Informatik - didaktische Weiterbildung von Lehrenden

Dr. Nicole Weicker

Universität Stuttgart
Universitätsstr. 38
D-70569 Stuttgart
weicker@informatik.uni-stuttgart.de

Abstract: Die in diesem Artikel beschriebene Weiterbildung für Informatiklehrende zielt auf die Vermittlung eines Gesamtbildes der Informatik, dass die Lehrenden dazu befähigen soll, ihren Unterricht an größeren Lernzielen und der Förderung von Schlüsselkompetenzen auszurichten. Ein „top-down"-Ansatz zur Förderung einer Reflexion über die Charakteristika der Informatik und den sich daraus ergebenden Folgerungen wird verknüpft mit einem „bottom-up"-Ansatz zur Integration der Lehrerfahrung der Informatiklehrenden vorgestellt.

1 Motivation

Viele Lehrende, die in Schulen Informatik unterrichten, haben selbst keine Informatikausbildung absolviert und lehren daher die Dinge, die sie sich im Selbststudium oder in fachlichen Weiterbildungen erarbeitet haben. Das Resultat ist ein sehr breites Spektrum hinsichtlich Inhalt und Niveau. Insbesondere fällt es vielen Informatiklehrenden schwer, Lernziele für ihren Unterricht zu formulieren. Zudem sind die Lehrpläne und Bildungsstandards häufig so umfangreich, dass die Lehrenden in der Verlegenheit sind, Schwerpunkte setzen zu müssen. Ohne ein Verständnis für die größeren Zusammenhänge der Informatik kann jedoch der Unterricht nur schwerlich auf größere Zielsetzungen oder einen Kompetenzerwerb ausgerichtet werden.

Die Schwierigkeiten, die sich aus dieser Situation ergeben, zeigen sich insbesondere darin, dass viele Schulabgänger, die Informatikkurse in der Schule belegten, ein falsches Bild von der Informatik haben. Diese Tatsache ist mit ein Grund für die hohen Abbruchquoten in den Informatikstudiengängen an Universitäten und Fachhochschulen.

Wenn Lehrende durch Weiterbildungsmaßnahmen gezielt für den Informatikunterricht geschult werden, stellt sich die Frage, was die wichtigsten Aspekte sind, die ihnen vermittelt werden sollen. Da die Weiterbildungskapazitäten bei den Lehrenden beschränkt sind, kann jede fachliche Weiterbildung zu bestimmten ausgewählten Themen wie beispielsweise Algorithmen und Datenstrukturen nur punktuelles Wissen vermitteln, das die Lehrenden auch nur punktuell weitergeben können.

Im Sinne des Allgemeinbildungscharakters der Sekundarstufe II sollten Schüler und Schülerinnen im Informatikunterricht zum einen lernen, was Informatik als Wissenschaft der

Informationsverarbeitung ist und was Informatik ausmacht. Zum anderen sollten sie im Informatikunterricht möglichst viele Fähigkeiten erwerben, die sie, unabhängig von den Informatikinhalten, in ihrem weiteren Leben brauchen können.

Eine gezielte Weiterbildung von Informatiklehrenden sollte daher in erster Linie eine informatik-didaktische Weiterbildung sein, die einen Schwerpunkt auf die Frage legt, was Informatik charakterisiert und die Schlüsselqualifikationen aufzeigt, die für Informatikabsolventen unverzichtbar sind. Die Förderung von derartigen Schlüsselqualifikationen im Rahmen der Vermittlung von Informatikinhalten kann dem Ziel der Allgemeinbildung am ehesten gerecht werden.

Das Ziel einer informatik-didaktischen Weiterbildung besteht darin, wesentliche Charakteristika der Informatik aufzuzeigen und die sich daraus ergebenden Auswirkungen deutlich zu machen. Aus diesen können fachliche, methodische und soziale Kompetenzen sowie Selbstkompetenzen abgeleitet werden, die ein Informatiker benötigt, um mit diesen Eigenschaften der Informatik erfolgreich umgehen zu können. Dieses Bild kann die Grundlage bilden, um Lernziele und dazu passende Lehrmethoden für einen Informatikunterricht im jeweiligen gegebenen Kontext zu entwickeln.

