Material zum vierten Vorlesungstermin „Didaktik der Informatik“

Dr. Nicole Weicker
Universität Stuttgart
weicker@informatik.uni-stuttgart.de

13. Mai 2005

1 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 Claus and Schwill (2001):
  Informatik ist die Wissenschaft der systematischen Verarbeitung und Speicherung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Computern.

- Definition aus dem Studien- und Forschungsführer Informatik Brauer and Münch (1996):
  Informatik ist die (Ingenieur-)Wissenschaft von der theoretischen Analyse und Konzeption, der organisatorischen und technischen Gestaltung sowie der konkreten Realisierung von (komplexen) Systemen aus miteinander und mit ihrer Umwelt kommunizierenden (in gewissem Maß intelligenten und autonomen) Agenten oder Akteuren, die als Unterstützungssysteme für den Menschen in unsere Zivilisation eingebettet werden müssen – mit Agenten/Akteuren sind Software-Module, Maschinen oder roboterartige Geräte gemeint.

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.

Eine der Haupteigenschaften der Informatik, die sich daraus ergibt, ist die *Immaterialität ihres Hauptprodukts der Software*. Software ebenso wie Information als solche wird zu ihrer Speicherung und Verarbeitung an Materie oder Energie gebunden, existiert jedoch nicht nur im Zusammenhang mit einem speziellen Träger.

Das eigentlich 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).

1.1 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. Abb. 1). Diese Folgerungen sind ebenso wie die Liste von Charakteristika der Informatik in Abschnitt 1 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.

1.1.1 Abstraktion von Software

Aus der Immaterialität der Software ergibt sich direkt, dass Software *abstrakt* und nicht konkret oder fassbar ist. 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 schwierig ist. Insbesondere ist es schwierig, in der Kommunikation über Software den *Informationsverlust*, der in jeder Kommunikation 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,

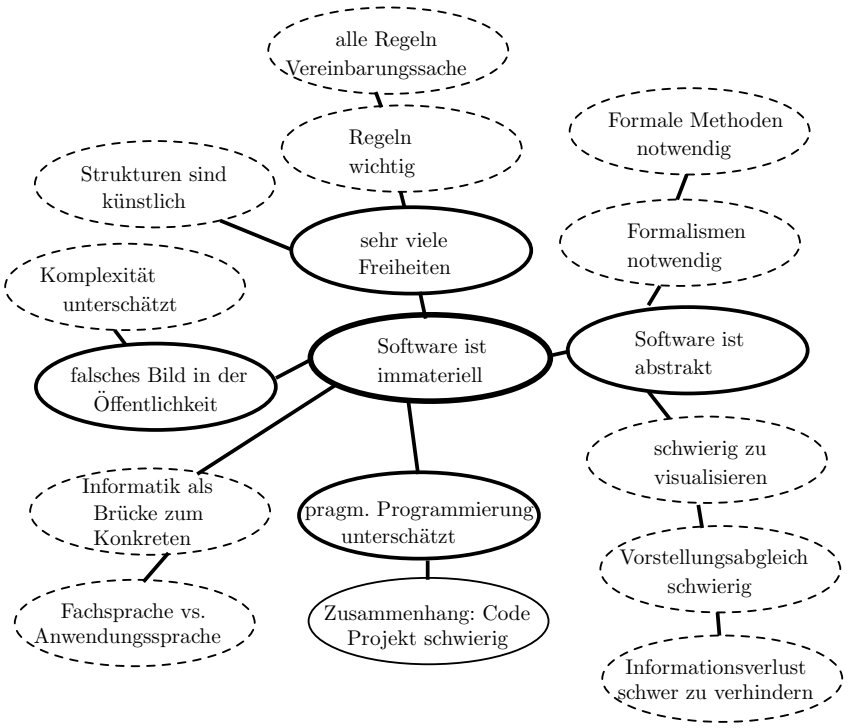

Abbildung 1: Eine zentrale Eigenschaft von Software ist ihre Immaterialität

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 Komplexitätsanalyse wie z.B. endliche Automaten, Turingmaschinen, Petrinetze oder Grammatiken zur Verfügung.

Die beschriebenen Eigenschaften von Software verlangen von einem Informatiker, wie in Abb. 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 zu mindest 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.

