15% für den Service und auf dem neuen Total noch 5% für die Mehrwertsteuer draufschlägt. Mathematische Allgemeinbildung bedingt auch ein Verständnis für die einem Thema zugrunde liegenden Konzepte. Unbestritten ist heute, dass die Informatik ebenfalls zur Allgemeinbildung gehört. Die Informatik hat unsere Gesellschaft im Laufe weniger Jahrzehnte verändert, wie kaum eine andere Wissenschaft zuvor. Fast jedermann nutzt Anwendungen wie Textverarbeitung oder Web-Browser. Um diese Computer-Werkzeuge bedienen zu können, benötigt man im Gegensatz zu früher keine Programmierkenntnisse mehr. Die effiziente und effektive Nutzung dieser Werkzeuge ist aber keineswegs einfach. Es genügt nicht, im richtigen Moment die richtige Taste zu drücken.

Die Komplexität von Informatiksystemen im Alltag wird einem bewusst, wenn man die Gebrauchsanweisung des neuen Einbaubackofens verlegt hat. Als gewiefter Internet-Surfer sucht man schnell mittels der Begriffe „Einbaubackofen", „Herstellername“ und „Gebrauchsanweisung“ auf Google, erhält 17 Treffer, aber allesamt aus dem Bereich

von Schnäppchenangeboten. Mehr Erfolg zeigt die Suche auf der Website www.hersteller.de. Nach 8 Klicks findet man wirklich ein Auswahlmenu für unzählige Gebrauchsanweisungen dieses Herstellers. Mittels einer getrennten Suche auf der Dokumentenkollektion des Herstellers bringt man auch in Erfahrung, welche Bedeutung die Abkürzung PNC hat: Produktnummerncode. Im Backofen findet sich die Produkt-Nr. 944 211 794 und damit kann die Gebrauchsanweisung als PDF-Datei der Größe 429 KB herunter geladen werden. Falls nicht schon installiert, gibt es auch einen Download Button für den Adobe Acrobat Reader. Auf Seite 11 findet sich die Übersicht der Bedien- und Anzeigeelemente (Abbildung 1).

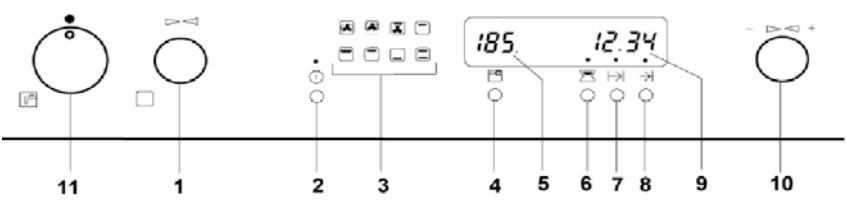

Abbildung 1: Bedien- und Anzeigelemente Einbaubackofen

Ohne das Studium der Gebrauchsanweisung ist es praktisch unmöglich, dieses Bedienfeld zu durchschauen. Beim Schalter 11 ganz links handelt es sich um eine analoge Drehschalter-Umsetzung der diskreten zweiwertigen Funktion „Feuchtigkeitszufuhr ja / nein". Beim Druckknopf 4 könnte man zuerst erwarten, dass sich damit die Temperatur regeln lässt. Hiermit lässt sich aber nur die Temperatur abfragen. Die diskrete Temperaturanzeige in Panel 5 wird durch den analogen Drehknopf 10 gesteuert. Die verschiedenen Startzustände im Panel 3 des endlichen Automaten „Einbaubackofen" (Heißluft, Pizzastufe, Turbogrill,...) werden durch den Drehknopf 1 ausgewählt. Wir gehen hier nicht ausführlicher ins Detail.

Das Beispiel spricht für sich: ohne Computerfertigkeiten, ohne Kenntnis wichtiger Werkzeuge oder Dateiformate und ohne eine zumindest intuitive Vorstellung endlicher Automaten lässt sich selbstständig kaum online die benötigte Gebrauchsanweisung finden, geschweige denn ein Kuchen backen. Die Frage ist: Wie gelingt es, solche komplexe „Informatiksysteme" zu beherrschen? Würde eine ausreichende informatische Bildung diesen Problemen vorbeugen? Braucht es dazu ein Schulfach Informatik? Und vermitteln wir heute im Informatikunterricht die benötigten grundlegenden Konzepte und Kompetenzen?

Das Beispiel „Backofen" illustriert zweierlei. Erstens steht es mit der Ausbildung der „Backofen-Ingenieure" nicht zum Besten, sonst würden nicht die grundlegenden Design-Prinzipien in solch sträflicher Art und Weise verletzt. Der heutige Informatikunterricht auf Stufe Ingenieurausbildung ist offensichtlich noch unzureichend und berücksichtigt zum Beispiel die Thematik der Mensch-Maschinen-Schnittstellen zu wenig.

Zweitens hilft das, beispielsweise im heutigen Informatikunterricht auf Stufe Gymnasium, vermittelte Wissen Anwenderinnen bei Backofen-Problemen wenig. Der gymnasiale Informatikunterricht vermittelt den Anwendern von Informatiksystemen nicht die im

Alltag benötigte informatische Ausbildung. Daraus den Schluss zu ziehen, es brauche kein Schulfach Informatik, wäre aber verfehlt. Vielleicht müsste der Informatikunterricht nur anders gestaltet werden.