Bei dieser Form der „top-down"-Beschreibung der Informatik geht es weniger um einen Anspruch auf Wahrheit oder Vollständigkeit als vielmehr um Anstöße zur Diskussion und Reflexion.

2 Charakteristika der Informatik

Das Bild der Informatik ist vielfältig und bunt. Dies zeigt sich unter anderem darin, dass es bisher für diese noch relativ junge Wissenschaft keine einheitliche Definition gibt. Zwei seien hier beispielhaft aufgeführt:

- Definition aus dem Duden Informatik [CS01]:
  Informatik ist die Wissenschaft der systematischen Verarbeitung und Speicherung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Computern.
- Definition von Kristen Nygaard zitiert in [Co89]:
  Informatics is the science that has its domain information processes and related phenomena in artifacts, society and nature.

Eine wesentliche Übereinstimmung, die sich bei nahezu allen Definitionsansätzen für Informatik findet, besteht darin, dass Informatik sich als Wissenschaft mit der Verarbeitung und Speicherung von Informationen befasst. Information als solche ist ein immaterieller Bedeutungsinhalt einer Aussage, Benachrichtigung, Botschaft, o.ä., der an einen Informationsträger aus Materie oder Energie gebunden ist, wobei der Träger austauschbar ist.

Eine der Haupteigenschaften der Informatik, die sich daraus ergibt, ist die *Immaterialität ihres Hauptprodukts der Software*. Software wird zu ihrer Speicherung und Verarbeitung

sowie Informationen an Materie oder Energie gebunden, existiert jedoch nicht nur im Zusammenhang mit einem speziellen Träger.

Das eigentliche Spannungsfeld für den Informatiker entsteht daraus, die *Brücke zu schlagen* zwischen konkreten Anwendungen und dem abstrakten, immateriellen Konstrukt einer Software.

Ein weiteres wichtiges Charakteristikum der Informatik besteht darin, dass sie inzwischen mit nahezu allen anderen *Disziplinen zusammenarbeitet bzw. Dienstleistungen erbringt*.

Eine weitere informatik-spezifische Schwierigkeit ergibt sich daraus, dass die allermeisten *Projekte in der Informatik sehr groß* und unüberschaubar sind. Das bedeutet, dass eine Teamarbeit fast immer unumgänglich ist.

Die Hardwaresysteme, die Programmiersprachen und auch die auf dem Markt verfügbaren Tools entwickeln sich rasant. Das bedeutet für den Informatiker, dass sich sein *Handwerkzeug permanent verändert*.

Eine Besonderheit der Informatik stellt ihre direkte *gesellschaftliche Auswirkung* dar. Entwicklungen im Bereich der Informatik betreffen immer wieder nahezu jeden Menschen (z. B. Handy oder Internet).

3 Beispiel: Immaterialität von Software

Die Produkte der Informatik bestehen in erster Linie aus Software, die als Werkzeug zur Verarbeitung der immateriellen Größe „Information“ selbst immateriell ist. Aus dieser Eigenschaft der Immaterialität von Software ergeben sich eine Reihe von Folgerungen, die für Informatik charakteristisch sind (vgl. Abbildung 1). Diese Folgerungen sind ebenso wie die Liste von Charakteristika der Informatik in Abschnitt 2 subjektiv und erheben damit weder den Anspruch auf eine Korrektheit im Sinne einer Beweisbarkeit noch auf Vollständigkeit. Vielmehr sollen sie ein mögliches Gesamtbild der Informatik aufspannen, das zur Reflexion und Diskussion anregt.

