Informatikunterricht zur Vermittlung allgemeiner Bildungswerte

Roland Mittermeir
Alpen-Adria-Universität Klagenfurt
roland@isys.uni-klu.ac.at

Als Informatikunterricht vor 25 Jahren in Österreich eingeführt wurde standen hehre Bildungsziele, die Vorbereitung junger Menschen auf die „Informationsgeneration", im Vordergrund. Im Lauf der letzten Jahre traten, vor allem im Informatikpflichtunterricht an Allgemeinbildenden Höheren Schulen (AHS), diese Bildungsziele gegen Anwendungskompetenzen zurück.
Diese Arbeit zeigt, dass der Informatikunterricht auch heute noch Bildungswerte vermitteln kann, die an jene Bildungsziele anschließen, welche bei seiner Einrichtung angestrebt wurden. Einige davon werden hier vorgestellt. Allerdings sind diese nur erreichbar, wenn auch die Lehrerschaft in der Lage ist, in die Materie „Informatik" einzudringen und unter dem Begriff Informatik nicht lediglich Elemente der IKT unterrichtet.

1 Visionen der Pioniere

Gute Überblicke der Pioniere der Informatik-Didaktik sind in Arbeiten von Reiter [Reit 05] und von Dagiene et al [DaDS 06] dargestellt. Insbesondere Reiter schildert die Motivation für die Einführung des Informatik Pflichtunterrichts an Österreichs AHS unter Verweis auf Proponenten des Informatikunterrichts in Deutschland wie auch auf die damals gemeinsame Initiative des österreichischen Unterrichtsministeriums, der Sozialpartner und der einschlägigen Industrie.
Einige waren bereits damals skeptisch, ob hier nicht zu viel „Industrie" in den Schulbereich vordringt. Doch studiert man den Aufsatz Reiters und Teile der dort umfangreich zitierten Literatur der frühen Achtzigerjahre, muss man zum Schluss gelangen, dass diese Pioniere den Blick in eine für sie noch unbekannte aber erahnbare Zukunft hatten. Computergestützte Informationsverarbeitung wurde als Zukunftssparte gesehen und kleine Personalcomputer machten es möglich, auch Schülerinnen und Schüler mit diesen Geräten, insbesondere mit ihrer Programmierung, vertraut zu machen. Dass dies anfangs kleine Prögrämmchen waren, meist in Assembler, später vorwiegend in Basic geschrieben, tut den guten Absichten keinen Abbruch. Zwar ist Programmierung nicht mit Informatik – und schon gar nicht mit Informatik im heutigen Verständnis – gleichzusetzen, aber sie vermittelt selbst im einfachsten Assemblerprogramm ein Grundverständnis darüber, was ein Computer ist und worauf die Basis seiner Leistungsfähigkeit und Universalität beruht.
Dieser Fokus auf Programmierung war kein österreichisches Spezifikum. Dagys et al [DaDG 06] zeigen, wie unter wirtschaftlich weit schwierigeren Bedingungen in Litauen volksbildnerische Informatikinitiativen initiiert wurden, indem Programmieraufgaben in Zeitungen veröffentlicht wurden und Interessierte ihre Lösungen an das Institut für Mathematik und Informatik einsenden konnten. Dort wurden die eingesandten Algorithmen geprüft und mit Korrekturen den Einsendern rückgemittelt.
Wie unterschiedlich die einzelnen Initiativen in unterschiedlichen Ländern auch gewesen sein mögen, die Achtzigerjahre standen dennoch unter dem gemeinsamen Topos, dass der Infor-

matikunterricht bildungsrelevante Inhalte konstruktiver Natur vermitteln soll. Informatik wurde als technisches Fach gesehen, das auf eine teils bereits eingetretene, teils noch erwartete technische Revolution, in der die maschinelle Verarbeitung von Information eine zentrale Rolle einnehmen würde, vorbereitet. Die wissenschaftlichen Wurzeln des Faches werden dabei für den Schulunterricht weniger relevant sein. Interessant ist jedoch die Einordnung des Fachs durch Luft [Luft 89, LuKö 94] als *Wissenstechnik*.

2 Die Trendwende

Eröffnete die technologische Entwicklung des Personalcomputers den Bildungsverantwortlichen die Chance, Informatik im oben beschriebenen Sinne in die Schulen zu bringen, führte die spätere Proliferation des PCs als Massenprodukt mit dem damit einhergehenden Preisverfall zum Eindringen des PCs in private Haushalte. Dazu reichte selbstverständlich nicht nur der Preisverfall. Es war auch nötig, dass die Geräte benutzerfreundlicher wurden und dass sie mit Anwendungen ausgestattet waren, die auch für kleinere Betriebe und private Haushalte relevant waren. Das Internet spielte dabei eine wesentliche Rolle.
Eine Konsequenz dieser Entwicklung war, dass die Schule meinte, sie müsse sich diesen neuen Trends anpassen. Dies gilt weniger für den berufsbildenden Bereich (BHS), der ja letztlich berufsspezifische Ziele verfolgt, die wenigstens skizzenhaft exogen gegeben sind. Der allgemeinbildende Bereich hat demgegenüber die weit größere Chance, aber auch Pflicht, für sich zu entscheiden, was denn allgemeinbildende Inhalte sind. Damit trat ein Dilemma auf. Ist *allgemeinbildend* das, was von der Allgemeinheit benötigt wird? Vermutlich ja. Allerdings wird von der Allgemeinheit vieles benötigt, das nicht bildend ist.
Ein wenig stand der Informatikunterricht auch früher schon im innerschulischen Stundenkonflikt auf einer Verteidigungsposition. Ob denn diese Fokussierung auf ein Gerät, den Computer, wirklich zu den von der Schule zu vermittelnden Bildungswerten gehöre? Müssen nicht am Ende des zwanzigsten Jahrhunderts weit mehr Menschen autofahren können als Computer zu programmieren oder auch nur zu bedienen? Dennoch vermittelt die Schule keine Fahrkenntnisse und keinen (klassischen) Führerschein. Auch waren in den Achtzigerjahren Maschinschreibkenntnisse noch weit mehr gefragt als PC-Bedienung. Aber Maschinschreiben war nur in einigen BHS Pflichtfach. In AHS war es bestenfalls angebotenes Wahlfach. – Sollte man dies kritisieren? Wohl nicht. Weder Maschinschreiben noch Autofahren sind Bildungswerte. In diesen Fächern werden Fertigkeiten (skills) vermittelt. Unter Bildung stellt man sich gemeinhin etwas anderes vor.
Dennoch reduziert sich der Informatik Pflichtunterricht an AHS weitgehend auf Grundlagen der Computernutzung wie sie etwa im ECDL-Curriculum umschrieben ist. Gegen die dabei vermittelten Fertigkeiten ist nichts einzuwenden. Sie können in anderen Unterrichtsfächern, insbesondere im Zusammenhang mit Blended-Learning, gut genutzt werden. Allerdings präsentieren sie den Jugendlichen ein Zerrbild des Faches Informatik und können für sich genommen kaum höhere Bildungswerte für eine Informationsgesellschaft vermitteln als Autofahren für eine motorisierte Gesellschaft oder eben Maschinschreiben, Während letzterem kein verpflichtender Bildungswert zugeordnet wurde, gilt dies für PC-basierte Textverarbeitung nicht mehr.
Interessant ist, dass dies nicht bloß die Sicht eines universitären Fachdidaktikers für Informatik ist. In der in diesem Tagungsband beschriebenen österreichweiten Datenerhebung bei Jugendlichen der 9. und 10. Schulstufe dominierten als Inhalte des Informatikunterrichts Begriffe wie Word, Excel und langweilig [Mich 10]. Kaum zu verwundern, dass nach dieser Einführung in das Fach (das zu einem hohen Anteil mit Inhalten überlappt, die in Wahlfacheinheiten der Unterstufe Teilen der Klasse bereits vermittelt wurden) in den letzten Jahren die Motivation, Informatik als Wahlpflichtfach, in dem ja durchaus spannende Inhalte

vermittelt werden die den Begriff *Informatik* rechtfertigen, deutlich sank. Dies gilt vor allem für Mädchen [AKLU 07] und zeigt sich unter anderem im rückläufigen Zugang zu Informatikstudien und in der geringen Frauenquote in diesen. Wer möchte schon Maschinschreiben oder etwas Ähnliches studieren?
Doch dies muss ja nicht so bleiben. Wenn Informatik in den Achtzigerjahren Bildungsinhalte, wenn vielleicht nur enge, anzubieten hatte, wird sie doch wohl auch zu Beginn des 21. Jahrhunderts solche anzubieten haben. Worin sollten diese bestehen? Vermutlich in der Rückbesinnung darauf, dass Informatik das einzige wissenschaftlich fundierte technische Fach ist,[1] das an allgemeinbildenden Schulen als Pflichtfach gelehrt wird. In diesem konstruktiven Element auf einer Ebene, die nicht als Fertigkeit beiseite geschoben werden kann, liegt die Chance, Informatikunterricht aus dem Spannungsfeld des Kampfes um Unterrichtsstunden und nicht zuletzt aus dem Spannungsfeld der Geschlechter, herauszuführen. Es ermöglicht, Jugendlichen sowohl die Faszination des Fachs zu vermitteln, ihnen aber auch zu zeigen, dass allgemeinbildende Inhalte in diesem Fach vermittelt werden (können), die mit einer späteren Berufsperspektive als Informatikerin oder Informatiker nichts zu tun haben, sondern die einfach auf das Leben in einer modernen Gesellschaft, in der Information eine dominante Rolle einnimmt, vorbereitet.