2 Schulfach Informatik: Was läuft schief?

Soll „Backofenkunde“ im Informatikunterricht Eingang finden? Sicher nicht: Ein auf die Vermittlung kurzlebiger Anwenderfertigkeiten ausgerichteter Informatikunterricht ist nicht nachhaltig. Wer nur das Backofenmodell 2005 bedienen kann, wird beim Modell 2007 wieder fast von vorne beginnen müssen. Soll informatische Bildung vermittelt werden, muss der Unterricht auch auf langlebiges Konzeptwissen ausgerichtet sein. Die Vielfalt und die rasante Entwicklung der Informatik machen es aber nicht leicht, die allgemein bildenden Grundlagen der Informatik zu bezeichnen. Das Anzweifeln des Stellenwertes der Informatik als allgemein bildendes Fach rückte zudem die Legitimation eines Schulfaches Informatik und die möglichen Inhalte eines solchen Faches in den Mittelpunkt der Diskussion. So entstanden Lehrplan-Dokumente mit einem Umfang von mehr als hundert Seiten, mit hohen Zielsetzungen und oft außer Reichweite der Vorkenntnisse vieler Informatiklehrpersonen. Die zu behandelnden Inhalte folgen zudem oft einer strengen Fachsystematik und der Bezug zum Alltag ist für die Schülerinnen und Schüler nicht ersichtlich. Zur Illustration: In den Einheitlichen Prüfungsanforderungen zur Abiturprüfung (EPA) der Kultusministerkonferenz vom 12.03.2004 wird als Aufgabenbeispiel die Entwicklung eines abstrakten Datentypen (ADT) *Dictionary* angeführt. Teil der Aufgabe ist das Beschreiben unten stehender Dictionary-Methode:

```
01 private int gibIndex(String schluessel) {
02   DictionaryElement element;
03   int i = 0;
04   while (i < size() ) {
05     element = (DictionaryElement) get(i);
06     if (element.schluessel.equals(schluessel) )
07       return i;
08     i++;
09     }
10   return -1;
11 }
```

Die Aufgabenstellung entspricht den Anforderungen im ersten Studienjahr eines Informatikstudiums und mag geeignet sein, um gegenüber Informatik-Experten den Tiefgang des Informatikunterrichtes zu dokumentieren. Für die meisten Schülerinnen dürfte ein Anwendungsbezug aber nur schwer herstellbar sein, für Kolleginnen aus anderen Unterrichtsfächern bleibt der Bildungswert solcher Aufgaben schleierhaft. Eine Verknüpfung zum eigenen Unterricht in Biologie oder Geschichte ist unvorstellbar.

Kann es sein, dass der Informatikunterricht hier den gleichen Irrweg beschreitet, wie vor einigen Jahrzehnten der „neue“ Mathematikunterricht? Mit „neuer Mathematik“ war die Mengenlehre gemeint. Anstelle von Fertigkeiten und Fähigkeiten des Rechnens als solches wurde das abstrakte Denken in den Vordergrund gesetzt. Die fachwissenschaftlich orientierte Mengenlehre entsprach aber nicht den pädagogischen und didaktischen An-

forderungen, war nicht schülergerecht und führte zu einer Verwissenschaftlichung des Mathematikunterrichtes. Nicht nur die Schülerinnen waren überfordert, auch die Lehrpersonen wurden durch diesen Hang zur Wissenschaftlichkeit in Bedrängnis gebracht.

Man mag nun einwenden, dass Abstraktion der Kern der Informatik sei und man deshalb im Unterricht das Schwergewicht auf das abstrakte Denken legen müsse. Es lohnt sich, dieser Frage genauer auf den Grund zu gehen. Peter J. Denning, ehemaliger Präsident der ACM, setzt sich immer wieder mit der Frage auseinander, was Computer Science ist. Das Studium einiger seiner Artikel ist für jeden Informatiklehrer ein Muss. In [De05a] stellt er die Frage, ob die Informatik überhaupt eine eigenständige Wissenschaft sei. Er weist darauf hin, dass zu dieser Frage auch unter den Informatikern kein Konsens besteht und folgert: "Computer science meets every criterion for being a science, but it has a self-inflicted problem." Könnte diese Aussage auch auf den Informatikunterricht in der heutigen Form zutreffen? In [De05b] führt Denning unter dem Titel *Look Beyond Abstraction to Define Computing* zum Thema Abstraktion aus:

> [..] computer science's agenda is, and always has been, information processes. Some are natural and some artificial. We study and imitate the natural; we design and test the artificial. Many of the artificial are inspired by their natural counterparts. Many programs and computer systems are means to study, simulate, and replace natural information processes. For example, cognitive scientists hypothesize that intelligence is the result of information processes in our brains. Experts in human-computer interaction study the interactions among artificial information processes in computers and natural processes in humans. Bioinformaticians study the transcription of DNA as an information process that can be managed and repaired. Computational scientists study natural phenomena with computer simulations. [..]
>
> Much programming practice consists of designing abstractions - objects and functions - to solve problems, then letting computers execute the abstractions to produce real action. What is distinctive about abstraction? Engineers, scientists, organizational leaders, and game makers do it all the time. Abstraction is an important but not the defining principle of computing.

Vielleicht müssten wir im Informatikunterricht Denning's Ausführungen mehr beherzigen. Genauso wie Mathematik nicht mit Mengenlehre gleichzusetzen ist, besteht Informatik nicht nur aus Abstraktion und Modellierung. Um sich als eigenständiges Unterrichtsfach nachhaltig etablieren zu können, muss für die Schülerinnen und Schüler Sinn und Zweck der Informatik ersichtlich sein. Lehrpersonen anderer Fächer und Bildungspolitikern muss der grundlegende Nutzen der Informatik überzeugend kommuniziert werden können. Um das Backofen-Beispiel aufzunehmen: Informatikunterricht muss auch einen Beitrag dazu leisten, dass wir unsere Pizza backen können. In den Lehrplänen müssen vermehrt Themen mit einem starken Alltagsbezug - etwa der Entwurf von Mensch-Maschinen-Schnittstellen - Einzug halten. Warum nicht im Informatikunterricht Norman's Klassiker *The Design of Everyday Things* [No88] lesen und begleitend unterschiedliche Mensch-Maschinen-Schnittstellen analysieren?