3.1 Abstraktion von Software

Die Immaterialität der Software beinhaltet die Eigenschaft der Informatik, dass bestimmte Teile der Software nämlich gerade ihre Bedeutung abstrakt bleibt. Ebenso kann sie nur *schwer visualisiert* werden. Der Code selbst zeigt einem Betrachter lediglich die Details, ohne ein Verständnis für Zusammenhänge und Strukturen innerhalb der Software aufzuzeigen. Diese müssen ohne weitere Informationen mühsam aus dem häufig umfangreichen Code erarbeitet werden. Systemübersichten wie beispielsweise in UML-Diagrammen zeigen wiederum nur einen bestimmten Ausschnitt der Gesamtzusammenhänge. Damit zeigt sich, dass für Software wie für andere abstrakte Gebilde ein *Abgleich der Vorstellungen* verschiedener beteiligter Personen notwendig ist. Insbesondere ist es schwierig, in der Kommunikation über Software den *Informationsverlust*, der in jeder Kommunikation

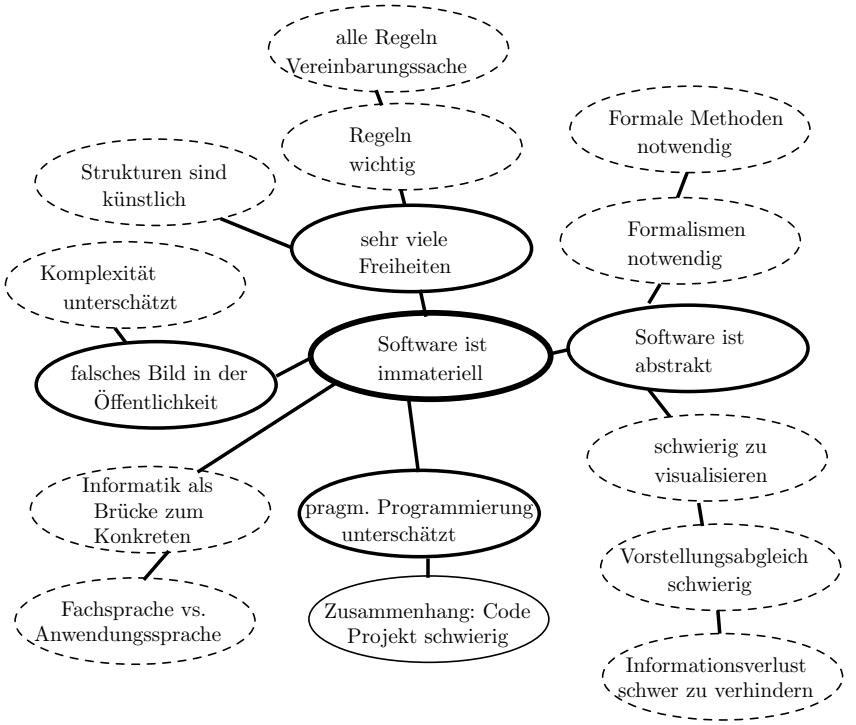

Abbildung 1: Eine zentrale Eigenschaft von Software ist ihre Immaterialität.

auftritt (Informationsverlusttreppe), ohne zusätzliche Hilfsmittel zu verringern. Ein notwendiges Hilfsmittel hierfür stellen die *Formalismen* dar, die helfen, Beschreibungen und Vereinbarungen eindeutig zu formulieren. Dabei ist es weniger wichtig, welche Formalismen es sind, als vielmehr, dass es Formalismen gibt, durch die eine Kommunikation eindeutig und unmißverständlich gehalten werden kann. Für die Verwendung von Formalismen stehen Informatikern eine Reihe von *formalen Methoden* zur Modellierung und zur Aufwandsabschätzung wie z. B. endliche Automaten, Turingmaschinen, Petrinetze oder Grammatiken zur Verfügung.

Die beschriebenen Eigenschaften von Software verlangen von einem Informatiker, wie in Abbildung 2 dargestellt, eine Reihe von Kompetenzen wie die Fähigkeit zur Abstraktion, eine Formalisierungskompetenz, die Fähigkeit, formale Methoden anwenden zu können und die Visualisierungskompetenz, um eigentlich nicht Visualisierbares zumindest in Ausschnitten doch sichtbar gestalten zu können. Andererseits ist es für einen Informatiker notwendig, die Idee der Sprache mit ihren Aspekten der Syntax und Semantik verinnerlicht zu haben. Erst durch ein derartiges Verständnis ist ein Informatiker in der Lage, Formalismen und formale Methoden wirklich sinnvoll in neuen Kontexten umzusetzen.