1.1.2 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. Abb. 1). Software kann auf der Ebene der Maschinensprache (Assembler) ebenso entwickelt werden wie auf einer der modernen Sprachen wie Java, die bereits bestimmte Struk-

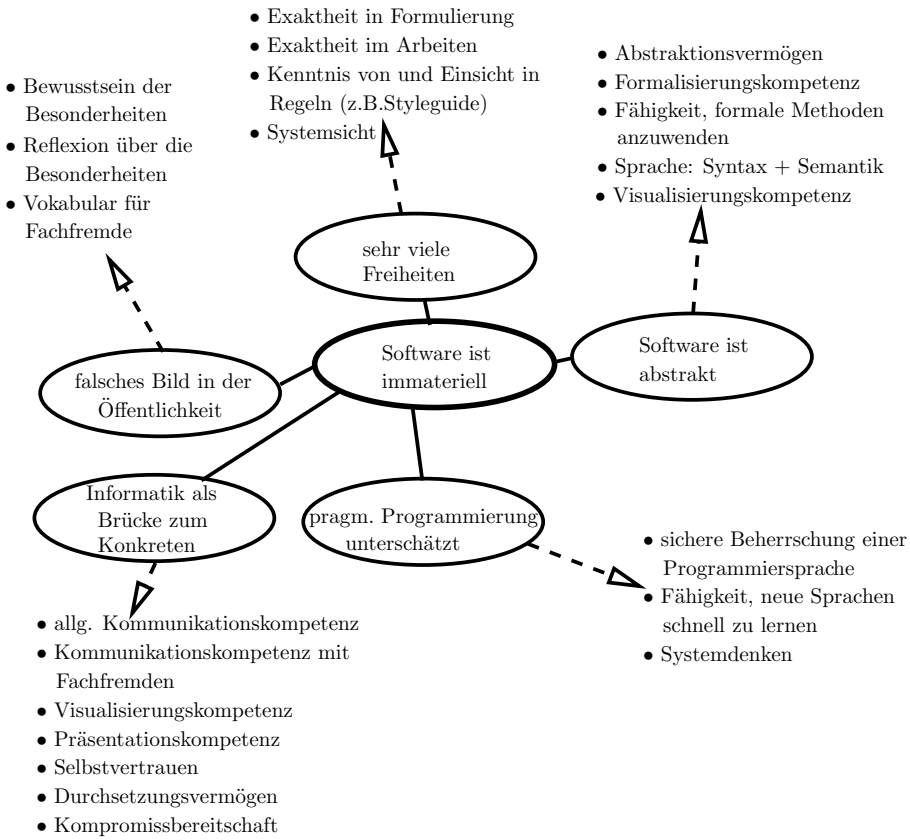

Abbildung 2: Aus der Immaterialität ergibt sich die Notwendigkeit für eine Reihe von fachlichen und überfachlichen Kompetenzen

turen 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. Abb. 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.

1.1.3 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. Abb. 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, 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 der Besonderheiten der Informatik bewusst ist, kann er seine Bedeutung und den Wert seiner Arbeit anders einschätzen und einordnen (vgl. Abb. 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.

1.1.4 Informatik als Brücke zum Konkreten

Obwohl Software immateriell ist, wird sie zur Lösung von konkreten Aufgaben verwendet (vgl. Abb. 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 Kundens 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. Abb. 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.

1.1.5 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. Abb. 1). Vor die Aufgabe gestellt, einen Tisch anzufertigen, ist jedem offensichtlich, das 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. Abb. 2), ohne allerdings sich 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 während auch schon in der Schule geschult werden kann, ist die Fähigkeit des Systemdenkens.

1.2 Informatik als Brücke zum Konkreten

Betrachtet man den Funktion der Informatik als Brücke zwischen der immateriellen Software zur Informationsverarbeitung auf der einen Seite und der konkreten, mitunter greifbaren Anwendung auf der anderen Seite, so ergeben sich daraus weitere wesentliche Charakteristika der Informatik sowie eine Reihe von Kompetenzen, mit denen ein Informatiker diesen Besonderheiten begegnen kann. Diese Eigenschaften der Informatik gelten selbstverständlich nicht für jeden Bereich dieser Disziplin.

Die theoretische Informatik beispielsweise beschäftigt sich weniger mit der konkreten Umsetzung als die anderen Bereiche der Informatik.