3 Was muss in der Informatik unterrichtet werden?

Die Auswahl des Unterrichtsinhaltes ist in der Informatik schwieriger als in anderen Fächern. Informatikunterricht deckt ein sehr breites Spektrum an Themen und Zielsetzungen ab. Betrachten wir als Beispiel Datenbanken. In der Hochschule wird man den Schwerpunkt auf die Modellierung und das Design von Datenbanken legen. In einem Ausbildungsgang für IT-Supporter wird man die Anwendersicht sowie das Anpassen von Datenbanken in den Vordergrund stellen. Für die Mehrzahl der Anwender stehen Datenbankabfragen im Vordergrund. Die verschiedenen Zielgruppen bedingen verschiedene Themenschwerpunkte und andere didaktisch-methodische Ansätze.

Als geeignetes Werkzeug zur Auswahl der Lerninhalte hat sich die Idee der fundamentalen Ideen von Bruner [Br60] bewährt, welche von Schwill [Sc94] auf den Informatikunterricht adaptiert wurden. Berücksichtigt man die Kriterien zur Auswahl der fundamentalen Ideen für den Unterricht, läuft man nicht Gefahr, sich in kurzlebigen produktspezifischen Details zu verlieren. Allerdings reicht es nicht aus, in einem Themengebiet einfach möglichst alle fundamentalen Ideen zu identifizieren. In Abhängigkeit der Zielgruppe können grundlegende Konzepte eines Themas für den Unterricht sehr relevant, aber auch vollkommen belanglos sein. Im Datenbank-Unterricht mit Zielpublikum „Anwender“ spielen die Normalformen bei relationalen Datenbanken keine Rolle, für künftige Datenbankentwickler hingegen schon. Es ist Aufgabe der Lehrperson, aus den fundamentalen Ideen diejenige Teilmenge zu wählen, welche für die Zielgruppe von Bedeutung ist. Ein Blick in die vergangenen zehn Jahre des gymnasialen Informatikunterrichtes lässt vermuten, dass dieser bewussten Auswahl in der Vergangenheit zu wenig Beachtung geschenkt wurde.

Wir plädieren dafür, dass auch im Informatikunterricht ausgehend von den *fundamentalen Ideen* eines Themengebietes vermehrt eine zielgruppengerechte Auswahl getroffen wird. Auch die Rahmenbedingungen wie Schultypus, zukünftiges Tätigkeitsgebiet der Lernenden, zur Verfügung stehende Unterrichtszeit und Infrastruktur sollten in die Auswahl der Lernziele einfließen. Abbildung 2 zeigt das von uns in der Lehrerausbildung in den letzten Jahren eingesetzte Vorgehensmodell.

Die *Leitideen* legen einen Bezugsrahmen fest. Sie begründen einerseits die Ausgangslage, warum etwas überhaupt gelehrt werden soll. Andererseits halten sie als Konsequenz daraus fest, *was* gelernt werden soll und schließen so gewisse Themen aus. Die Leitideen zeigen den Schülern die Relevanz eines Themas auf und ordnen das Thema in einen größeren Kontext ein.

Die Leitideen berücksichtigen das Zielpublikum und führen damit zur Auswahl der fundamentalen Ideen für einen bestimmten Unterrichtskontext. Die *Dispositionsziele* stehen zwischen der eher allgemein formulierten Leitidee und den fundamentalen Ideen einerseits und den operationalisierten Lernzielen andererseits. Sie dienen als Brücke zwischen diesen unterschiedlichen Ebenen und beschreiben angestrebtes Verhalten, beispielsweise die Bereitschaft, beim Programmieren den Programmcode aussagekräftig zu dokumentieren.

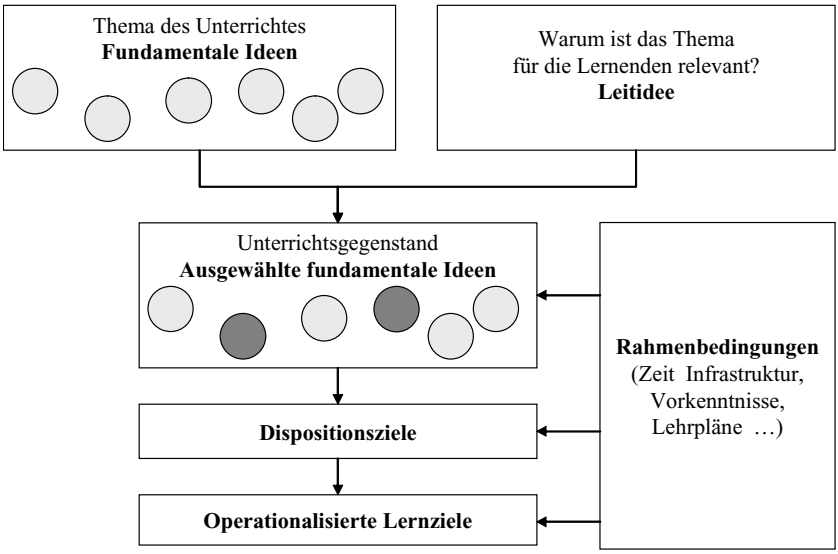

Abbildung 2: Vorgehensmodell Auswahl Lernziele

Operationalisierte Lernziele halten fest, was die Lernenden nach dem Unterricht können und wie dieses Können überprüft werden kann. Lernziele haben positive Effekte sowohl für die Lernenden als auch für die Lehrpersonen. In den Einheitlichen Prüfungsanforderungen zur Abiturprüfung (EPA) der Kultusministerkonferenz vom 12.03.2004 sind die allgemeinen Lernziele für den gymnasialen Informatikunterricht ausführlich beschrieben. Zum Thema Strukturieren und Modellieren findet sich unter anderem:

> [..] Informatische Methoden wie das Strukturieren, das systematische Zerlegen komplexer Systeme in überschaubare Teile, das Formalisieren und Interpretieren fördern und fordern die Abstraktionsfähigkeit und das Erfassen logischer Zusammenhänge. Bei der Modellbildung, die bei der Konstruktion und Analyse von Informatiksystemen eine entscheidende Rolle spielt, üben die Schüler in besonderem Maße eine Situation von verschiedenen Standpunkten aus zu beurteilen; die systematische Überprüfung und kritische Beurteilung der Ergebnisse sowie des gewählten Modells fördern die Fähigkeit zu konstruktiver Kritik. Gleichzeitig werden die für den erfolgreichen Einsatz des Computers nötige Sorgfalt, Genauigkeit und Ausdauer gefördert. [..]

Gegen diese Zielsetzungen ist nichts einzuwenden. Genauso wie sich die Mathematik die Förderung des logischen Denkens auf die Fahnen schreibt, kann und muss die Informatik auch ihren Beitrag zur Bildung leisten. Im Unterschied zur Mathematik ist es aber der Informatik noch nicht gelungen, diesen Beitrag zur Allgemeinbildung in einer breiteren Öffentlichkeit zu verankern. Ganz unschuldig sind die Informatiker an diesem Umstand nicht. Informatiker wirken auf den, von den Unzulänglichkeiten der Informatik geplagten Nicht-Informatiker oft ein wenig überheblich.

Denken wir zum Beispiel an den zweifelhaften Ruf, den die ECDL-Zertifikate in der Computer Science Community genießen. Wie schnell mokiert man sich über Aufgaben der folgenden Art:

> Wechseln Sie in die Ansicht Folienmaster in der Datei infos.ppt. Kopieren Sie die Grafik aus der Datei dresden.ppt und positionieren Sie sie so, dass die obere rechte Ecke der Grafik mit der oberen rechten Ecke des Titelbereichs deckungsgleich ist. Formatieren Sie den Hintergrund der Masterfolien mit folgenden Farbeinstellungen: Farbton 106, Sättigung 255, Intensität 102. Weisen Sie die Änderung allen Folien ab Folie 5 zu.

Nicht selten sind es gerade die Vertreter der Hochschulinformatik, welche selber mit den praktischen Werkzeugen der Informatik große Mühe bekunden und denen es nicht leicht fällt, die abstrakten Konzepte allgemein verständlich und mit Alltagsbezug zu vermitteln. Der Informatikunterricht kann sich solchen praktischen Problemen gegenüber aber nicht verschließen. Will die Informatik den ihr zustehenden Platz als allgemein bildendes Grundlagenfach einnehmen, muss sie die Leitideen des Unterrichts vermehrt an den Interessen und Bedürfnissen der Lernenden ausrichten. Für ein Gymnasium musischer Ausrichtung dürfte eine Einführung in die objektorientierte Programmierung mit Java wenig Sinn machen. Hingegen würden sich mit Flash interessante kleine Projekte aus dem Erfahrungsumfeld der Schülerinnen umsetzen lassen. Auch in einem auf Flash basierenden Informatikunterricht lässt sich eine Vielzahl wichtiger Informatikkonzepte thematisieren. Angefangen von Grafikformaten, über programmgesteuerte Ablaufplanung bis hin zu objektorientierter Modellierung. Nicht das Werkzeug „Flash“ darf im Vordergrund stehen, sondern die diesem Produkt zugrunde liegenden Konzepte.

4 Was ist heute in der Informatik gerade aktuell?

Neben Überlegungen zur Auswahl von Inhalten und Lernzielen für den Unterricht lohnt es sich auch, ab und zu einen Blick auf das Geschehen in der Informatikwelt insgesamt zu werfen. In den monatlich publizierten „Top 10 Downloads from ACM’s Digital Library“ spielen Modellierung, UML, Java und C in allen Varianten zurzeit eine wichtige Rolle. Daraus den Schluss zu ziehen, man müsse im Informatikunterricht diese Themen bevorzugt behandeln, wäre aber kurzsichtig. Die Informatik ist bekanntlich großen Veränderungen unterworfen und neigt im Vergleich zu anderen Wissenschaften stärker dazu, einem aktuellen Trend zu erliegen. Im Bereich der Softwareentwicklung spricht man heute objektorientiert und denkt in UML. Auch die Schulinformatik konnte sich diesem Trend nicht entziehen und legt großes Gewicht auf den Prozess und die Werkzeuge zur Modellierung. Eine Beispielaufgabe aus dem Modul 226 „Teilapplikationen objektorientiert analysieren und implementieren“ in der Schweizer Informatik-Berufsausbildung lautet:

> Unsere Kundin, die Leiterin einer Jugendherberge, steht vor einem immer wiederkehrenden Problem. Wenn eine Reisegruppe ankommt, dauert es immer sehr lange, bis die Mitglieder auf die Zimmer verteilt sind. Die Jugendherberge besteht aus: 3 Zimmern mit 4 Betten, 2 Zimmern mit 5 Betten, 2 Zimmern mit 3 Betten und 1 Zimmer mit 2 Betten.
>
> Die Leiterin ist der Meinung, dass dieses Problem mit einer IT-Teilapplikation gelöst werden kann. Sie gibt Ihnen den Auftrag, eine Teilapplikation objektorientiert zu erstellen und zu dokumentieren. Dazu gehören das Use Case Diagramm, das Klassendiagramm und das Sequenzdiagramm, das den Ablauf des Zuordnens der Mitglieder einer

Reisegruppe zu den einzelnen Zimmern darstellt. Danach soll die Teilapplikation benutzerfreundlich realisiert sein.
Realisieren Sie gemäß den Vorgaben die Teilapplikation. Testen und dokumentieren Sie die Anwendung.