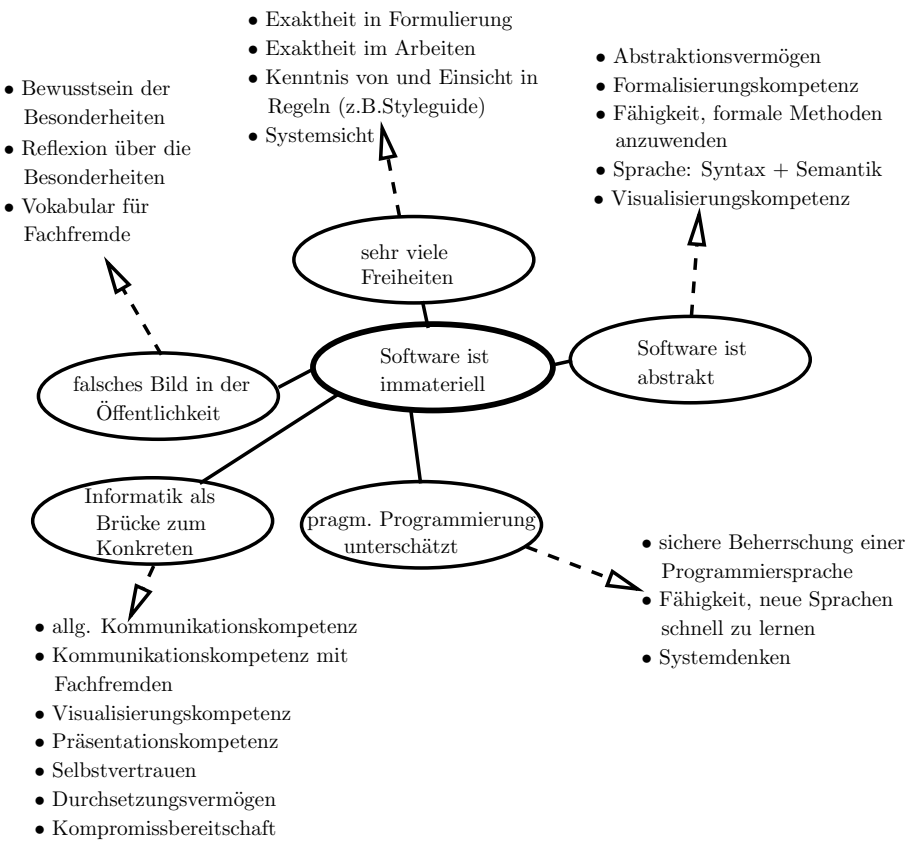

Abbildung 2: Aus der Immaterialität ergibt sich die Notwendigkeit für eine Reihe von fachlichen und überfachlichen Kompetenzen.

3.2 Große Freiheiten bei der Erstellung von Software

Eine weitere Folgerung der Immaterialität von Software ist die Tatsache, dass Informatiker eine sehr große Freiheit bei der Erstellung von Software besitzen (vgl. Abbildung 1). Software kann auf der Ebene der Maschinensprache (Assembler) ebenso entwickelt werden wie auf einer der modernen Sprachen wie Java, die bereits bestimmte Strukturen vorschreiben. Unstrukturierte Programmierung kann ebenso bzw. z. T. in noch weniger Code zum gleichen Ergebnis führen, wie strukturierte und gut kommentierte Programmierung. Um jedoch mit den entstandenen Softwareteilen umgehen zu können, sie verändern oder in andere Softwareumgebungen einpassen zu können, ist es notwendig, sich auf *Regeln* zu einigen. Dabei handelt es sich jedoch um Regeln, über die *Vereinbarungen* getroffen wurden und nicht um Gesetze, die immer gültig wären. Eine andere Folge der Tatsache, dass Software keinen Naturgesetzen unterliegt, besteht darin, dass Software auch keine natürliche Lokalität besitzt. Softwareteile, die sich untereinander möglichst nicht beeinflussen sollen,

sind durch *künstliche Strukturen* wie z. B. eine Modularisierung oder Hierarchisierung zu trennen.