3 Bildungsinhalte, die über Informatik vermittelbar sind

3.1 Aus der Perspektive des Fachs

Sucht man nach Bildungsinhalten der Informatik, fallen einem im deutschen Sprachraum primär die von Schwill postulierten und in Schubert und Schwill in entsprechendem Kontext ausgearbeiteten *fundamentalen Ideen* ein [Sch 93], [ScS 04]. Jüngere Arbeiten von Hromkovič [Hrom 06] und die im Informatik Spektrum veröffentlichte Artikelserie von Wedekind et al. [WeOI 04] sind in diese Betrachtung miteinzubeziehen. Als Pendant in der angloamerikanischen Literatur sind die von Denning publizierten „*great principles*“ [Denn 04] zu nennen.
Den Listungen Schwills und Dennings ist gemeinsam, dass beide Autoren darzustellen versuchen, durch welche Inhalte das Fach Informatik den Erkenntnisraum der Menschheit bereichert hat und weiterhin bereichern wird. Die jeweils gewählten Publikationsorgane zeigen, dass sich die Autoren dabei primär an das Bildungssystem und damit auch an das Schulsystem wenden, doch ist die dabei gewählte Perspektive jene des Fachs auf die Welt. Welche Methoden brachte/bringt Informatik, um Probleme der Welt zu lösen? Hromkovič stellt demgegenüber, so wie diese Arbeit, auf allgemeinbildende Inhalte ab, die durch Informatikunterricht vermittelt werden können. Seine Argumentation unterscheidet sich primär von dieser Arbeit durch das gewählte Abstraktionsniveau.
Die in [Schw 93] bzw. [ScSc 04] genannten Ideen sind auf höherem Abstraktionsniveau angesiedelt als die in [Denn 04] genannten Prinzipien. Insbesondere wenn man die oberste Ebene der fundamentalen Ideen betrachtet, stößt man mit *Algorithmisierung*, *strukturierte Zerlegung* und *Sprache* auf Konzepte, auf die Informatik sicherlich noch keinen Alleineigentumsanspruch stellen darf, die aber einen prägenden Einfluss auf das Fach ausüben. Auf der zweiten Hierarchieebene dieses strukturierten Ideengebäudes findet man Konzepte wie Entwurfsparadigmen, Programmierkonzepte, Ablauf und Evaluation oder Modularisierung, Hierarchisierung, Orthogonalisierung. Wieder könnte man kritisieren, dass dies doch (Programmierkonzepte ausgenommen) mentale Werkzeuge jedes Technikers bzw. mentale Werkzeuge jedes Wissenschaftlers sind. Charakteristisch für Informatik werden diese allerdings in Zu-

[1] Viele sehen im Werkunterricht ebenfalls ein technisches Fach. Dem ist zuzustimmen. Als wissenschaftlich fundiertes Fach kann Werkunterricht jedoch kaum gelten. Es geht um Fertigkeiten, weniger um Bildung per se.

sammenschau mit dem Konzept *Sprache*. Steigt man eine weitere Hierarchieebene tiefer, treten Begriffe auf, für die zwar das Fach Informatik noch immer kein Alleinstellungsmerkmal reklamieren darf, die aber zweifellos so charakteristisch für dieses Fach sind, dass für den Gesamtkomplex die Überschrift *Fundamentale Ideen der Informatik* zweifellos gerechtfertigt ist.
Diese fundamentalen Ideen geben somit dem Bildungssystem Orientierung für den anzubietenden Informatikunterricht. Gilt dies allerdings auch für das Schulsystem, insbesondere für das allgemeinbildende Schulsystem? Hier fällt die Antwort nicht so deutlich aus. Auf den oberen Hierarchieebenen sind hier zweifellos – eben weil sie für sich genommen noch nicht sonderlich fachspezifisch sind – allgemeinbildende Elemente gegeben. Umso tiefer man dringt, umso fachspezifischer werden die „Ideen" jedoch. In Dennings flacher Liste landet man sogar recht unmittelbar bei Begriffen, die typisch fachspezifisch sind. Das klärt die Abgrenzung zu anderen Fächern. Es reduziert allerdings im Verständnis jener, die Informatikunterricht skeptisch gegenüberstehen, die Rechtfertigung, warum denn eigentlich gerade dieses Fach in der Schule vertreten sein muss und dem eigenen Fach, dass doch viel **allgemein**bildender ist, Stunden entzieht. Weben die als fundamental herausgestellten Ideen nicht allesamt das Konzept der Programmierung ein? Wer von unseren Schülern und Schülerinnen wird schon den Beruf des Programmierers / der Programmiererin ergreifen? Außerdem: Programmieren ist schwierig. Warum sollen wir unsere Schüler/innen mit etwas (vermutlich einer reinen Fertigkeit) belasten, die so schwierig ist und die nur ganz wenige von ihnen aufgrund der inzwischen guten Benutzerführung von Computern benötigen werden?
Machen wir also lieber im Informatikunterricht etwas, das allen nützt und sowohl für Schüler/innen wie Lehrer/innen leichter erlernbar und damit – so meint man wohl – auch leichter unterrichtbar ist.

3.2 Brauchbares für die Allgemeinheit: IKT-Unterricht

Wie bereits in Kapitel 2 erwähnt, führten technologische Entwicklungen sowie didaktische wie schulorganisatorische Kritik zu einem Wechsel der Inhalte des (weiterhin gleich benannten) Informatikunterrichts. Dieser Wandel war m.E. radikal genug, dass die gemeinsame Bezeichnung eigentlich nicht mehr gerechtfertigt, ja für die Weiterentwicklung des Fachs Informatik in jenen Gebieten, in denen dieser Wechsel radikal vollzogen wurde, zu einem falschen Bild von Informatik führt und daher schädlich ist.
Die am Ende von Abschnitt 3.1 gestellten Fragen dürfen nicht ignoriert werden. Zweifellos erscheint es in einer Zeit, in der die Großelterngeneration im Umgang mit modernen informationstechnischen Medien noch wenig vertraut ist und auch die Elterngeneration noch nicht wirklich *computer literate* im vollen Sinn des Wortes ist, nötig, Jugendliche an das Potential moderner Informations- und Kommunikationstechnologie (IKT) heranzuführen. Doch wie soll dies erfolgen? Welcher Ziele und daher welcher Inhalte bedarf es, IKT-Unterricht nicht zu einem Unterricht in Fertigkeiten (im Extremfall *teaching to the test* für ein schulexternes Zertifikat) verkommen zu lassen?
Der aus [ScSc 04] übernommenen Passage „... dass der Umgang mit und die gezielte Nutzung von Computern zu einer zentralen Kulturtechnik wie Lesen, Schreiben und Rechnen wird" möchte ich als Rechtfertigung für IKT-Unterricht widersprechen. Eine derartige Formulierung mag im Anglo-Amerikanischen aufgrund der wissenschaftlichen Fachbezeichnung *Computer Science* noch durchgehen. Im deutschen Sprachraum, mit der Bezeichnung *Informatik* für das wissenschaftliche Fach, stellt diese Begründung den Computer viel zu sehr in den Vordergrund. Es geht doch eigentlich nicht um den Computer, sondern um konstruierte (also technische) Systeme, die es erlauben, Daten (im weitesten Sinn) so zu interpretieren, dass Aktionen bewirkt werden.

IKT-Unterricht müsste also, um allgemein**bildend** zu werden, sowohl das konstruktive Element als auch das Informationselement sehr stark betonen. Dies ist möglich, wie etwa Arbeiten von Voß [Voß05], Antonitsch [Anto 06] oder Stadtmann [Stad 10] gezeigt haben. Stadtmann hat sogar versucht, den Unterricht von Tabellenkalkulationssoftware so zu positionieren, dass damit ein relativ breites Spektrum der fundamentalen Prinzipien abgedeckt wird.
Man könnte IKT-Unterricht also offenbar so gestalten, dass er zu Recht als Informatikunterricht bezeichnet werden dürfte. Allerdings geschieht dies in der Praxis eher selten. Diese Art von Unterricht würde nicht zuletzt von der Lehrerschaft voraussetzen, dass diese tief genug in der Informatik als Fach verwurzelt ist, um die Querbeziehungen zwischen diesem technisch konstruktiven Fach und der reinen Nutzung von Produkten, die dieses Fach hervorgebracht hat, didaktisch angemessen darzustellen.
Einen weiteren Kritikpunkt möchte ich noch aufgreifen: Ist IKT-Unterricht (in der gegenwärtigen Form) nicht eine rein transiente Erscheinung? Computernutzung lernt man in der Schule, Handy-Nutzung und Computer-Spiele lernt man von Peers. Aber Mobiltelefone, selbst solche in der Hand von Kindern und Jugendlichen, sind heute nichts anderes als kleinste Personalcomputer, mit denen man auch telefonieren kann. Innerhalb einer Generationsspanne – so könnte man meinen – würde IKT-Unterricht ohnehin obsolet. Dem ist zu widersprechen. Tort et al [ToBB 08] zeigten aufgrund einer empirischen Erhebung unter französischen Jugendlichen, dass diese etwa im Bereich der Tabellenkalkulation nur sehr basale Kenntnisse als Selbstverständlichkeiten mitbrachten, der größere Rest jedoch nur durch formalen Unterricht vermittelbar war.

3.3 Informatikunterricht aus Perspektive der Allgemeinbildung

In diesem Abschnitt möchte ich mich von IKT entfernen und wieder dem Fach zuwenden, dessen Namen das Unterrichtsfach *Informatik* trägt. Können aus diesem Fach – unbesehen der Visionen, die jene Pioniere, die es vor 25 Jahren in das Schulsystem integrierten – **allgemeinbildende** Inhalte abgeleitet werden? Mit „allgemeinbildend“ meine ich dabei Inhalte, oder besser Ideen, die für die Allgemeinheit der Kinder und Jugendlichen im späteren Leben relevant sind (also nicht nur für jene, die sich später dem Fach Informatik zuwenden wollen) und die bildend im klassischen Sinn des Wortes sind, also über die reine Vermittlung von Fertigkeiten deutlich hinausgehen und Bildungsinhalte bieten, die in neuen, von der Erwerbssituation abweichenden Situationen zur Anwendung gelangen, also übertragen werden können.
Der Weg in diese Richtung scheint mir durch Lufts Charakterisierung des Fachs als *Wissenstechnik* vorgezeichnet zu sein [Luft 89, LuKö 94]. Wissen ist das, was Schule im weitesten Sinn vermitteln möchte. Wie dies am besten erfolgt, wird durch unterschiedliche Theorien beschrieben bzw. propagiert. Konstruktivismus nimmt dabei eine bedeutende Rolle ein. Gerade im Begriff Wissenstechnik schwingt im Wort *Technik* das konstruktive Element mit. Ja, Informatik ist eigentlich das einzige Schulfach, das sich eindeutig als technisch-konstruktives wissenschaftliches Fach charakterisieren lässt.
Mit der Fokussierung auf Wissenstechnik gelingt einerseits der Anschluss an die Bildungsziele der Informatikpioniere, die das 21. Jahrhundert als das Jahrhundert einer Informationsgesellschaft sahen. (Die Übertragung auf Wissensgesellschaft mag streng genommen etwas zu kühn sein, doch sei sie hier gestattet.) Andererseits stellt sie das Wesen des Technisch-Konstruktiven in Bereichen dar, die relativ fern von klassischer physischer Technik sind.
Informatik kann somit auf höchster Abstraktionsebene als Fach angesehen werden, das zur *Kreativitätsschulung* auch für jene Personen geeignet ist, deren motorische Fähigkeiten (Werken, Zeichnen, künstlerische Gestaltung) unzureichend sind, die aber in einer modernen Industriegesellschaft ihren Lebensunterhalt durch kreative Lösungen verdienen müssen.