1. Notation und Symbolik von UML ist für die verlangten Diagramme angewendet.

2. Die Aufgabenstellung ist objektorientiert programmiert.

3. Die Diagramme sind überprüft und der Code der Teilapplikation ist methodisch getestet.

4. Analyse und Code sind unter Beachtung von Wartbarkeit und Nachvollziehbarkeit dokumentiert.

Das Beispiel zeigt sehr deutlich eine Schwachstelle des Informatikunterrichtes auf. Um aktuell zu sein und um den Anforderungen der Firmen als Abnehmer der Absolventen einer Berufslehre Rechnung zu tragen, versucht man den Spagat zwischen der komplexen Realität im Software-Entwicklungsalltag und einer vertretbaren schulischen Umsetzung. Das Resultat ist eine Verteilungsaufgabe von n Gästen auf x Zimmer mit y Betten, wobei z Betten belegt sind und Männlein und Weiblein wohl getrennt werden müssten. Ein rein kombinatorisches Problem, einfach prozedural zu lösen, noch einfacher mit einer Excel-Tabelle und wenn schon die Jugendherberge modelliert werden muss, ein klassischer Fall für eine relationale Datenbank. Mit solchen Themen und Aufgabenstellungen können im besten Fall auf dem Papier hochgesteckte Ziele dokumentiert werden. Einen Dienst erweist man aber weder den Lehrenden noch den Lernenden.

Vielleicht wäre ein bisschen mehr Bescheidenheit der Sache eines Faches Informatik förderlicher? Wir müssen ja in der Schule keine großen Softwaresysteme in Angriff nehmen. Wer in seiner Freizeit gern Modellflugzeuge baut, hat in der Regel auch klein angefangen. An einfachen Modellen lernt man die grundlegenden Methoden und Techniken kennen: Arbeiten mit Bespannpapier, Schleifklotz und Spannlack, Kurvensteuerungen einstellen oder Auswiegen. Natürlich hat man dabei immer ein Fernziel vor Augen: große, schöne und elegante Modellflieger.

Es ist nicht leicht, sich gegen einen Trend zu stellen. Wer heute nicht objektorientiert spricht und UML denkt, läuft schnell Gefahr, zu den Ewiggestrigen gezählt zu werden. Natürlich sind Objekte in vielen Situationen extrem nützlich, aber so wie ein Hammer nicht immer das richtige Werkzeug ist, so ist der objektorientierte Ansatz nicht immer der Beste. Und nicht vergessen werden darf, dass kein Objekt von sich aus irgendwas macht. Dazu braucht es Methoden, programmiert in prozeduraler Art und Weise. Früher wurden die Diskussionen um den Informatikunterricht von Glaubenskriegen zur Wahl der richtigen Programmiersprache beherrscht. Heute finden ähnliche Auseinandersetzungen auf dem gehobeneren Niveau der Programmierparadigmen statt.

Für den Informatikunterricht wäre es wohl nur von Gutem, wenn man diese Diskussionen außen vor lassen und im Unterricht die verschiedenen Sicht- und Herangehensweisen beim Erstellen von Software aufzeigen würde. Ein kleiner Trost mag sein, dass das UML-Fieber nicht nur die Schulen erfasst hat. In einem Aufsehen erregenden Artikel

Death by UML Fever beschreibt Alex E. Bell, Software-Entwickler bei Boeing, 4 Metafieber und 18 Fieber [Be04]. Bell modelliert die verschiedenen UML Fieberarten und ihre Abhängigkeiten gleich in Form von UML-Diagrammen. Unter anderem diagnostiziert er folgende Fieberarten:

> Fundamental Metafever:
> At the root of most UML fevers is a lack of practical experience in those individuals responsible for […] software-development efforts.
>
> Curator Fever <<instanceof>> Delusional Metafever:
> Victims believe that production of UML diagrams, as opposed to the engineering content behind them, is the single most important activity in the software-development life cycle.

Wir gehen hier nicht näher auf diese Krankheit in der Informatikwelt ein, merken nur noch an, dass Grady Booch – einer der Begründer von UML – in einem Kommentar zu Bell's Artikel schreibt:

> The entertaining tenor of „Death by UML" should not be inferred to suggest that UML fever is an imaginery ailment. It is genuinely real, it is thriving, and its presence is causing cost and schedule trauma on many software programs right now.

Auch größere Studien zum Programmierunterricht stützen die weit verbreitete Ansicht nicht, objektorientierte Programmierung erleichtere Anfängern den Einstieg ins Programmieren (vgl. zum Beispiel [RRR03]).

> The papers reviewed do not support this position. They show that identifying objects is not an easy process and that objects identified in the problem domain are not necessarily useful in the program domain, that the mapping between domains is not straightforward, and that novices need to construct a model of the procedural aspects of a solution in order to properly design objects/classes.

Diese Erkenntnisse sind bei genauerem Hinsehen nicht überraschend. Die Hauptaufgabe bei OO-Analyse und OO-Design besteht ja in der Modellierung und Beschreibung einer realen Situation durch kommunizierende Objekte. Die verwendeten Techniken wie etwa Klassen, Assoziationen, Interfaces, Vererbung, Design Pattern oder UML beschreiben letztlich einen komplexen Graphen mit Objekten als Knoten. Im Artikel *Scale-Free Geometry in OO Programs* [Po05] wird sehr schön beschrieben, dass die Struktur eines solchen „Objektgraphen" nicht den gängigen Vorstellungen entspricht. Große OO-Programme sind nicht aus vielen kleinen, überschaubaren Objekten mit ungefähr der gleichen Größe und der gleichen Komplexität aufgebaut. OO-Programme entsprechen in ihrer Struktur vielmehr dem World Wide Web. Es gibt viele Objekte mit nur lokaler Bedeutung, aber auch einige wenige Objekte, welche mit praktisch allen anderen Objekten in Beziehung stehen.