Die Fähigkeiten, die einen Informatiker in die Lage versetzen, mit diesen Folgerungen der Immaterialität von Software umgehen zu können (vgl. Abbildung 2), sind auf der einen Seite exaktes Formulieren und exaktes Arbeiten sowie die Kenntnis um die gängigsten Regel wie beispielsweise Styleguides oder Prozessmodelle des Software Engineering und die Einsicht in die Notwendigkeit diese Regelwerke einzuhalten. Auf der anderen Seite benötigt ein Informatiker die Fähigkeit des Systemdenkens. Damit ist die Eigenschaft gemeint, sowohl selbst bei der Arbeit am Detail die systemischen Zusammenhänge der eigenen Arbeit in einem größeren Projekt im Blick behalten zu können.

3.3 Bild der Informatik in der Öffentlichkeit

Die *Vorstellung in der Gesellschaft*, was Informatik ist, wird wesentlich von den Auswirkungen der Informatik auf den Alltag des Einzelnen geprägt (vgl. Abbildung 1). Dabei spielen Handys, Computer, Internet, E-Commerce sowie gängige Textverarbeitungs- oder Kalkulationssysteme eine wesentliche Rolle. Die tatsächliche *Komplexität* der Software hinter diesen nach außen sichtbaren Leistungen der Informatik bleibt dem Nichtinformatiker dagegen verborgen. Gerade die Eigenschaft der Immaterialität der Software hat zur Folge, dass Informatik an der sichtbaren und oft simpel erscheinenden Benutzerschnittstelle gemessen wird. Die Auswirkungen dieser Fehleinschätzung sind, wie bereits in Abschnitt 1 erwähnt, am deutlichsten in der überdurchschnittlichen Anzahl an Studienanfängern in der Informatik zu sehen, die innerhalb weniger Semester ihr Studium abbrechen. Doch auch in Betrieben, in denen die Informatik als Dienstleistung benötigt wird, herrscht häufig Unverständnis für die Probleme, denen sich ein Informatiker zu stellen hat. Ein deutliches Kennzeichen hierfür zeigt sich in der oft sehr geringen Berücksichtigung für wichtige Phasen des Softwareentwicklungsprozesses wie z. B. der Anforderungsspezifikations- bzw. der Testphase.

Wenn sich ein Informatiker den Besonderheiten der Informatik bewusst ist, kann er seine Bedeutung und den Wert seiner Arbeit anders einschätzen und einordnen (vgl. Abbildung 2). Doch erst durch eine tiefere Reflexion über diese Besonderheiten und einem geeigneten Vokabular zur Vermittlung der speziellen Probleme der Informatik – auch an einen Nichtinformatiker – besteht eine Chance, in Betrieben mehr Geld und Zeit für eine gute Entwicklung und Wartung von Software durchzusetzen.

3.4 Informatik als Brücke zum Konkreten

Obwohl Software immateriell ist, wird sie zur Lösung von konkreten Aufgaben verwendet (vgl. Abbildung 1). Das Spannungsfeld des Informatikers besteht darin, eine *Brücke zwischen konkreten Anwendungen und dem abstrakten Konstrukt einer Software* zu schlagen. Dabei ist seine Aufgabe, reale Situationen geeignet zu analysieren und in Modellen

abzubilden, so dass die Anforderungen des Kunden und zusätzlich die der späteren Anwender der Software in den Modellen geeignet berücksichtigt werden. Ein wesentliches Problem, dem sich der Informatiker in dieser Phase seiner Arbeit zu stellen hat, liegt in der Unterschiedlichkeit der jeweiligen *Fachsprachen*. Informatik hat ebenso wie jede andere wissenschaftliche Disziplin seine eigene Fachsprache, wobei damit weniger die Programmiersprachen, sondern die informatikspezifische Ausdrucksweise in der Kommunikation über Software gemeint ist. Kunde und Anwender haben in aller Regel eine andere Fachsprache, so dass gerade die Anforderungsspezifikation sehr schwierig werden kann.