Zum Unterschied von künstlerischen Fächern, die selbstverständlich auch kreativitätsfördernd sind, ist Informatik allerdings ein technisches Fach. Es endet nicht im kreativen Vorschlag, sondern dieser „gilt" erst, wenn er umsetzbar ist, also funktioniert. Dies bedeutet Kreativität in einem durch Randbedingungen beschränkten Raum. Randbedingungen hat jegliches künstlerische Schaffen. Doch die Randbedingungen, die etwa ein kreativ gestaltetes Bild zum Kunstwerk werden lassen, sind von anderer Natur als ein relativ einfacher Funktions- oder Leistungstest eines technischen Produkts. Sie erschließen sich daher kaum der Welt des Regelunterrichts.
Informatik verbindet somit das klassische Bildungsziel der Schule mit modernen Anforderungen unserer Gesellschaft, die alle Angehörigen dieser Gesellschaft betreffen.
In der Folge soll an einigen Beispielen dargestellt werden, welche Konzepte als Nebenprodukte fundierten Informatikunterrichts entstehen sollten, die unmittelbar transferierbar sind und die auch als Beiträge zur Schulung von Kreativität angesehen werden können.

4 Beitrag der Informatik zu allgemeinen Bildungszielen

4.1 Die Macht der Sprache

„Ich ersuche Dich und Du machst es!" – Das ist doch ganz normal und alltäglich, darüber muss man keine Worte verlieren. – Oder doch? „Ich ersuche Dich und obwohl Du es machen wolltest, hast Du etwas anderes gemacht." – Ist uns diese Erfahrung nicht ebenso geläufig. Warum? „Hast Du mich falsch verstanden oder habe ich Dich unpräzise ersucht?" Wir Menschen gehen mit unserer natürlichen Sprache nicht zuletzt aufgrund unserer natürlichen Intelligenz und der Möglichkeit gesprochenes nonverbal zu ergänzen, eigentlich recht sorglos um. Welche Konsequenzen ein zu salopper Umgang mit Sprache hat, erleben wir nicht zuletzt dann deutlich, wenn ein eMail-Disput zu Flaming ausartet und wir auf andere Kommunikationskanäle ausweichen müssen, um die Flammen wieder zu löschen und zu einem normalen Dialog zurückzukehren.

In Informatik geht es allerdings nicht darum, einer Freundin ein eMail zu schreiben. Es geht darum einer lediglich aus Halbleitern und deren Verbindungen aufgebauten Maschine durch rein sprachliche Formulierungen extrem unterschiedliches Verhalten aufzuzwingen. Hier tritt Sprache in einer Stringenz auf, die die sprachlich-manipulativen Reden publikumswirksamer Politiker vor Massen bei weitem überschreitet.[2] Auf einer Seite steht die „tote" Maschine, auf der anderen Seite das „Wort", das sie zu einer ganz konkreten Form des „Lebens" erweckt.

> *Im Unterschied zu anderen technischen Fächern konstruieren wir in der Informatik nicht durch physische Materialbearbeitung, sondern durch sprachliche Formulierung.*

Um dies zu erkennen, muss man sich von der „Point-&-Click-Metapher" entfernen und wirklich schreiben lassen. Ob dieses Schreiben in einer Programmiersprache oder in strukturiertem Deutsch erfolgt, ist nebensächlich. Wichtig ist, dabei zu erkennen, dass unser Gegenüber (z.B. der Computer, es kann aber auch ein Mensch sein) unseren Text nur dann richtig verstehen und ausführen wird, wenn

- wir uns der Sprache unseres Gegenübers bedienen,

[2] Vielleicht nicht in der Wirksamkeit, wenn man an ausgewählte politische Reden denkt, sicherlich jedoch in der Vorhersehbarkeit.

- wir die Randbedingungen unseres Vis-a-Vis berücksichtigen, (Hat er Strom? Hat er die Ressourcen, diese Leistung zu erbringen? Hat er Vertrauen?)
- und wir in der Erteilung unserer Anweisungen antizipieren, in welche Situationen unser Vis-a-Vis während der Ausführung dieser Anweisungen gelangen kann.

Diese Punkte erscheinen einsichtig. Doch zu viele Missverständnisse und Konflikte beruhen darauf, dass Einzelne sie in der Kommunikation mit anderen nicht einhalten. Mit Informatikunterricht scheinen diese Probleme am ersten Blick wenig zu tun zu haben. Oder doch? Sind das nicht genau die Probleme, warum Programmieren Anfängern mitunter schwer fällt. Doch wir dürfen Programmieren nicht als Vorbereitung auf professionelles Software Engineering auffassen. Es ist Vorbereitung auf Antizipationsfähigkeit und Training der Fähigkeit, sich exakt auszudrücken.

4.2 Emotionen sind menschlich – Maschinen sind emotionsfrei

„Computer haben angeblich ‚künstliche Intelligenz'. Da haben sie sicherlich auch Emotionen?" „Ja gewiss, man sieht es doch, wie mich dieses Textverarbeitungsprogramm ‚beißt' und auf böswilligste Art missversteht. Erst wenn ich dann recht schimpfe, dann geht es irgendwie doch, dann erbarmt sich der Blechtrottel meiner."
Wie oft haben wir Ähnliches nicht nur von SchülerInnen, sondern auch von Erwachsenen schon gehört? Mystifizierungen, die auf Unverständnis fußen. Damit meine ich nicht Unverständnis im Detail, sondern prinzipielles Unverständnis vom Umgang mit Technik. Dies ist für Angehörige einer technikbasierten Industriegesellschaft nicht zeitgemäß. Erinnert uns solches nicht an Zeiten, in denen Frauen mit besonderen Fähigkeiten als Hexen verbrannt wurden, wenn ihre Fähigkeiten einmal nicht so „funktionierten", wie dies von Autoritäten gewünscht wurde?
Hexenverbrennungen und Technikfeindlichkeit so knapp nebeneinander zu stellen erscheint etwas kühn. Ist es aber nicht. Mystifizierungen von Technik führen zu einer unreflektierten Technikfeindlichkeit, die nur all zu oft andere gegenüber berechtigter Technikkritik immunisiert. Denken wir doch nur an aufständisch zerstörerische Bewegungen, in denen Technik nicht bloß als eine Manifestation von Kapital, sondern auch als Fortschritt, an dem die aufständische Klasse nicht teilhaben kann, bekämpft wird. Hier bedarf es der Aufklärung im Großen. Diese kann Informatik nicht leisten. Sie kann aber ein Grundverständnis für Technik leisten, indem der Dualismus von statischer Hardware und dem von dieser ausgeführtem Code verstanden wird.

> *Hardware besteht aus (relativ) anwendungsneutral konzipierten Bauteilen.*
>
> *Software besteht aus anwendungsspezifisch konzipierten sprachlichen Anweisungen. Diese müssen die innerhalb einer Anwendung auftretbaren Situationen antizipieren und behandeln. Software ist jedoch nicht anwendungsfall- oder nutzerspezifisch. Sie wurde lange vor und unabhängig von Anwendungssituation und Nutzer (also mir) geschrieben. Daher kann das technische Gesamtsystem auch keinerlei Emotionen für oder gegen mich, für oder gegen meine Nutzung des Systems haben.*

Dies ist letztlich nichts anderes als ein wenig Aufklärung des ausgehenden zwanzigsten Jahrhunderts. „Brauchen wir dazu Informatikunterricht?" Je nach gesamtschulischem Umfeld wird die Antwort unterschiedlich klar ausfallen. „Der Physikunterricht könnte das doch auch!?" Ja, aber die Demonstrationsprojekte des Physik- oder des Chemieunterrichts sind doch offensichtlich (maschinentechnisch) so weit von industriell eingesetzten Maschinen und Verfahren entfernt, dass die Brücke zwischen Demonstrationsexperiment und Realität, auch

wenn die Prinzipien im Experiment korrekt herausgearbeitet werden, konzeptionell recht schmal wird und daher nur all zu leicht bricht.
Im Informatikunterricht können wir durch *Programmierung einfachster Bauteile* (ich meine da keinen komplexen Laptop, eher schon einen einfachen Spielzeug-Roboter oder primitive Schaltkreise, die Lichtsignale senden) die Grenze zwischen Hardware und Software und das Wesen der Konstruktion wie auch das Wesen von Konstruktionsfehlern unmittelbar erkennen. Wir können wahrscheinlich auch viel leichter als bei anderen technischen Geräten erkennen, dass wirkliche Computer eben „nur" leistungsfähiger sind und wirkliche Software „nur" größer ist. Dass dieses „nur" dabei in der Regel falsch eingeschätzt wird, soll nicht stören. Wir wollen unsere SchülerInnen ja nicht zu InformatikerInnen ausbilden. Wir wollen sie nur technikfit machen.

4.3 Prüfe Anweisungen bevor sie zum Gesetz oder zur Regel erhoben werden

„Wenn wir schon nicht alle SchülerInnen zu InformatikerInnen machen wollen, zu Angehörigen gesetzgebender Körperschaften wollen wir sie auch nicht machen." – Sicher nicht! Doch bei der Festschreibung von Gesetzen und Regeln dürfen wir nicht nur an Mitglieder des Nationalrats denken. Die moderne Gesellschaft basiert auf einem komplexen, teils gesatztem teils frei antizipertem, teils auf ethischem oder kulturellem Hintergrund basierenden Regelwerk. Dies gilt für die Gesellschaft als Ganzes, für ihre öffentlichen wie privaten Teile wie Gebietskörperschaften oder Unternehmen, sowie für Untereinheiten (Abteilungen) dieser und letztlich bis hinein in die Familien.
Da laufend neue Situationen auftreten, sind auch laufend neue Entscheidungen nötig, die in der Antizipation künftig ähnlicher Situationen leicht zu Regeln werden, denen teils impliziter, teils expliziter Vorschriftscharakter zukommt. Das alles ist nicht neu. Die daraus resultierenden Phänomene und Konsequenzen werden von Soziologie und Jurisprudenz, teils von Psychologie und Wirtschaftswissenschaften analysiert. Doch dies sind alles keine Schulfächer. Wohl deshalb nicht, weil die Materie zu schwierig ist und man zu bald in Spezifika abgleiten müsste, denen es wiederum des Charakteristikums der Allgemeinbildung ermangelt.
Warum könnte guter Informatikunterricht hier in die Bresche springen und wenigstens das Prinzip der Überregulierung und der voreiligen (nicht ausreichend geprüften) Regulierung aufzeigen und Komplementärstrategien anbieten? Weil er ein konstruktives Fach ist, dessen Konstruktionen relativ rasch umgesetzt und getestet werden können.