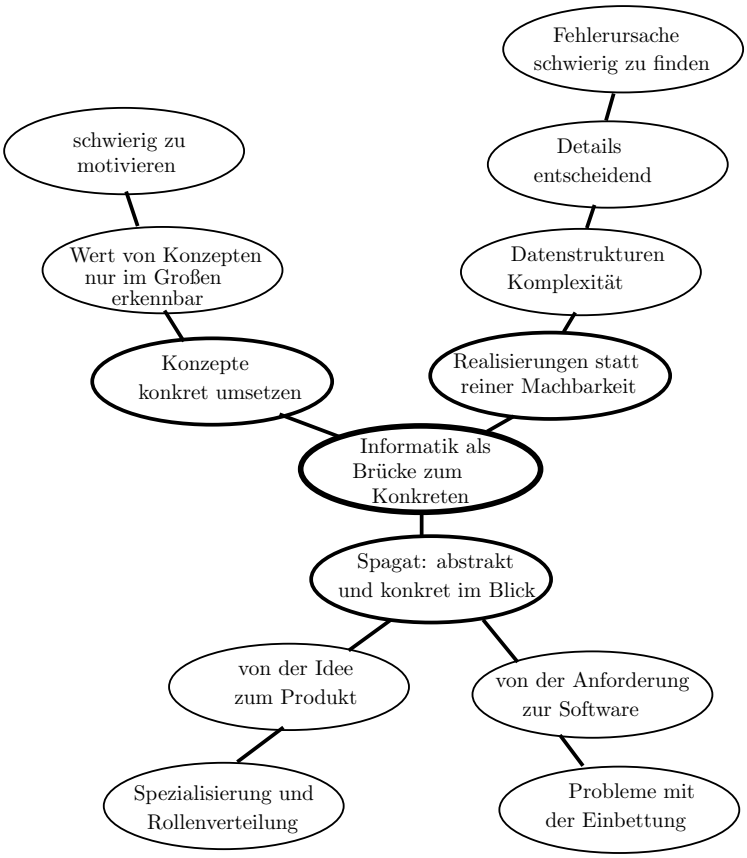

Abbildung 3: Informatiker haben die Brücke zwischen abstrakter Software und konkreten Anwendungen zu schlagen.

1.2.1 Realisierungen statt reiner Machbarkeit

Im Gegensatz zur Mathematik, die sich ebenfalls im Rahmen der Abstraktion bewegt, ist für die Informatik gefordert, gedanklich gefundene Lösungen tatsächlich zu realisieren, sprich in ein lauffähiges Programm umzusetzen, das den Anforderungen entspricht. Damit wird deutlich, dass es nicht genügt, die Existenz einer Lösung zu einem Problem zu zeigen. Das genaue Aussehen der Lösung ist vielmehr ebenso interessant. So spielen beispielsweise die Fragen, welche *Datenstrukturen* verwendet werden und welche *Komplexität* die angestrebten Lösungen besitzen, für die Qualität der zu erstellenden Software eine große Rolle. Viele *Details*, die für den Benutzer nicht sichtbar sind, wie beispielsweise interne Darstellungen von Daten oder die Systemarchitektur können entscheidende Auswirkungen auf die Brauchbarkeit der Software für die konkrete Anwendung besitzen, da von ihnen die Laufzeit oder die Erweiterbarkeit direkt betroffen sein können. Eine weitere Eigenschaft von Software liegt in