Der Brückenschlag der Informatik vom Konkreten zum Abstrakten verlangt vom Informatiker neben einer allgemeinen Kommunikationskompetenz die Fähigkeit, sich auch mit Fachfremden über seine Arbeit auszutauschen (vgl. Abbildung 2). Für den Informatiker gilt es, sich immer wieder in sein Gegenüber einzudenken (Empathie) und entstehenden Mißverständnissen möglichst schnell entgegen zu wirken. Zusätzlich sollte der Informatiker über geeignete Visualisierungs- und Präsentationskompetenzen sowie über ausreichend Selbstvertrauen, Durchsetzungsvermögen und Kompromissbereitschaft verfügen, um seine Lösungsansätze Kunden und Anwendern verdeutlichen zu können.

3.5 Wert der pragmatischen Programmierung

Auch für den angehenden Informatiker stellt die Immaterialität der Software ein Problem dar, da es häufig zu genügen scheint, wenn Zusammenhänge begriffen oder Lösungsansätze im Modell gefunden werden (vgl. Abbildung 1). Vor die Aufgabe gestellt, einen Tisch anzufertigen, ist jedem offensichtlich, dass es nicht genügt, verstanden zu haben, wie der Tisch glatt gehobelt wird. Es ist notwendig, selbst die Technik des Hobelns pragmatisch zu erlernen. Bei der Erstellung von Software dagegen wird gerade der *pragmatische Aspekt der Programmierung* häufig unter- und die eigenen Fähigkeiten in dieser Hinsicht werden entsprechend überschätzt. Die *Zusammenhänge* zwischen der Planung eines Projekts (Analyse, Spezifikation und Entwurf) und der tatsächlichen Umsetzung sind nicht einfach zu überblicken. So sind einerseits aus der Sicht der Planung die Probleme, die sich bei der Realisierung ergeben, häufig kaum abzuschätzen. Andererseits haben die Codierer Mühe, Vorstellungen und Nebenbedingungen, die nicht exakt festgehalten sind, umzusetzen, weil ihnen das Gesamtbild und die Bedeutung der Software im Projekt fehlen.

Konkret sollte im Unterricht der Informatik von Anfang an die Bedeutung der pragmatischen Programmierfähigkeit als ein wichtiger Aspekt der Informatik deutlich in den Vordergrund gestellt werden (vgl. Abbildung 2), ohne sich allerdings auf diesen wichtigen Bestandteil der Informatikkenntnisse zu beschränken. Ein Informatiker sollte in der Lage sein, eine Programmiersprache sicher zu beherrschen. Daneben ist es notwendig, dass die entsprechenden Grundkonzepte der Programmierung verstanden sind, so dass der Transfer von dieser Sprache in eine andere leicht und schnell möglich ist. Eine weitere, bereits angesprochene Kompetenz in diesem Zusammenhang, über die ein Informatiker verfügen sollte und die auch schon in der Schule geschult werden kann, ist die Fähigkeit des Systemdenkens.

3.6 Zusammenfassung der Kompetenzen bzgl. der Immaterialität

Die Qualifikationen, die sich durch die Immaterialität von Software ergeben, lassen sich untergliedern in fachliche, methodische, soziale und Selbstkompetenzen.

Die fachlichen Kompetenzen, die einen Informatiker befähigen, mit der Immaterialität von Software umzugehen, sind u. a. das Abstraktionsvermögen, die Formalisierungskompetenz, das Verständnis für das Konzept der Sprache mit der Unterscheidung in Syntax und Semantik sowie die Kenntnis von und Einsicht in Vereinbarungen und Regeln, wobei die beiden ersten Punkte hauptsächlich in der Theorie der Informatik geschult werden, während letzteres in Veranstaltungen zum Software Engineering vermittelt wird. Die Sprache als fundamentale Idee der Informatik [Sc93] zieht sich durch jede Form des Informatikunterrichts.