> *In einem technischen Fach wie Informatik ist ein Entwurf erst dann als realisierbar zu betrachten, wenn ausreichende Tests ergaben, dass dieser Entwurf an sich korrekt ist und dass er auch auf Implementierungsebene korrekt umgesetzt wurde.*

Daraus folgt unmittelbar die bereits erwähnte Prüfungsverpflichtung. Es folgt in weiterer Folge jedoch auch die notwendige Offenheit des kreativ entwickelnden Partners gegenüber Kritik. Sei es die psychologisch etwas leichter ertragbare „Kritik" der Maschine, die emotionslos feststellt, dass die Regelungen der Entwicklerin bzw. des Entwicklers, die ja nur sprachliche Ausdrücke waren, eben nicht das ausdrücken, was ausgedrückt werden sollte, oder sei es die Kritik eines Peers innerhalb eines Reviews, der/die feststellt, dass hier aus der Sicht der prüfenden Person etwas anders bewirkt werden wird, als vorgesehen. Entwickler müssen lernen, Kritik an ihrem Werk zu akzeptieren und kritisierende Partner müssen lernen, ihre Kritik so zu verfassen, dass sie konstruktiv, möglichst präzise und nicht verletzend ist.

> *Informatik lehrt Kritik zu akzeptieren und in fortgeschrittenem Unterrichtsstadium auch Kritik in angemessener Form zu artikulieren.*

Dass die beiden letztgenannten Kriterien nicht trivial sind, erfahre ich nicht nur auf universitärem Niveau, wo ich Seminare mit Peer-Reviews der Seminararbeiten durch andere Seminarteilnehmer abhalte. Wir erleben Defizite in beiden in diesem Kapitel genannten Aspekten im beruflichen wie im schulischen Umfeld und nicht zuletzt im Beziehungsgeflecht zwischen Jugendlichen und Erwachsenen.

4.4 Informatik ist ein Teamsport

„Schon die Überschrift sagt, dass wir dazu nicht Informatik benötigen. Wir haben doch ohnehin den Turnunterricht!“ – Ja selbstverständlich nehmen im Turnunterricht Spiele eine bedeutende Rolle ein und daher ist Teambildung ebenfalls ein wesentliches Element des Turnunterrichts. Allerdings ist in der Regel die körperliche Ertüchtigung Primärziel des Turnunterrichts und der Transfer von Erlebnissen dieses Unterrichts geht doch eher in spätere Freizeitaktivitäten denn in berufliche Aktivität.
Doch modernes Berufsleben, auch das Berufsleben von InformatikerInnen, ist durch Teamaktivitäten charakterisiert. Im Team zu arbeiten kann erfüllend sein. Es kann aber auch zur Qual werden, wenn sich Team-Mitglieder nicht an die Regeln des Teams halten. Dieses Einhalten von Regeln ist jedoch im etwas fortgeschrittenen Informatikunterricht wohl so klar wie kaum sonst wo zu vermitteln. Die Zugänge zu diesem Bildungsinhalt sind mannigfach:

- *Im Peer-Review vertraue ich meinen prüfenden KollegInnen, dass sie jene Fehler, die mir unterlaufen sind, entdecken.*
- *Bei Umsetzung der fundamentalen Idee der Modularisierung entstehen Schnittstellenvereinbarungen. Ich werde nur dann einen wertvollen Beitrag zum Gesamtprojekt leisten können, wenn ich mich entweder strikt an diese Vereinbarungen halte oder, wenn dies nicht möglich sein sollte, zum frühestmöglichen Zeitpunkt mit meinen Partnern in einen Aushandlungsprozess über neue Schnittstellen trete. (Letzteres sollte allerdings nicht zu oft passieren.)*
- *Betriebssysteme koordinieren den Ablauf im Computer. Damit dieser möglichst rasch erfolgt, laufen mehrere Aktivitäten zeitlich verschränkt (quasi-parallel) ab. Dies bedingt Unterbrechungen. Begriffe wie Cycle-Stealing tauchen auf. Das Ganze kostet einiges an Overhead. Dennoch lohnt sich der Aufwand um mehreren Programmen eine faire Chance auf rasche Beendigung zu geben und insgesamt hohen Durchsatz zu erreichen.*
- *Einzelne können interessante Datenbestände aufbauen. Trägt man diese zusammen, entsteht ein wesentlicher Mehrwert. Am deutlichsten sehen wir dies durch die Vernetzbarkeit von Datenbeständen im World-Wide-Web.*

Die vier obgenannten Aspekte lassen sich aus sehr unterschiedlichen Blickwinkeln auf Informatik und den Unterricht dieses Faches ableiten. Sie führen jedoch alle zu demselben Ziel, dem Erkennen des Wertes von Kooperation. Sie zeigen aber auch, dass Kooperation keine Einbahnstraße ist und dass sie nur dann erfolgreich sein wird, wenn jeder Kooperierende darauf verzichtet, lokale Optima anzustreben sondern sich dem Globalziel unterordnet. Andererseits zeigt die Analyse von Betriebssystemen den Unterschied zwischen unterbrechbaren (preemptive) und nichtunterbrechbaren Aufgaben (non-preemptive tasks).
Der Zusatzaspekt - *Kooperation bedingt ein Zurückstecken von Individualzielen* - tritt wohl am anschaulichsten im Zusammenhang mit Modularisierung vor Augen. Doch auch im Zu-

sammenhang mit WWW oder anderen Internetdiensten wird es nicht schwierig sein, Jugendlichen vor Augen zu führen, dass dies nur dann funktionieren kann, wenn durch Standardisierungen, also vorweg ausgearbeitete und dann eingehaltene Vereinbarungen, überhaupt Kommunikationsfähigkeit geschaffen wurde.

4.5 Vergessen können ist eine Qualität

„Welch schreckliches Postulat! Wir unterrichten doch, dass sich die uns Anbefohlenen merken, was wir unterrichten." – Hoffentlich! Aber das Leben ist mehr als nur Schule. Vieles was wir erleben und erlernen, sollen und dürfen wir uns merken. Nur so wird letztlich ein erfülltes Leben entstehen. Doch das Leben hat nicht nur Sonnenseiten, und ich muss auch in der Lage sein, mit Menschen zusammenzuarbeiten, mit denen ich Erlebnisse hatte, die besser niemals stattgefunden hätten. Glückliche Menschen sind in der Lage, über solche Erfahrungen „Gras wachsen zu lassen". Solche, die dazu nicht in der Lage sind, haben es schwerer.
Hier sind wir an einer Stelle angelangt, an der sich menschliche Informationsverarbeitung von maschineller Informationsverarbeitung fundamental unterscheidet. Was auf einem persistenten Datenspeicher abgelegt ist, versickert nicht und wird daher auch nicht „vergessen". Schlimmstenfalls wird das Speichermedium kaputt. Doch selbst in diesem Fall sichert ein ausgeklügeltes Datensicherungssystem, dass es (hoffentlich) irgendwo Duplikate gibt, aus denen wichtige Datenbestände rekonstruiert werden können. Eigentlich bietet die Informatik aus klassischer Schul- und SchülerInnenperspektive somit eine heile Welt des Nichtvergessens.
Doch so heil ist diese Welt nicht. Nichtvergessen ist so lange eine gute Strategie, als wir uns in einer statischen Umgebung aufhalten. Doch die Welt in der wir leben, ist dynamisch. Der gestern richtige Satz kann übermorgen falsch geworden sein. Menschen sind anpassungsfähig. Sie interpretieren. Die Bandbreite, die Interpretation bieten kann, sieht man vielleicht am deutlichsten bei einem Vergleich moderner und fundamentalistischer Strömungen in den verschiedenen großen Weltreligionen. Doch diesen Blick von einem konfessionell geführten Religionsunterricht zu verlangen, überfordert diesen in der Regel.
Informatik ist jedoch ein Feld, in dem rund um Begriffe wie Privacy und Datenschutz einerseits, Evolution von Software- und Informationssystemen andererseits, die Frage von Erinnern und Vergessen wertneutral diskutiert werden kann.

> *Informatiksysteme vergessen nur, wenn man dies explizit anstößt. Dies ist in vielen Fällen erwünscht, manchmal wird es jedoch auch zur Last. Wir müssen lernen, den Wert von Vergessen und von Erinnern richtig einzuschätzen.*

Für sich genommen, mag diese Weisheit noch etwas abstrakt klingen. Wir können sie jedoch, insbesondere mit Blick auf das Social-Web auch etwas anders formulieren:

> *Daten leben lang und können missverwendet werden. Überlege Dir, welche Daten Du der Öffentlichkeit bereitstellst.*

Mit diesem Aspekt verlassen wir die rein technischen Teilgebiete der Informatik und wenden uns Datenschutzaspekten und gesellschaftlichen Bezügen der Informatik zu. Das mag für einige Schüler uncool sein und LeherInnen haben genug zu tun, wenn diese „ohnehin nicht so wirklichen Informatik-Gebiete, auch wenn sie im Lehrplan vorkommen" aus Zeitgründen und quasi im beiderseitigen stillschweigenden Einverständnis unter dem Tisch fallen. Man sollte jedoch bedenken, dass dadurch auch ein wichtiges allgemeinbildendes Lehrziel aufgegeben wird.