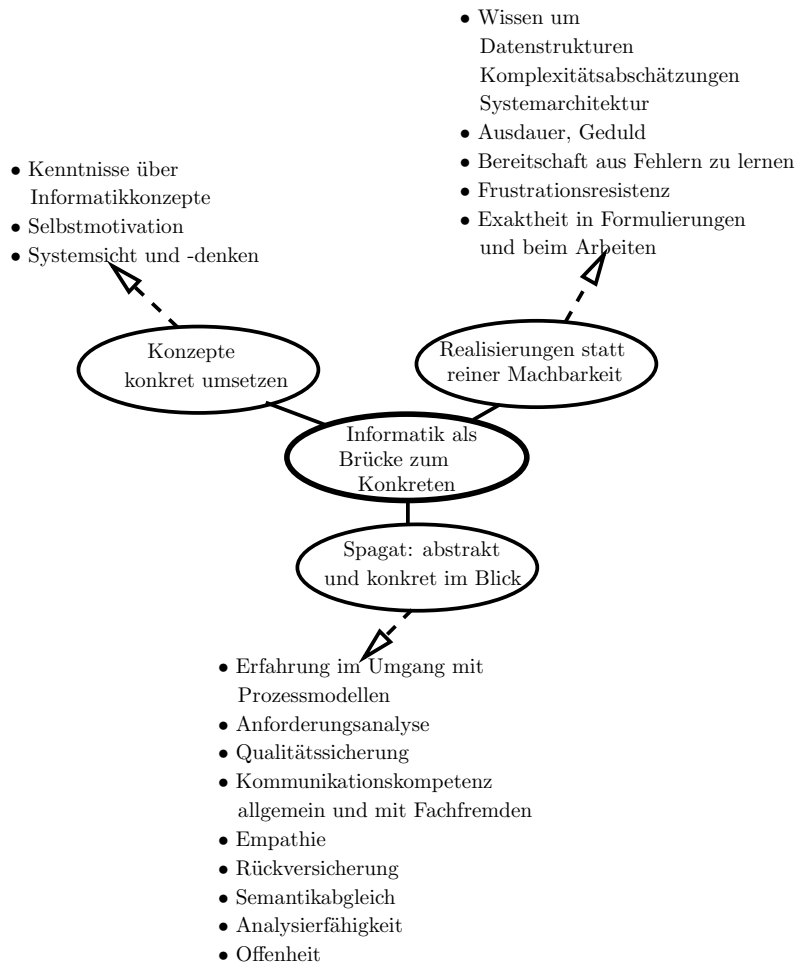

Abbildung 4: Durch die Rolle der Informatik als Brücke zwischen Abstraktem und Konkretem ergeben sich einige wünschenswerte Kompetenzen für Informatiker.

der Schwierigkeit, die Ursache für ein Fehlverhalten zu identifizieren. Dies gilt beim Implementieren, da Fehler sowohl vom eigenen Code als auch von eingeladenen Paketen, dem Entwicklungssystem oder dem Zusammenspiel mit den Programmteilen, die andere Entwickler erstellt haben, verursacht sein können. Das selbe Problem der *Fehleridentifizierung* besteht auf der Ebene der Einbettung einer Software in die Anwendungsumgebung.

Die wesentlichen Qualifikationen (siehe Abb. 4), um mit diesen Aspekten der Informatik umgehen zu können, sind neben dem fachlichen Wissen um Datenstrukturen, Komplexitätsabschätzungen, Systemarchitekturen etc. die Ausdauer und Geduld, Dinge auch unter z.T. unüberwindlich scheinende Widerstände zum Ende zu bringen. Ganz entscheidend ist die innere Einstellung, aus den Fehlern, die unvermeidlich auftauchen, lernen zu wollen, verbunden mit einer hohen Frustrationsresistenz. Zusätzlich gilt auch für diese informatikspezifische Schwierigkeit, dass exaktes Arbeiten unumgänglich ist.

1.2.2 Umsetzung von Konzepten

Abstrakte Informatikkonzepte wie beispielsweise die Algorithmisierung oder die strukturierte Zerlegung als fundamentale Ideen der Informatik oder der objektorientierte Entwurf gilt es für konkrete Anwendungen umzusetzen. Eine der wesentlichen Schwierigkeiten der Informatik liegt darin, dass der *Wert* vieler dieser abstrakten Konzepte wie z.B. der Objektorientierung anhand von kleinen Spielbeispielen nicht wirklich deutlich wird. Die überschaubaren Beispiele, die in Vorlesungen behandelt oder in Übungen vertieft werden können, lassen sich oft auch ohne den „Overhead“ einer objektorientierten Struktur elegant lösen. Die Notwendigkeit einer derartigen Struktur zeigt sich häufig erst an großen Aufgaben, wie sie die Anwendungen außerhalb der Informatik stellen. Bis ein angehender Informatiker eine solche Anwendung selbst miterlebt hat, bleibt die *Motivation* für eine Auseinandersetzung mit derartigen Informatikkonzepten schwierig und unkonkret.

Für die Umsetzung von Infomatikkonzepten benötigt ein Informatiker zunächst fundierte Kenntnisse über diese Konzepte, wie beispielsweise das Modellierungskonzept, die Objektorientierung oder die fundamentalen Ideen der Algorithmisierung bzw. der Modularisierung. Zusätzlich ist ein großes Maß an Selbstmotivation zumindestens während des Studiums sehr hilfreich. Eine weitere wichtige Eigenschaft, die einem Informatiker bei der Konkretisierung der abstrakten Konzepte zugute kommt, ist die Systemsicht sowie das Systemdenken, mit deren Hilfe es dem Informatiker gelingen kann, die Vorteile spezieller Konzepte bei größeren und unüberschaubaren Anwendungen zu erkennen.