Die wichtigsten Methoden zur Beherrschung der Immaterialität von Software bilden die Exaktheit in Formulierung und Arbeit sowie die Fähigkeit, formale Methoden anwenden zu können. Ebenso wichtig ist die sichere Beherrschung einer Programmiersprache und die Kenntnis der programmiertechnischen Grundlagen und Konzepte, so dass weitere Programmiersprachen schnell erlernt werden können. Darüber hinaus ist die Fähigkeit, abstrakte Zusammenhänge visuell darstellen zu können, eine weitere wichtige Methode für den Umgang mit Immaterialität. Für diese Methodenkompetenzen kann im Informatikunterricht bereits eine Grundlage gelegt werden. Wesentlich ist dabei die Beschäftigung mit den Grundlagen der Informatik sowie das Erlernen einer Programmiersprache. Die Visualsierungskompetenz kann im Rahmen von Referaten gefördert werden.

Soziale Qualifikationen sind vor allem die Kommunikationskompetenz ,sowohl allgemein als auch speziell, im Umgang mit Fachfremden und die scheinbar widersprüchlichen Aspekte Kompromissbereitschaft und Durchsetzungsvermögen. Bis auf die fachliche Kommunikation mit Nichtinformatikern können diese Kompetenzen durch Projektarbeit gefördert werden. Für eine gezielte Förderung des Umgangs mit Fachfremden bieten sich Industriepraktika oder disziplinübergreifende Arbeiten an.

Die wesentlichen Aspekte der Selbstkompetenz, die ein Informatiker im Umgang mit der Immaterialität von Software erwerben sollte, ist ein Bewusstsein für und Reflexion über die Besonderheiten der Informatik sowie Selbstvertrauen und Präsentationskompetenz. Zusätzlich sind auch die Punkte der Systemsicht und des Systemdenkens der Selbstkompetenz zu zuordnen. Die Reflexionsebene über die Informatik kann ebenso wie die Systemsicht und -denken während des Studiums gefördert werden, indem zu verschiedenen Zeitabschnitten im Verlauf des Studiums immer wieder Denk- und Diskussionsaufgaben zu den Charakteristika der Informatik angeboten werden. Für die Ausbildung eines systemischen Denkansatzes kann es hilfreich sein, häufig wechselnde Positionen in Projekten vertreten zu müssen. Selbstvertrauen und Präsentationskompetenz können im Rahmen von Projekten und Seminaren geschult werden.

4 Ziele und methodisches Vorgehen

Das wichtigste Ziel besteht darin, die Teilnehmer der Weiterbildung dazu zu befähigen, über Informatik von einer höheren Warte aus zu reflektieren. Da es aus Zeitgründen nicht möglich ist, eine solche Grundlage auf detailliertem Wissen in den unterschiedlichen Teildisziplinen der Informatik aufzubauen, soll dies durch den Fokus auf die Besonderheiten der Informatik und deren Auswirkungen auf die Arbeit mit Informatik erreicht werden. Je größer der Weitblick der Lehrenden in Bezug auf Informatik ist, desto besser kann die Entwicklung von Schlüsselqualifikationen bei den Schülern und Schülerinnen gefördert werden.

Informatiklehrende, die an einer derartigen Weiterbildungsveranstaltung teilnehmen, sind in aller Regel keine unbeschriebenen Blätter, was die Informatik betrifft. Ein wesentliches didaktisches Prinzip ist deshalb die aktive Einbeziehung des unterschiedlichen Vorwissens der Teilnehmer. Ein weiteres wichtiges didaktisches Moment der Weiterbildung stellt die angeleitete Diskussion in Kleingruppen sowie moderierte Diskussionen im Plenum dar. Zur Unterstützung einer Reflexionsebene ist die aktive eigene Auseinandersetzung auf dieser Ebene unverzichtbar.