4.6 Welche Farbe hat diese Information?

„Information besteht aus in ihrem Kontext interpretierten Daten. Da ist kein Raum für Farbe." – Oh doch! Lesen Sie die Begründungen für den Einmarsch der US-Army in den Irak in der New York Times bevor dieser stattfand, nachdem er stattgefunden hatte, aber jene Massenvernichtungswaffen, die von IAEO-Inspektoren nicht gefunden wurden auch von den Invasionskräften nicht aufgespürt werden konnten und als weitere Leseprobe nachdem Saddam Hussein entdeckt, gefangen und verurteilt wurde. Vergleichen Sie die Informationen, die in diesen Artikeln eines guten amerikanischen Publikationsorgans erschienen, mit Information in den dazu passenden Artikeln eines guten europäischen (deutschsprachigen) Publikationsorgans, vielleicht auch noch mit den Artikeln eines Organs des Boulevards. Die Schülerinnen sollen dann untersuchen, ob beispielsweise die deutschsprachige Information einfach eine Übersetzung der englischsprachigen war oder ob darüber hinaus weitere Unterschiede vorlagen. Ob der Boulevard nur eine Verkürzung des Qualitätsjournalismus darstellt oder ob hier nicht auch Interpretationen und Längungen, also schlicht Färbungen eingezogen werden. Das Experiment kann in verschiedene Richtungen ausgebaut werden.
„Gut, so ein Experiment mag ja ganz wertvolle Bildungsinhalte bieten. Aber es hat noch immer nichts mit Informatikunterricht zu tun. Das sollen doch die KollegInnen, die Deutsch oder Englisch unterrichten, leisten." – Vielleicht. Es ist nichts dagegen einzuwenden, wenn diese Kollegen oder Kolleginnen sich auch der Interpretation von politischer und entscheidungsrelevanter Information annehmen. Man könnte derlei auch mit Fug und Recht in den Geschichtsunterricht verbannen. Es wäre jedoch falsch, es dann aus dem Informatikunterricht auszuklammern, wenn man sich nicht sicher wäre, dass Informationsfärbung nicht in einem dieser anderen Fächer mit Sicherheit behandelt wird. Darüber hinaus unterscheidet sich Information, die Jugendliche aus dem Web beziehen von Information in klassischen Informationsmedien. Bei letzteren ist die „Färbung" in der Regel konstant und daher bekannt. Bei Informationen, die man aus dem Internet bezieht gilt dies nicht.
Es gibt also mehrere Gründe, sich mit Informationsfärbung zu beschäftigen. Schließlich darf Informatikunterricht ja nicht zum Computer-(Science)-Unterricht schrumpfen. Dies bringt uns wieder ein wenig zum Spannungsfeld Informatikunterricht versus IKT-Unterricht. In letzterem haben Themen, wie das eben aufgezeigte, tatsächlich nichts verloren. Es geht ja nur um Computer-Literacy.

> *In Informatik geht es aber um die Verarbeitung von Information, damit auch um die Interpretation und um den Interpretationsraum von Daten.*
> *Die Interpretation von Daten trägt stets subjektive Elemente. Diese Subjektivität kann vom menschlichen Interpreten eingebracht werden oder sie wird vom Program (vom Programmierer, von der Erstellerin des Datenbank-Schemas) zur Zeit der Erstellung dieses Programms bzw. der Definition der Datenbank eingebracht. Sie wird aber erst zur Laufzeit des Programms bzw. der Verwendung der Datenbank wirksam.*

Wir bleiben aus schulischer Sicht mithin im Bereich gesellschaftlicher Bezüge der Informatik und im Bereich Datenbanken und Informationssysteme. Dass als Einleitungsexperiment die Analyse von Printmedien, und somit „alte Technologie" vorgeschlagen wird, hat lediglich den Sinn, dass man an bekannte Themen anknüpft und dadurch genügend Zeit bleibt, zu reflektieren. Diese Zeit besteht nicht, wenn Daten automatisch zu Informationen verdichtet werden und schnelle Computer dadurch aus der Zusammenschau einer Vielzahl von Datenelemente in Windeseile Konsequenzen ableiten.
Die dadurch entstandene Problematik hat man im Bereich des automatisierten Börsenhandels spätestens seit dem Börsenkrach 1989 erkannt und in diese Art von Software automatische

„Bremsen“ eingebaut, sodass nach Abkühlphasen, Phasen menschlicher Interpretation und notfalls Börsenschließtagen der Handel mit kühlem Kopf und realistischerer (?) Interpretation der Daten wieder aufgenommen werden kann. Schwieriger ist es, wie der vorerst unerklärliche nur ganz kurz andauernde extreme Kurssturz des Euro am 6. Mai 2010 zeigte, hochvolumige Arbitragegeschäfte in den Griff zu bekommen, insbesondere dann, wenn ein Player plötzlich diesen Markt verlässt.
Doch Interpretationsfehler treten nicht nur an der Schnittstelle zwischen informationsverarbeitenden und sozialen Systemen auf. Auch spektakuläre technische Misserfolge sind auf Interpretationsfehler von Daten rückführbar.
Das Börsenbeispiel zeigt (halbautomatische Kriegsführung oder der Abschuss eines Zivilflugzeugs durch eine US Fregatte, würde es noch drastischer zeigen), dass zwischen der Interpretationsproblematik und der Verarbeitungsgeschwindigkeit von Daten ein sehr enger Zusammenhang besteht. Diesen wichtigen Aspekt einer auf automatische Informationsverarbeitung abgestützten Gesellschaft können wir sicherlich nicht mehr an die Kollegenschaft, die Sprachen oder Geschichte unterrichtet, delegieren.

4.7 Kreativität braucht Freiheit

„Ja, das ist bekannt. Insbesondere wissenschaftssoziologische Studien über den Zusammenhang wissenschaftlicher Produktion und relativer politischer Freiheit in der früheren UdSSR haben dies nachgewiesen. Letztlich ist die betriebswirtschaftliche Motivationsliteratur doch voll mit derartigem. Im Informatikunterricht haben wir aber genug Wichtigeres zu tun, als uns mit derlei auseinanderzusetzen.“ – Vielem ist zuzustimmen. Allerdings ist Wissenschaftssoziologie sicherlich kein Schulfach und selbst in Wirtschaftskunde wird man kaum auf betriebswirtschaftliche Motivationsliteratur zurückgreifen. Doch selbst wenn, in den Ohren der Jugendlichen wären dies nette Erzählungen, die man ja vielleicht glauben kann. Erleben kann man dies im Unterricht nicht.
Anders im Informatikunterricht. Hier erklärt man doch den Computer als deterministische Maschine, die von deterministischen Programmen gesteuert wird. So war auch die bisherige Argumentation dieses Aufsatzes aufgebaut. Allerdings wurde dabei der Bereich der künstlichen Intelligenz (AI) bewusst ausgeklammert. AI ist m.E. auch nichts für den Anfängerunterricht in Informatik. Begänne man zu früh damit, könnte man mit verkürzten Darstellungen von ELIZA oder aktuellen Hochleistungs-AI-Systemen Faszination erwecken, kaum jedoch Verständnis dafür, was AI, was regelbasierte Systeme oder Metaheuristiken leisten können. Faszination zu wecken ist zweifellos positiv, doch darf Faszination nie zu falschen Vorstellungen führen. Dieses Risiko ist jedoch groß, wenn man das Pferd beim Schwanz aufzäumt (siehe auch Abschnitt 4.2).
Dennoch, fortgeschrittener Informatikunterricht sollte, ja muss vielleicht, auch darauf hinweisen, dass es neben deterministischen Algorithmen und prozeduraler bzw. objektorientierter Programmierung noch Alternativen gibt.
Als guten Einstieg sehe ich regelbasierte Programmierung, etwa in Prolog. Hier lässt sich auf Basis der klassischen Eltern-Vorfahren-Beziehung etwa das lokale Flusssystem modellieren. Dann könnte man erschließen lassen, ob sich eine lokale Wasserverschmutzung in Richtung Schwarzes Meer oder Nordsee ausbreitet.
Damit ist noch nicht viel in Bezug auf Kreativität und Neues geleistet. Doch immerhin sieht man, dass dieses Programm zwar die Topologie des Flusssystems exakt abgebildet haben muss (eine völlig deterministische Komponente), doch der Ort der Verschmutzung sowie der Ort, an dem geprüft werden soll, ob diese dort jemals ankommen kann, sind frei wählbar. Die Prüfung muss auch keineswegs der Fließrichtung entsprechen. Man könnte ja etwa fragen, ob eine Verschmutzung der Möll bei Winklern später in Hermagor detektiert werden kann.

Spannender noch, man kann das Programm auch quasi in die Gegenrichtung laufen lassen. Welche österreichischen Flüsse müssen untersucht werden, wenn in Lavamünd eine Verschmutzung entdeckt wurde?
Genug der Fragen, langsam wird es Zeit für Erklärungen über die Rolle der Inferenzmaschine und die elementaren Prinzipien der Unifikation. Der Charme dieses Ansatzes besteht in seiner vollständigen Erklärbarkeit. Diese ist bei der Anwendung von Metaheuristiken nur mehr bedingt gegeben.
Als Einstieg in heuristisches Problemlösen sind vermutlich Genetische Algorithmen am besten geeignet. Sie zeigen das Prinzip und knüpfen an ein für Jugendliche nicht unspannendes Thema an. Sie erfordern zwar elementare Kenntnisse in Wahrscheinlichkeitsrechnung, doch keinen aufwändigen mathematischen Apparat. Jedenfalls sind sie geeignet, ein von konventioneller Programmierung eklatant unterschiedliches Problemlösungsverfahren zu zeigen, das jedoch für sich genommen wieder in Form eines konventionellen Programms (unter Aufruf einer Zufallsfunktion) realisiert wird. Die Problemstellung, die mit diesem Algorithmus gelöst wird, ist ein Optimierungsproblem. Welches? Gleichviel! Eines, das für die Klasse als spannend erscheint.
Die Erkenntnisse, die aus diesem Ausflug in die AI gewonnen werden können, sind mannigfach.