Das Programmieren ist und bleibt eine schwierige Sache, unabhängig vom zugrunde liegenden Paradigma. Gerade in der kurzlebigen Informatik ist es gefährlich, sich einseitig an aktuellen Trends zu orientieren. Die Zeit der Lehrpersonen ist kostbar und jedes Einarbeiten in ein neues Gebiet, eine neue Methode oder ein neues Werkzeug muss wohlüberlegt werden. Ein „konservativer", aber guter Unterricht, beispielsweise Programmieren mit Pascal, ist unter Umständen effektiver als ein „moderner" Unterricht,

der Schüler und Lehrer gleichermaßen überfordert. Der Autor erlaubt sich hier auch einen Seitenhieb an die Gemeinschaft der Informatik-Didaktiker, der er selbst angehört: Wenn sich die Informatik im Fächerkanon etablieren soll, müssen die Verantwortlichen in der Informatiklehrer-Ausbildung vermehrt am gleichen Strick ziehen und dürfen sich nicht in Inhaltsdiskussionen weitab vom realen Schulalltag verlieren.

5 Werkzeuge und Objekte – Konzepte und Produkte/Fertigkeiten

Nachdem in den vorangehenden Kapiteln das „Was“ beleuchtet wurde, soll zum Schluss ein Blick auf das „Wie“ geworfen werden. Beim Erstellen dieses Dokumentes war der Autor mit der scheinbar läppischen Aufgabe konfrontiert, aus der als PDF-Datei vorliegenden Gebrauchsanweisung eines Backofens ein Bild in ausreichender Qualität zu extrahieren. Mit allen notwendigen Programmen ausgestattet, hat dieses einfache Unterfangen eine gute Stunde Arbeit verursacht. Der Autor kennt zwar die Konzepte hinter den verschiedenen Grafikformaten und auch Algorithmen in der Bildbearbeitung. Sobald es aber um die konkrete Umsetzung dieser Konzepte geht, entsteht ein Transferproblem.

Das Vermitteln vieler Informatikinhalte findet auf zwei Ebenen statt. Im Beispiel der Bildbearbeitung gibt es zum einen die Objekt-Ebene, die einzelnen Bilder und die damit zusammenhängenden Konzepte: Bitmap- vs. Vektorbilder, verlustbehaftete vs. verlustfreie Komprimierung von Bildern, Farbräume und vieles mehr. Zum anderen gibt es die Werkzeug-Ebene: Bilder werden mit Hilfe einer Software zur Bildbearbeitung oder mit Grafikprogrammen erstellt und bearbeitet. Auch auf dieser Ebene gibt es Konzepte, beispielsweise die Verwendung von Ebenen, Masken oder Filtern. Sowohl auf der Werkzeug-Ebene als auch auf der Objekt-Ebene müssen die Konzepte und deren konkrete Umsetzung unterschieden werden. So sind zum Beispiel auf der Objekt-Ebene GIF und JPEG beides Bitmap-Formate, aber sie unterscheiden sich dennoch beträchtlich in ihren Eigenschaften sowohl die Unterscheidung der Objekt- und Werkzeug-Ebene als auch der Konzepte und ihrer Umsetzung auf diesen beiden Ebenen ist für die Schüler nicht immer einfach. Das eigentliche Lernziel ist häufig auf den primären Unterrichtsgegenstand ausgerichtet, im Beispiel der Bildverarbeitung etwa auf die verschiedenen Arten von Bildern und die konkreten Bildformate, also die Objektebene. Die Werkzeug-Ebene stellt einen sekundären Lerngegenstand dar. Es besteht die Gefahr, dass der Zusammenhang zwischen primärem und sekundärem Lerngegenstand verloren geht: Wo finden sich in Photoshop die Farbtabellen, die Auflösung oder die Ebenen? Wie kombiniert Illustrator Vektorgrafiken mit Bitmaps? Lehrpersonen fühlen sich zudem auf der sich schnell ändernden Werkzeug-Ebene oft unsicher und neigen dazu, die Werkzeugaspekte im Informatikunterricht unbewusst zu verdrängen oder aber nur den Werkzeugaspekt zu behandeln.

Man könnte vermuten, dass die Aufteilung in Konzepte und deren Umsetzung in Produkte sowie die Aufteilung in Objekte und Werkzeuge informatikspezifisch sei. Ein Blick über den Gartenzaun zeigt, dass andere Berufe mit genau der gleichen Problematik konfrontiert sind. Bleiben wir in der Nähe der Bildbearbeitung und schauen wir die Dimensionen der Ausbildung in Malerberufen an. Im Schweizer Reglement über die Ausbildung von Malern steht als Leitidee:

> Der Maler befasst sich mit dem Auftragen von Anstrich-, Beschichtungs- und Strukturmaterialien sowie mit dem Aufziehen von Tapeten, Belägen und Geweben. Er verschönert damit Bauten, Einrichtungen und Gegenstände und schützt sie gegen Witterungs- und andere Einflüsse.

Anschließend werden Konzepte zu den in Malerberufen wichtigen Objekten (Materialien) und Werkzeugen sowie das für die praktische Arbeit notwendige Produktwissen und die benötigten Fertigkeiten aufgelistet. In unten stehender Tabelle sind exemplarisch einige dieser Konzepte und Produkte/Fertigkeiten auf der Objekt- bzw. Werkzeug-Ebene aufgeführt.