1.2.3 Spagat zwischen Abstraktem und Konkretem

Eine wesentliche Besonderheit der Informatik liegt darin, in Projekten die beiden *entgegengesetzten Pole des Abstrakten und des Konkreten im Blick* behalten zu müssen. Dieser Spagat zeigt sich in zweierlei Hinsicht. Zum einen gilt es, die Anforderungen, die sich aus der *konkreten Anwendung* ergeben während der Entwicklung der *abstrakten Software* ständig so mit im Blick zu behalten, dass die fertige Software erfolgreich in der Anwendung eingesetzt werden kann. Die Schwierigkeiten, diesem Anspruch gerecht zu werden, zeigt sich in den häufig auftretenen *Problemen bei der Einbettung* einer erstellten Software. Auf der anderen Seite hat die Informatik den *Bogen von der Idee bis zum Produkt* abzudecken. Das bedeutet, dass Informatiker die Planung, Entwicklung ebenso wie die Umsetzung zu beherrschen haben. Da konkrete Anwendungsprojekte in aller Regel zu groß sind, um von einem Menschen in angemessener Zeit realisiert werden zu können, führt diese Eigenschaft zwangsläufig zu einer *Spezialisierung* einzelner Informatiker und damit zu einer *Festlegung von bestimmten Rollen.*

Die Kompetenz, die einen Informatiker in die Lage versetzt, mit diesem Spagat zwischen Abstraktem und Konkretem konstruktiv arbeiten zu können, besteht zum einen aus der Erfahrung im Umgang mit Prozessmodellen, um die beide Bögen von der Idee zum Produkt ebenso wie von der Anwendung zur Software aufspannen zu können.

Um Problemen bei der Einbettung der Software vorbeugen zu können, sollte ein Informatiker insbesondere die Anforderungsanalyse und die Qualitätssicherung sicher beherrschen. Dabei ist ein guter Kundenkontakt unverzichtbar, für den allgemeine Kommunikationskompetenz ebenso wie eine Kommunikationskompetenz im Umgang mit Fachfremden wichtig ist. Besonders wichtig ist dabei für einen Informatiker die Eigenschaft der Empathie, die es einem Informatiker ermöglicht, sich in die Rolle des Kunden bzw. des Anwenders zu versetzen, um aus dieser Perspektive, die Anforderungen an das gewünschte Produkt besser nachvollziehen zu können. Neben einer guten „Analysierfähgkeit“ einer Situation oder Aufgabe ist im Kontakt mit dem Kunden wichtig, dass der Informatiker offen ist, und nicht Gehörtes nur in bereits in seinem Kopf bestehende Bilder einfügt, und dass er immer wieder die Semantik von Begriffen durch Rückversicherungen abgleicht, um nicht mit gleichen Worten über Unterschiedliches zu reden.

Literatur

Rüdiger Baumann. *Didaktik der Informatik*. Klett, Stuttgart, 1996.

Wilfried Brauer and Siegfried Münch. *Studien- und Forschungsführer Informatik*. Springer Verlag, Berlin, 3 edition, 1996.

Volker Claus and Andreas Schwill. *Duden Informatik*. Dudenverlag, Mannheim, 3 edition, 2001.

Franz Eberle. *Didaktik der Informatik bzw. einer informationstechnologischen und kommunikationstechnologischen Bildung auf der Sekunda*. Sauerländer GmbH Verlag, Aarau, 1996.

Peter Hubwieser. *Didaktik der Informatik: Grundlagen, Konzepte, Beispiele*. Springer, Berlin, 2000.

Sigrid Schubert and Andreas Schwill. *Didaktik der Informatik*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2004.

Kompetenzen **Charakteristika**	fachlich	methodisch	sozial	selbst
Immaterialität	abstrahieren formalisieren Idee: Sprache Regeln	Exaktheit formale Methoden programmieren neue Sprachen lernen visualisieren	Kommunikation allg. und mit Fachfremden Kompromissbereitschaft Durchsetzungsvermögen	Bewusstsein für und Reflexion über Besonderheiten Selbstvertrauen präsentieren Systemsicht Systemdenken
Informatik als Brücke zum Konkreten	Algorithmisierung strukturierte Zerlegung Datenstrukturen Komplexität Systemarchitektur	Exaktheit Prozessmodelle Analysierfähgikeit Semantikabgleich Qualitätssicherung	Kommunikation allg. mit Fachfremden Empathie Teamfähigkeit	Selbstmotivation Systemsicht Systemdenken Ausdauer Geduld Umgang mit Fehlern Frustrations- resistenz Offenheit

Abbildung 5: Zusammenhang zwischen den Charakteristika der Informatik und möglichen Lernzielen