- *Durch Auflösen der in klassischer Programmierung fix vorgegebenen Struktur entsteht neues Potential. (Allerdings verliert man dadurch an Effizienz.)*
- *Durch Freigabe von Zufall als Lösungskomponente lassen sich zwar zufällige, aber systematisch generierte, Lösungen erarbeiten, die zusehends besser einer vorgegebenen Zielfunktion genügen.*
- *Wenn man Zufall zulässt, bedarf es mehrerer voneinander unabhängiger Experimente, um in die erarbeitete Lösung Vertrauen zu haben.*
- *Heuristische Programmierung erzielt sehr gute Lösungen. Ob wirklich die beste dabei ist, können wir aber leider nicht feststellen.*

Wenn man schon auf algorithmische Komplexität zu sprechen gekommen ist, lässt sich auch noch folgende wichtige Erkenntnis ziehen:

> *In manchen Fällen würde wohl ein deterministisches Verfahren existieren, mit dem die wirklich optimale Lösung berechnet werden kann. Doch dieses durchzuführen würde zu lange dauern (vielleicht selbst auf schnellsten Rechnern Jahrzehnte oder Jahrhunderte). Man muss sich daher in solchen Fällen an Stelle von optimalen mit sehr guten Lösungen zufrieden geben.*

Fasst man all dies zusammen und wirft einen distanzierten Blick auf Problemlösungsverfahren und Programmierparadigma, die bis zu diesem Zeitpunkt erarbeitet wurden, bleibt wohl als Resümee bestehen:

> *Durch Auflösung rigider Strukturen entstehen Freiheiten, die selbst von der Maschine genützt werden können. Und dies, obzwar wir ja längst wissen, dass die Maschine aus sich heraus keinerlei Kreativität hervorbringen kann. Was bei den Metaheuristiken vielleicht wie maschinelle Kreativität erscheint, ist nichts anderes als das Zusammenwirken von zufällig entwickelten Lösungsvorschlägen mit der durch die Zielfunktion vorgegebenen Bewertungsstrategie. Dies, gepaart mit dem Prinzip des Überle-*

bens (und Weiterwirkens) guter Lösungen (survival oft the fittest) führt letztlich zum Erfolg. Doch selbst dabei müssen wir bescheiden bleiben. Ein zu großes Maß an Erfolgsdruck führt dazu, dass sich diese Algorithmen in einem lokalen Optimum fangen und dadurch das Globalziel verfehlen.

Nochmals: Um zu derartigen Erkenntnissen zu gelangen, müssen wir uns jeglicher „AI-Zauberei“ enthalten. Wenn wir es jedoch schaffen, wenigstens einige dieser Erkenntnisse nicht vorzubeten, sondern durch eigene Konstruktionen der Schülerinnen und Schüler erarbeiten lassen, bleibt Bildung zurück, die losgelöst von ihrer Entstehungssituation im Informatikunterricht allgemein angewendet werden kann.

4.8 Die Universalität kleinster Teilchen

„Gut, aber über den Aufbau von Atomen wird zweifellos im Physikunterricht gesprochen. Computer werden zwar immer kleiner und leistungsfähiger und Halbleiter basieren auf physikalischen Prinzipien. Aber lassen wir die Halbleiterphysik doch bei unseren KollegInnen.“ – Ja selbstverständlich. Wir sollten zwar, wenn elementare Hardware besprochen wird, darauf hinweisen, dass diesbezüglich im Physikunterricht schon (?) vieles behandelt wurde. Doch lassen wir jenen, die das Fach und die zugehörige Fachdidaktik beherrschen, ihr Terrain. Dennoch meine ich, dass wir das Thema der Kleinteiligkeit aus Perspektive der Informatik nochmals aufgreifen sollten.
Der Computer wird schon frühzeitig als universelle Maschine dargestellt. Die Begründung dafür wird in den von Neumann-Prinzipien, insbesondere in der Speicherprogrammierung gesehen. Dies reicht, um die Flexibilität von klassischen Großrechnern zu erklären und es reicht fast aus, um die Flexibilität eines PCs zu erklären. Doch reicht es wirklich aus, um das Anwendungsspektrum eines PCs oder eines Handys zu erklären?
Gemäß von Neumann gibt es Daten und Programme. Beide liegen in demselben Speicher. Das Programm entscheidet, ob der Inhalt eines Speicherworts als Datum oder als Anweisung zu interpretieren ist. So weit so gut. Daten sind wahrscheinlich Zahlen (Computer, Rechner) und Programme sind Anweisungen in irgendeiner Programmiersprache.
Dieses klare Bild kommt etwas ins Wanken, wenn wir die Funktionsweise eines Compilers betrachten. Er fasst unser Programm (mehrheitlich Text und kaum Zahlen) als Eingabedaten auf, verarbeitet es und gibt ein funktional äquivalentes Programm in Binärcode aus. Erst dieser Binärcode kann von der Maschine interpretiert und ausgeführt werden. Doch diese erste Erkenntnis reicht noch nicht aus, um zu begründen, warum Computer heute nicht bloß Rechner und Textverarbeitungsmaschinen, sondern auch Telefonapparate, Fotoapparate, Film- und Fotobearbeitungsgeräte, und vieles mehr sind. Hier werden von ein und derselben Maschine doch Datenformate bearbeitet, denen scheinbar keinerlei Gemeinsamkeit mehr anhaftet.
So ist es. Auf der Anwendungsebene sind diese Datenformate so weit voneinander entfernt, dass die Gemeinsamkeiten verschwinden. Was bleibt als Gemeinsames? Die Codierung! – Letztlich ist ein heutiger Computer eine Bitverarbeitungsmaschine. Was er nicht in Form von Bitfolgen präsentiert bekommt, ist unverdaulich. Also müssen sämtliche Daten, welches externe Erscheinungsbild sie auch immer haben mögen, in Bitfolgen umgewandelt (codiert) werden. Der größte gemeinsame Nenner der Datenverarbeitung ist also das Bit, die kleinstmögliche Einheit der Informationsverarbeitung.
Wir gewinnen daraus die Erkenntnis, dass die Universalität heutiger Rechner eigentlich sehr stark darauf basiert, dass Computer „nur“ Bits verarbeiten können. Durchaus gemeinsam mit den Erkenntnissen des Physikunterrichts eröffnen sich dadurch zwei getrennte Wege zur Aussage

Die Welt ist so vielfältig wie sie ist, weil sie aus unterschiedlich kombinierten kleinsten Teilchen aufgebaut ist. Dies gilt für die physische Welt ebenso wie für die Welt der Daten und Informationsdarstellung.

4.9 Problemlösen als Übersetzungsproblem

„Klar, Übersetzen ist für viele Schüler ein Problem. Aber das heißt noch lange nicht, dass Problemlösen etwas mit Übersetzen zu tun hat. Hier wurde doch die Richtung der Kausalität vertauscht." – Das scheint so, wenn wir Übersetzen als die Transkription eines Textes von einer natürlichen Sprache in eine andere auffassen. Doch in Informatik haben wir es mit natürlichen Sprachen nur auf der Ebene der Problemformulierung zu tun. Bereits am Ausgabemedium treten bei den meisten Anwendungen Sprachkonstrukte auf, die nur Verkürzungen natürlichsprachlicher Texte sind. Dazwischen bewegen wir uns auf einem formalsprachlichen Niveau. Dies gilt für die Programmiersprachen. Dies gilt aber auch für die zu verarbeitenden Daten, die einer bestimmten Syntax und Semantik genügen müssen, um von der Maschine bzw. vom Programm interpretierbar zu sein.
Abschnitt 4.8 zeigte eben, dass es neben den algorithmischen (prozeduralen und objektorientierten) Sprachen etwa auch regelbasierte Sprachen gibt. Während bei ersteren ein Verfahren in sämtlichen erwarteten Varianten seines Ablaufs anzugeben ist, sind letztere deklarativ. Dies bedeutet, man gibt ein erwünschtes Ergebnismuster an und überlässt es dem Computer, den Weg von der Problemstellung zum Ergebnismuster selbst zu finden.
Wir kommen somit zur ersten Aussage dieses Abschnitts:

Problemlösen kann als Suchproblem aufgefasst werden. In welcher Form die Suche zu gestalten ist, hängt vom Wesen der Problemformulierung ab.

Diese Problemformulierung wird in vielen Fällen in deklarativer Form erfolgen. Erfolgt sie jedoch in prozeduraler Form, bemühen sich InformatikerInnen, sie erst einmal in eine deklarative Form zu bringen, weil diese einen Lösungsraum mit mehr Freiheitsgraden bietet, als die Vorab-Festlegung eines bestimmten Algorithmus. Diese Aussage gilt nicht nur für die Lösung von Informatikproblemen.

Effizientes Problemlösen bedingt, dass man sich das Problem erst einmal so herrichtet, dass man es mit dem verfügbaren Instrumentarium bestmöglich bearbeiten kann und es so entweder der Lösung zuführt oder in eine Form transformiert, die es lösbarer erscheinen lässt, als die ursprüngliche Formulierung.

Somit sind wir aber bei der Aussage gelandet, dass Problemlösen auch ein Übersetzungsproblem ist. Wir müssen die Problemformulierung, die in der Fachsprache der Problemsteller erfolgte, in eine Form bringen, die InformatikerInnen erst einmal verstehen. Das erfordert Dialog. Anschließend müssen die InformatikerInnen das so verstandene in die ihnen geläufige Sprachen- und Gedankenwelt transformieren. Wieder ein Übersetzungsproblem. Dort soll es so formuliert werden, dass der Lösungsraum noch möglichst offen bleibt und erst sukzessive eine Transformation von der informatisch verstandenen Problemformulierung in eine informatische Problemlösung stattfindet. Dabei wünscht man sich, ohne Übersetzungsproblem auszukommen. In der Praxis gelingt dies in den seltensten Fällen.
Man muss nicht den oben gezeichneten Weg über deklarative Sprachen gehen und man wird in der Schule nicht mit rein deklarativen Spezifikationssprachen arbeiten. Problem- und Lösungsformulierungen in Teilaspekten von UML ist jedoch durchaus ein der Schule zugänglicher Weg [Hubw 08]. Er bietet bereits die Übersetzungsfolge natürliche Sprache - UML -

Implementierungssprache. Ein anderer leicht gangbarer Weg wäre, deklarative Sprachen über Datenbank-Subsprachen einzuführen und etwa einige SQL-Queries mit ihrer prozeduralen Ausformulierung zu vergleichen.
Wichtig erscheint es, in diesem Zusammenhang darauf hinzuweisen, dass man, wenn man Problemlösen im Wege des Zwischenschritts der Modellierung unterrichtet, wirklich die gesamte Reise von der Problemstellung bis zur (schuladäquaten) Lösung durchwandert. Irgendwo auf halbem Weg (etwa nach der zweifellos wichtigen Modellierung) abzubrechen, lässt Verwirrung im Kopf der SchülerInnen zurück. Lieber an einem überschaubaren Problem die Vorteile der nötigen Übersetzungsschritte zeigen und diese zu üben, als den Versuch zu starten, die Welt „einzureißen“ und doch stecken zu bleiben, bevor erprobt werden konnte, ob das entworfene Modell tatsächlich implementierbar und realitätstauglich ist.