	Konzepte	**Produkte/Fertigkeiten**
Werkzeuge	Verschiedene Arbeitsvorgänge und die dabei verwendeten Materialien und Werkzeuge bei Vorarbeiten auf Holz und Holzwerkstoffen, mineralischen Untergründen, Metallen, Kunststoffen, alten Anstrichen und Beschichtungen, Geweben, Vliesen etc. kennen. [..]	Untergrundvorbereitung wie Schleifen, Entrosten, Isolieren, Neutralisieren, Aufhellen usw. Lasurarbeiten und Imprägnierungen auf mineralische Untergründe und Holz sowie Beiz- und Lackierarbeiten. [..]
Objekte	Grundlagen der Farbenlehre, Mischen von Farben, Einfluss des Glanzgrades und der Untergrundstruktur auf den Farbton. [..]	Gängige Abbeiz-, Ablauge- und Neutralisationsmittel, Aufheller und Reinigungsmittel, Pigmente, Bindemittel, Lösungs- und Verdünnungsmittel, Additiv, Farben und Lacke kennen. [..]

Auch im Informatikunterricht muss die Aufteilung nach den Dimensionen, Objekte und Werkzeuge bzw. Konzepte und Produkte/Fertigkeiten transparent gemacht werden. Im Rahmen der Bildbearbeitung dürfen nicht nur einseitig die Konzepte oder die Produkte thematisiert werden. Es ist wichtig, dass auch ein Transfer zwischen dem Konzeptwissen und der Umsetzung und den Anwendungen im Alltag stattfindet.

Nach einer Übergewichtung der anwendungsspezifischen Informatikaspekte in Zeiten der „Integrierten Informatik“ hat das Pendel in den letzten Jahren im Informatikunterricht vielleicht zu stark in die andere Richtung ausgeschlagen. Eine zu einseitige Gewichtung der informatischen Grundprinzipien macht es für Nicht-Informatiker schwierig, den Stellenwert einer fundierten informatischen Bildung richtig einzuschätzen. Hier ist der Informatikunterricht in Zukunft gefordert. Es muss aufgezeigt werden, dass eine informatische Allgemeinbildung den effizienten Umgang mit den heute gängigen Werkzeugen massiv erleichtert. Die Behandlung digitaler Bilder im Unterricht darf nicht auf einer theoretischen Ebene stehen bleiben. Das erworbene Wissen muss in Handlungskompetenz umgesetzt werden, Schülerinnen müssen mit Digitalkameras praktisch arbeiten oder einfache Videosequenzen schneiden. Nur so schafft der Informatikunterricht einen für andere Schulfächer und Politiker ersichtlichen Mehrwert.

6 Informatikunterricht: Es kommt auch auf das „Wie“ an!

Im Vordergrund der Diskussionen in der Informatik-Didaktik standen in den letzten Jahren curriculare Fragen und Überlegungen zur Begründung eines Schulfaches Informatik. Das „Wie“, etwa für den Informatikunterricht geeignete Unterrichtsmethoden und -techniken, wurde wenig thematisiert. Ein Blick auf die Zielsetzungen der INFOS 05 zeigt hier erfreulicherweise eine Trendwende. Eine der Hauptaufgaben der Informatik-Didaktik ist es, konkrete und praktisch umsetzbare Hinweise für die Gestaltung des Unterrichts zu geben.

Informatik ist geprägt durch Abstraktion. Die Methode der Abstraktion ist oft nützlich, sie macht aber einem Neuling das Leben schwer. Kinder lernen anhand konkreter Gegenstände und müssen sich Ereignisse und Abläufe vorstellen können. Je älter wir werden, umso zugänglicher sind wir für Erklärungen, die bildlich oder lediglich in Textform vorliegen. Aber Hand aufs Herz: Oft wären wir froh um ein gutes Beispiel aus unserem Alltag. Vielleicht müssten auch im Informatikunterricht vermehrt die drei klassischen Repräsentationsebenen eingesetzt werden. *Enaktive Ebene*: Erfassen von Sachverhalten durch eigene Handlungen. *Ikonische Ebene*: Erfassen von Sachverhalten durch Bilder. *Symbolische Ebene*: Erfassen von Sachverhalten durch Symbole (Text, Zeichen etc.). Abstraktes im wahrsten Sinne des Wortes begreifbar machen, ist kein neues Thema in der Ausbildung. Unterricht und Denken muss nicht unbedingt formal, abstrakt in Text, Symbolen und Formeln ablaufen. Viele Themen lassen sich ebenso gut in Bildern visualisieren oder gar in aktivem Handeln begreifen.

Auch in der Informatik lassen sich die meisten Sachverhalte anhand einfacher Alltagsanalogien erklären. Meist braucht es nicht mehr als WC-Rollen, Schnur, Papier, Büroklammern und eine zündende Idee. Zur Illustration ein Beispiel aus der Anwenderschulung: Wir alle haben uns schon über Word geärgert, Stunden damit verbracht, ein langes Dokument zu formatieren. Der effiziente Einsatz von Word ist keineswegs einfach. Auch für Word gilt es, grundlegende Konzepte zu vermitteln und den Überblick vor lauter Funktionalitäten nicht zu verlieren.

In der Regel wird im Unterricht zuerst ein Konzept kurz erläutert. Wozu braucht man Tabulatoren? Welche Arten gibt es? Wie setzt man sie ein? Dann folgt eine kurze Demonstration durch die Kursleiterin: Gespannt schauen die Kursteilnehmer auf die Leinwand. Die Kursleiterin blickt auf ihren Laptop, die Kursleiterin und die Kursteilnehmer schauen buchstäblich aneinander vorbei. Mit der Maus zeigt die Kursleiterin, wie man Tabulatoren auswählt und platziert. Nur ist die Benutzeroberfläche viel zu klein, als dass man den Ausführungen auf der Leinwand folgen könnte.