Um die Inhalte, die in der Weiterbildung vermittelt werden, an das bestehende Wissen der Teilnehmer anzuknüpfen und damit die Entstehung von „trägem Wissen“ zu verhindern, wird aufbauend auf Beispiele aus ihrem Lehralltag an den Schulen das Zusammenspiel von Lernzielen und Lehrmethoden im Informatikunterricht verdeutlicht. Die Teilnehmer können selbst beispielhaft Unterrichtsabschnitte vorstellen, die für den weiteren Verlauf der Weiterbildung einen Bezugspunkt für die Diskussion und Reflexion über Charakteristika und mögliche Lernziele der Informatik aus multiplen Perspektiven dienen. Durch diese konkrete problemorientierte Lernumgebung wird der Transfer des Gelernten in Alltagssituationen unterstützt [Re96, RM01, Se03].

Ausgehend von der Kombination der „top-down“-Sichtweise auf die Informatik und der „bottom-up“-Analyse von konkreten Beispielen aus dem Informatikunterricht können die Teilnehmer in ihrem Lehralltag gezielter Lernziele wie die Förderung von Abstraktionsfähigkeit, Formalisierungsfähigkeit oder die Entwicklung von Problemlösestrategien anstreben und die Schüler und Schülerinnen anhalten, sich über ihren eigenen Fortschritt in diesen Bereichen bewusst zu werden.

5 Konkrete Umsetzung und Ausblick

Die beschriebe Weiterbildung findet zur Zeit im Sommersemester 2005 an der Universität Stuttgart statt. Die Lehrer und Lehrerinnen kommen etwa zur Hälfte von allgemein bildenden bzw. von berufsbildenden Gymnasien. Knapp die Hälfte der Teilnehmenden (16 von 35) gab an, über eher weniger Informatikkenntnisse verfügen zu können. Nur sehr wenige Lehrer und Lehrerinnen (4 von 35) fühlten sich in der Informatik sattelfest. In wöchentlichen, zweistündigen Treffen werden neben dem beschriebenem „top-down“-Bild der Informatik, aus dem Lernziele abgeleitet werden, Lehrmethoden diskutiert, mit

denen diese Lernziele konkret umgesetzt werden können. Beispiele für die Lehrmethoden werden aus der Runde der Lehrer und Lehrerinnen vorgestellt und in ihrer Wirkung und Anwendbarkeit in den unterschiedlichen Kontexten der Teilnehmenden reflektiert.

Es ist geplant, die konkreten Beispiele der Teilnehmenden auf einer Webseite zu sammeln, so dass alle auf die benötigten Unterlagen und Programme Zugriff erhalten und diese selbst in ihrem Unterricht einsetzen können.

Neben der konkreten Umsetzung in Stuttgart ist das Ziel, das angedeutete Gesamtbild der Informatik anhand ihrer wesentlichen Charakteristika weiter auszubauen, um langfristig alle wichtigen Ideen, Konzeptionen und Kompetenzen, die in der Informatik vermittelt werden, zu erfassen. Je nach Zielgruppe können aus diesem Material unterschiedliche Ausschnitte für die Aus- bzw. Weiterbildung von Lehrenden oder für Diplomstudierende zusammengestellt werden, die zu einer integrierenden Sichtweise auf die jeweils relevanten Teile der Informatik verhelfen können.

Ein weiteres Ziel besteht darin, geeignete Lehrmethoden zu den Lernzielen, die sich aus den Charakteristika der Informatik ergeben, zusammenzutragen und durch Beispiele zu verdeutlichen.

Literaturverzeichnis

[Co89] W. Coy. Brauchen wir eine Theorie der Informatik? Informatik-Spektrum, 12:256-266, 1989.

[CS01] Volker Claus und Andreas Schwill. Duden Informatik. Dudenverlag, Mannheim, 3. Auflage, 2001.

[Re96] A. Renkl. Träges Wissen: Wenn Erlerntes nicht genutzt wird. Psychologische Rundschau, 47:78-92, 1996.

[RM01] Gabi Reinmann-Rothmeier und Hanz Mandl. Unterrichten und Lernumgebungen gestalten. In Andreas Krapp und BerndWeidenmann, Hrsg., Pädagogische Psychologie, Seiten 601-645. Beltz, Weinheim, 4. Auflage, 2001.

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

[Se03] N. M. Seel. Psychologie des Lernens. Reinhardt, München, 2003.