4.10 Problemlösen als Strukturierungsproblem

„Endlich ein Thema, das zweifelsfrei in den Informatikunterricht passt.“ – Freut mich. Doch bitte fassen Sie es nicht so auf, als wäre Informatik das einzige Fach in dem strukturiert und in weiterer Folge vielleicht modularisiert wird. Das Divide-and-Conquer-Prinzip finden wir praktisch in jeder technischen Fachdisziplin und wie unschwer zu erkennen ist, hat es einen martialischen Hintergrund. Dem entsprechend ist es uralt.
Dennoch, Informatik ist eben das einzige technische Schulfach und darüberhinaus haben Strukturierung, Modularisierung und divide-and-conquer in der Informatik (erinnern wir uns der fundamentalen Ideen) eine besondere Bedeutung. Weiters ist es unmittelbar einsichtig, dass man sich im Werkunterricht eine Skizze oder einen sauberen Entwurf macht, bevor man die Säge an ein Brett ansetzt. Doch im Informatikunterricht könnte das ja anders sein. Man arbeitet ja nur mit Daten und die sind doch leicht wiederherstellbar, wenn sich der erste Lösungsansatz als Flopp herausstellt.
Nein, Informatikunterricht ist weder Bastel- noch Probierunterricht. Man darf, ja muss gelegentlich, schon explorativ arbeiten. Prototyping und explorative Entwicklung sind sogar Profimethoden und sie können manchmal sehr motivierend sein.
Jedoch *„Think first. – Act later!“* ist nicht nur ein wichtiges Prinzip für die Entwicklung technischer Artefakte. Es ist ein wichtiges Prinzip für jegliches soziale Zusammenleben. So gilt es etwa für jede Art von Kommunikation in der Abwandlung: *„Think first. – Speak later!“*
Beim Begriff Kommunikation sind wir jedoch wieder auf der Informationsebene und damit beim primären Gegenstandsbereich des Fachs Informatik. Somit mag es dem Informatikunterricht gut anstehen, auch dieses Prinzip experimentell erfahrbar zu machen. Meist bekommt man die Ergebnisse von der Klasse ohnehin frei Haus und ohne jegliche LehrerInnen-Intervention geliefert.
Aufgabe der Lehrkraft bleibt hier, die Phänomene in einer Form ans Licht zu bringen, dass es dabei keine übertriebenen Sieger und Verlierer gibt. Doch dies wird aufgrund der schon in Abschnitt 4.2 eingeforderten Offenheit gegenüber und dem Erwerb zur Fähigkeit von konstruktiver Kritik ja kein Problem sein.

5 Konsequenzen – Wie fundiert müssen Lehrende sein?

Die in Kapitel 4 aufgestellten Forderungen scheinen weit aus dem Informatikunterricht herauszuführen und diesen daher zu überfordern. Unser imaginärer Gesprächspartner hat dies ja unmissverständlich ausgedrückt. Dem Autor stellt sich jedoch die Frage, ob sie zu anspruchsvoll für die Jugendlichen oder zu anspruchsvoll für große Teile der österreichischen Lehrerschaft sind.
Bei den Jugendlichen bin ich optimistisch. Sehr sogar. Einerseits haben sie noch einen offenen lernfähigen Kopf. Andererseits zeigten Studien, dass der gegenwärtige Informatikunterricht einen sehr starken Motivationskeil in die Klasse treibt. Dies wurde einerseits von [KnSc 07] nachgewiesen, anderseits zeigten [AKLU 07] eine Fülle von Vorurteilen sowohl gegen den Knöpfchen-Drück- als auch gegen den Computer-Kastl-Unterricht. Geht man ein Stück weiter, also vom informationsverarbeitenden Gerät zu Information die vom Gerät, aber letztlich auch von und für Menschen verarbeitbar ist, sollte man beiden Extremgruppen der Klasse (und hoffentlich auch jenen dazwischen) etwas Motivierendes bieten können.
Schwieriger sehe ich die Umsetzbarkeit durch die Lehrenden. Um mit SchülerInnen den gesamten Weg von einer informatischen Fragestellung bis zum Transfer auf allgemeinbildende Inhalte zu gehen, bedarf es eines guten Überblicks über Informatik. Dieser ist jedoch in einem Fach, das zu großen Teilen von Personen unterrichtet wird, die ihre fachliche und fachdidaktische Ausbildung in anderen Lehramtsfächern erhalten haben, nicht einfach bewältigbar. Die von den meisten dieser Unterrichtenden in Anspruch genommene Zusatzausbildung durch Kurse der damaligen Pädagogischen Institute war, den Randbedingungen gehorchend, stets auf bestimmte eng umgrenzte Fragestellungen fokussiert. Das reicht, um das eben Erlernte weiterzugeben. Es reicht aber nicht, um das Fach in seiner Tiefe und Schönheit zu erfassen und Faszination wie Überfachliches weiterzugeben.
Es darf nicht verschwiegen werden, dass es viele motivierte LehrerInnen gab und gibt, die sich darüberhinaus im Selbststudium eine umfassende Wissensbasis aneigneten und die daher sehr wohl in der Lage wären, das eben skizzierte Programm umzusetzen. Vielleicht fehlte es nur eines Anstoßes.
Doch viele sind nicht alle, und daher haben wir uns bemüht einen Hochschullehrgang zu definieren, der ausreichen sollte, Unterricht wie den oben beschriebenen abzuhalten [HoMi 09]. Die Prinzipien dieses Lehrgangs und Querbeziehungen zwischen den dort konzipierten Themensträngen werden in [HoMi 10] vorgestellt.

6 Evaluative Betrachtungen

Eine faire Evaluation der hier unterbreiteten Vorschläge ist Sache der LeserInnen. Dennoch soll hier versucht werden, die unterbreiteten Vorschläge nochmals kritisch zu beleuchten.
Was allgemeinbildend ist, müsste sicherlich viel breiter argumentiert werden, als dies hier der Fall war. Doch das Medium setzt Grenzen. Ob Informatikunterricht überhaupt allgemeinbildend sein soll, mag ebenfalls Gegenstand eines offenen Diskurses sein. Im in Österreich gut ausgebauten BHS-Bereich wird man dies nicht fordern. Dort wäre der hier gebrachte Vorschlag in seiner Extremform interpretiert wohl sogar kontraproduktiv.
Im AHS-Bereich hat das Wort *Allgemeinbildung* jedoch einen anderen Stellenwert und daher sollte sich dort die Informatik getrost dem Kampf um Stunden stellen. Doch dieser Kampf darf nicht nach dem Gesichtspunkt „Wer als Letzter kam, verliert als Erster“ und auch nicht unter dem Gesichtspunkt „Der Stärkere hat die wahreren Argumente und wir Informatiker sind stark, weil wir haben die Industrie und wesentliche Teile der Elternschaft hinter uns“ geführt werden. Wissenskämpfe, und um einen solchen handelt es sich hoffentlich bei einem Kampf um Schulstunden, sollten mit möglichst rationalen Argumenten ausgefochten werden.

In einem solchen Wettstreit hat die Informatik allerdings am Gebiet der Allgemeinbildung weit mehr zu bieten, als vom aktuellen Informatikunterricht tatsächlich geboten wird. Es würde mich freuen, in diese Richtung Anstöße gegeben zu haben.
Die dazu in Kapitel 4 angeführten Topoi erheben keinerlei Anspruch auf Vollständigkeit. Sie sind auch nicht alle im Rahmen des Informatik-Pflichtunterrichts der 5. AHS (9. Schulstufe) ansprechbar. Doch lassen sich einige davon in einem Informatik-Pflichtunterricht umsetzen, der nicht von Themen belastet ist, die mit Informatik nur deshalb in Beziehung gebracht werden können, weil es sich um die Anwendung von Computer- und Informationstechnologie handelt, die aber nicht in der Lage sind, in Denkstrukturen von TechnikerInnen, auch nicht jenen, deren technische Entwicklungen sich auf Informations- und Wissensstrukturen konzentrieren, also InformatikerInnen, einzuführen.
Die zehn Themenstränge, die Kapitel 4 behandelt, sind aus Sicht der Unterrichtsgestaltung nahezu beliebig angeordnet. Jene, die sie im Unterricht erproben, werden andere Sequenzen wählen. Hier wurde eher versucht, einen roten Faden zu finden, der von basalen Überlegungen zu darauf aufbauenden führt und somit von der Klasse schon ein wenig Einblick in die Gedankenwelt der Wissenstechnik Informatik erwartet.
Dass dabei Programmieren in einfachster Form den Anfang macht, wird nicht überraschen. Doch unterliegen Sie nicht dem Irrtum, Programmieren ist (zu) schwer. Dies gilt nur dann, wenn man die Jugendlichen in einem Schwall mit all dem überschwemmt, was bei professioneller Programmierung nötig ist. Es geht mir dabei bewusst um die Exaktheit der Sprache und nicht um eine Programmiersprache. Dass man hier durch „Programme“ in natürlicher Sprache sehr viel erreichen kann, haben Antonitsch et al [AnLS 07] sowie Bischof und Mittermeir [BiMi 08] gezeigt.
Sicherlich gilt, dass an die erwähnten Bildungswerte auch aus anderen Schulfächern herangeführt werden kann. Es mag überraschend, ja am ersten Blick absurd erscheinen, dass ich dafür gerade den Informatikunterricht auswähle. Der Grund dafür ist jedoch einfach. Als technisches Fach wird in Informatik konstruiert. Diese Konstruktionen können zwar einfach sein, sie tragen aber dennoch alle Charakteristika einer umfassenden Informatiklösung. Im Informatikunterricht werden Jugendliche nicht nur informiert sondern sie erarbeiten sich ihre Erkenntnisse im Zuge der vorzunehmenden Entwicklungen und des dabei zu durchlaufenden Prozesses selbst. Dies führt jedoch zu Einsichten, die sonst nicht erreicht werden. Als kleinen Beleg dazu darf ich Aussagen eigener Studierender anführen. Im Rahmen einer Feedback und Evaluationssitzung über ein neu eingeführtes Software-Entwicklungssimulationssystem kam Kritik zu einigen Details, im Wesentlichen jedoch viel Lob über den Erkenntnisgewinn mit diesem System. Ich stellte darauf hin fest, dass all diese Erkenntnisse doch bereits im 3. Semester vorgetragen wurden. Antwort darauf: „Ja, aber nun haben wir es erlebt!“
Schließlich, wenn wir schon bei der Exaktheit der Sprache sind, muss ich auch für die Exaktheit von Begriffen plädieren. Wenn IKT-Unterricht und Informatikunterricht von Inhalt und Zielsetzung so wenig Gemeinsames haben, wie dies bei den derzeitigen Manifestationen der Fall ist, wäre es den Schülerinnen und Schülern gegenüber mehr als fair, diese Unterschiede durch entsprechend unterschiedliche Fachbezeichnungen auszuweisen.
Dass der in [HoMi 09, HoMi 10] vorgestellte Universitätslehrgang geeignet ist, LehrerInnen zu befähigen, die in Kapitel 4 postulierten Ziele in ihrem Unterricht umzusetzen ist derzeit noch eine unbewiesene Behauptung. Wir hoffen jedoch, die Lehrerschaft für diesen Lehrgang zu interessieren. Dann sollte es auch möglich sein, die von der Bildungsbehörde nötige Unterstützung zu bekommen.