Warum Tabulatoren nicht enaktiv einführen? Abbildung 3 zeigt ein auf den ersten Blick wohl nur ein Lächeln hervorrufendes Holz-Word, ein Modell von Word nachgebaut aus Dachlatten, Vorhangschienen, Wäscheklammern usw. Die Kursleiterin wählt links oben einen Tabulator aus und platziert ihn von Hand in der Formatierungsleiste. Alles ist transparent, groß und sofort einsichtig. Tabulatoren können verschoben werden, die begrenzende Schnur wandert mit. Und selbstverständlich können auch viele andere Konzepte und Bedienungsschritte an diesem Holzmodell gezeigt werden. Das Holzmodell

lässt sich zu einem eigentlichen Framework für enaktive Demonstrationen vieler Aspekte der Office-Palette erweitern.

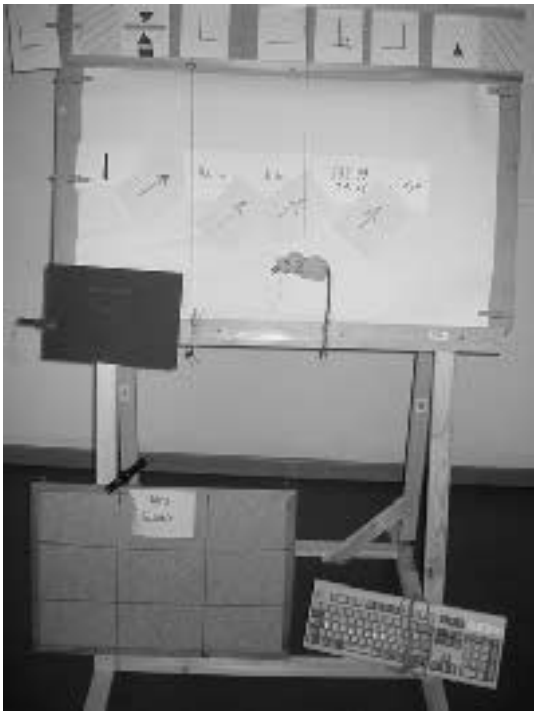

Abbildung 3: Holz-Word (Idee und Umsetzung: Paul Miotti)

Zugegeben: Holz-Word wird im Umfeld der Hochschulinformatik nicht gerade viel zur eigenen Reputation beitragen. Schülerinnen und Informatiklehrer sind aber für konkrete Anregungen dankbar. Die Informatik-Didaktik muss wieder einen Schritt näher zu den Schulen hin machen. Vielleicht müsste die nächste INFOS in einem Schulhaus stattfinden und die Hälfte der Beiträge von Schüler/innen bestritten werden?

7 INFOS 2015: Informatik als Schulfach etabliert

In der Vergangenheit haben zwei Technologien den Unterricht revolutioniert: Die Erfindung der Wandtafel und Gutenbergs Erfindung des Buchdruckes. Gerade diese beiden Technologien könnten in den nächsten Jahren abgelöst werden. Die Wandtafel durch Weiterentwicklungen von Tablet PC's mit drahtloser Möglichkeit der Projektion auf die „Wandtafel", Bücher – im Zusammenhang mit der Schule insbesondere Lehrmittel – durch vernetzte, elektronische Bücher. Daneben ist damit zu rechnen, dass einfache und handliche Sprachübersetzungssysteme die Bedeutung des Fremdsprachenunterrichtes verändern. GIS-Navigationssysteme werden ganze Teile des Geographieunterrichtes überflüssig machen und auch andere Fächer müssen ihre Inhalte an die neuen Gegeben-

heiten einer globalen Informationsgesellschaft anpassen. Auf jeden Fall wird aber der Stellenwert der Informatik immer größer und die Bedienung von Backöfen nicht einfacher. Informatik wird zu einem Schulfach wie Mathematik oder Deutsch. Es liegt zu einem großen Teil an uns Informatiklehrpersonen, wie lange und wie steinig der Weg zur endgültigen Etablierung des Schulfaches Informatik sein wird. Gehen wir ohne EINbildung auf unsere Kolleginnen in anderen Fächern zu und zeigen wir den Nutzen sowohl einer informatischen Bildung wie auch einer informatischen AUSbildung auf. Dann heißt es nicht erst an der INFOS 2025 „Informatik EIN"!

Literaturverzeichnis

[De05a] Denning, P. J.: Is Computer Science Science? Communications of the ACM, 2005; 48(4); S. 27-31.

[De05b] Denning, P. J.: Look Beyond Abstraction to Define Computing. Communications of the ACM; 2005; 48(5); S. 11-12.

[No88] Norman, D. A.: The Design of Everyday Things (Originally published: The psychology of everyday things). New York; Basic Books; 1988.

[Br88] Bruner, J. S.: The Process of Education. Harvard University Press; 1960.

[Sc94] Schwill, A.: Fundamental Ideas of Computer Science. EATCS-Bulletin, 1994, 53; S. 274-295.

[Be04] Bell, A. E.: Death by UML Fever. ACM Queue - Tomorrow's Computing Today; March 2004; 2(1); S. 72-80.

[RRR03] Robins, A., Rountree, J., Rountree, N.: Learning and Teaching Programming: A Review and Discussion. Computer Science Education; 2003; 13(2); S. 137–172.

[Po05] Potanin, A., Noble, J., Frean, M., Biddle, R.: Scale-Free Geometry in OO-Programs. Communications of the ACM, 2005; 48(4); S. 99-103.

Der Autor dankt R. Reichert (Swiss Centre of Innovations in Learning St. Gallen), M. Näf (Departement Informatik ETH Zürich) und M. Lehmann (Universität Bern) für viele anregende Diskussionen im Zusammenhang mit diesem Beitrag.