7 Zusammenfassung

Die vorliegende Arbeit geht von den Visionen jener aus, die vor 25 Jahren den Informatikunterricht an Österreichs Allgemeinbildenden Höheren Schulen (AHS) einführten. Während diese Pioniere noch versuchen mussten mit beschränkten technischen Ressourcen unklare Visionen zu realisieren, kennen wir heute die „Informationsgesellschaft", die sich gelegentlich auch „Wissensgesellschaft", nie jedoch „Computergesellschaft" nennt. Dem sollte in einem Unterricht, der sich Informatikunterricht nennt (und nicht etwa Computerunterricht) Rechnung getragen werden.
Informatik ist ein Kunstwort aus der Verschmelzung von Information und Automatik. Der Wortstamm Computer kommt dabei nicht vor. Dies bedeutet, dass Informatik, obzwar ein technisches Fach, so doch kein gerätespezifisches Fach sein sollte. Wie man dies erreicht – und wie man dabei hoffentlich auch jene Teile der Klasse, die ein „Gerätefach" ablehnen erreicht – wurde in Kapitel 4 skizziert. Doch dies zu erreichen setzt ein höheres Maß an Informatik-Fachkompetenz der Lehrenden voraus, als heute landläufig gegeben ist. Durch gut gestaltete Lehramtsstudien in Informatik sowie durch den in [HoMi 09, HoMi 10] skizzierten Universitätslehrgang ist dieses jedoch erreichbar.
Als Nutznießer des Konzepts sehen wir einerseits die Jugendlichen, denen ein technisches Fach und damit Denkstrukturen von TechnikerInnen geboten werden, die im Fach erlebbar aber auch jenseits dieses Faches anwendbar sind. Andererseits gewinnt die Lehrerschaft, die sich nicht mit Klassen abmühen muss, in denen technikaverse Jugendliche passiv bis störend sind und die durch eine tiefere Fundierung im Fach mehr Sicherheit in ihrem Unterricht bieten können. Schließlich nützt dies der Schulbehörde, die bemüht ist, mit neuen Lehrplänen Fächergrenzen zu durchdringen und schulische Kompetenzfelder zu öffnen. Wie kann das besser gelingen, als aus Beiträgen einzelner Fächer heraus zu fachübergreifenden Kompetenzen hinzuführen.

Literatur und Referenzen

[Anto 06] Antonitsch Peter K.: *Databases as a Tool of General Education*. In Mittermeir R.T.: From Computer Literacy to Informatics Fundamentals; Proc. ISSEP 2006, LNCS 4226, Springer 2005, S. 59 – 70.

[AKLU 07] Antonitsch Peter K., Krainer Larissa, Lerchster Ruth, Ukowitz Martina: *Kriterien der Studienwahl von Schülerinnen und Schülern unter spezieller Berücksichtigung von IT-Studiengängen an Fachhochschule und Universität*; IFF-Forschungsbericht, Universität Klagenfurt, März 2007.

[AnLS 07] Antonitsch Peter K., Lassernig Ulrike; Söllei Andreas*: Lehrarrangements in der Informatiklehrerausbildung*; in: Schubert S. (Hrsg.): Didaktik der Informatik in Theorie und Praxis; Proc. 12 GI-Fachtagung „Informatik und Schule – INFOS2007"; Lecture Notes in Informatics 112, Gesellschaft f. Informatik, 2007, S. 91 – 99.

[BiMi 08] Bischof Ernestine, Mittermeir Roland: *Informatik erLeben: Beispiele für schülerinnen- und schüleraktivierenden Informatikunterricht*; Institut f. Informatik-Systeme, AAU Klagenfurt, 2008. (siehe auch: http://informatik-erleben.uni-klu.ac.at)

[DaDS 06] Dagienė Valentina, Dzemyda Gintautas, Sapagovas Mifodijus: *Evolution of the Cultural-Based Paradigm for Informatics Education in Secondary Schools.* In Mittermeir R.T. (Ed.) Informatics Education – The Bridge between Using and Understanding Computers, Proc. 2nd ISSEP 2006, LNCS 4226, Springer, 2006, S. 1 – 12.

[DaDG 06] Dagys Viktoras, Dagienė Valentina, Grigas Gintautas: *Teaching Algorithms and Programming by Distance: Quarter Century's Activity in Lithuania*. In Dagienė V., Mittermeir R. (Eds.): Information Technologies at School, Proc. 2nd Internat. Conf. "Informatics in Secondary Schools: Evolution and Perspectives", TEV Publishing House, Vilnius, 2006, pp. 402 – 412.

[Denn 04] Denning P.J.: *Great Principles in Computing Curricula*. In Proc. 35th SIGCSE technical symposium on Computer Science education, acm, 2004, pp 336-341.

[HoMi 09] Hodnigg Karin, Mittermeir Roland: *Weiterbildungsangebote für Informatik-LehrerInnen Untersuchung des Umfelds und Konzept für Österreich*; Forschungsbericht, Institut für Informatiksysteme, Alpen-Adria Universität Klagenfurt, 2009.

[HoMi 10] Mittermeir Roland, Hodnigg Karin: *Konzept einer stufenweisen Fortbildung für InformatiklehrerInnen*. In diesem Tagungsband.

[Hrom 06] Hromkovič Juraj: *Contributing to General Education by Teaching Informatics*. In Mittermeir R.T. (Ed.) Informatics Education – The Bridge between Using and Understanding Computers, Proc. 2nd ISSEP 2006, LNCS 4226, Springer, 2006, S. 25 – 37.

[Hubw 08] Hubwieser Peter: *Analysis of Learning Objectives in Object Oriented Programming*. In Mittermeir R.T., Syslo M.M. (Eds.): Informatics Education – Supporting Computational Thinking Proc. 3rd ISSEP; LNCS 5090, Springer 2008, S. 142 – 150.

[KnSc 07] Knobelsdorf Maria, Schulte Carsten: *Das informatische Weltbild von Studierenden*; in: Schubert S. (Hrsg.): Didaktik der Informatik in Theorie und Praxis; Proc. 12 GI-Fachtagung „Informatik und Schule – INFOS2007"; Lecture Notes in Informatics 112, Gesellschaft f. Informatik, 2007, S. 69 – 79.

[Luft 89] Luft Alfred L.: *Informatik als Technikwissenschaft – Thesen zur Informatik-Entwicklung*; Informatik-Spektrum Okt. 1989, Jg. 12, Heft 5, S. 267 – 273.

[LuKö 94] Luft Alfred L., Kötter Rudolf: *Informatik – Eine moderne Wissenstechnik*; BI-Wissenschaftsverlag, 1994.

[Mich 09] Micheuz Peter: *Zahlen, Daten und Fakten zum Informatikunterricht an den Österreichischen Gymnasien*. In Koerber, Bernhard: Zukunft braucht Herkunft: 25 Jahre „INFOS – Informatik und Schule"; Proc. INFOS 2009; 13. GI-Fachtagung Informatik und Schule, Springer 2009, S. 243-254.

[Mitt 10] Mittermeir Roland: *Wenn unsere Zeit kurzlebig ist, warum sollte die Schule dann langfristiges unterrichten?* CD Austria, Sonderheft 25 Jahre Schulinformatik – Zukunft mit Herkunft, Juni 2010, S. 11 – 12.

[Reit 05] Reiter Anton: *Incorporation of Informatics in Austrian Education: The Project „Computer-Education-Society" in the School Year 1984/85*. In Mittermeir R.T.: From Computer Literacy to Informatics Fundamentals; Proc. ISSEP 2005, LNCS 3422, Springer 2005, S. 4 – 19.

[Schw 93] Schwill Andreas: *Fundamentale Ideen der Informatik*; Zentralblatt für Didaktik der Mathematik, S. 20 – 31, 1993.

[ScSc 04] Schubert Sigrid, Schwill Andreas: *Didaktik der Informatik*, Spektrum Akademischer Verlag, Heidelberg, 2004.

[Stadt 10] Stadtmann Cornelia: *Tabellenkalkulation im Unterricht*; Diplomarbeit, Universität Klagenfurt, 2010.

[ToBB 08] Tort Françoise, Blondel Françoise-Marie, Bruillard Éric: *Spreadsheet Knowledge and Skills of French Secondary School Student*s. In Mittermeir R.T., Syslo M.M. (Eds.): Informatics Education – Supporting Computational Thinking; Proc. 3rd ISSEP; LNCS 5090, Springer 2008, S.305 – 316.

[Voß 05] Voß Siglinde; *Informatic Models in Vocational Training for Teaching Standard Software*. In Mittermeir R.T.: From Computer Literacy to Informatics Fundamentals; Proc. ISSEP 2005, LNCS 3422, Springer 2005, S. 145 – 155.

[WeOI 04] Wedekind Hartmut, Ortner E., Inhetveen R.: *Informatik als Grundbildung*, Teil (I) bis V; Informatik Spektrum, Jg. 27, Heft 2, April 2004, bis Heft 6, Dez. 2004.