# Handout 3 zur informatik-didaktischen Fortbildung von Lehrenden

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

9. Juni 2005

## 1 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.

### 1.1 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 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,

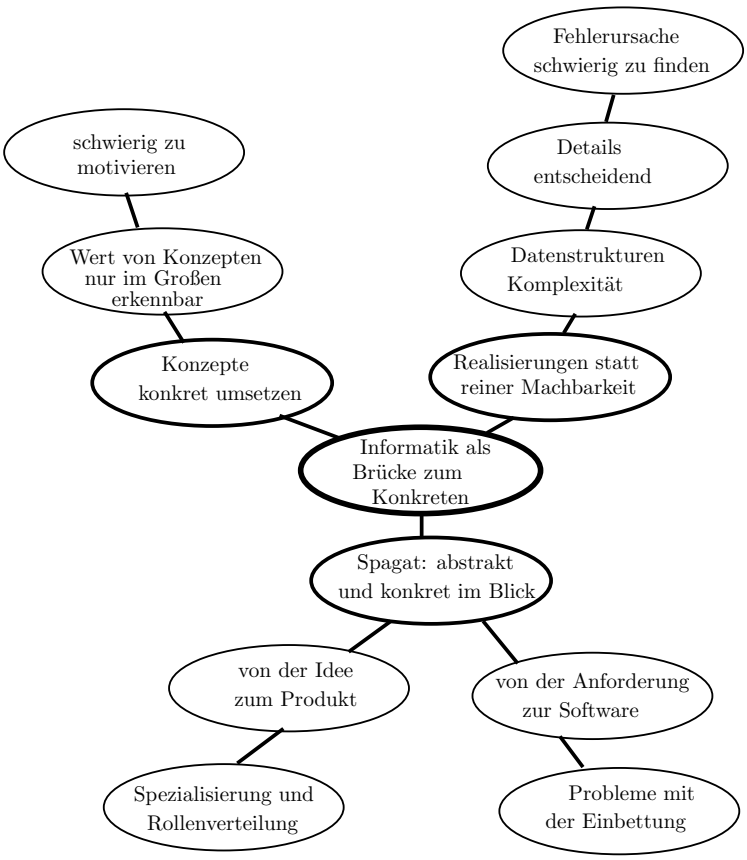


Abbildung 1: Informatiker haben die Brücke zwischen abstrakter Software und konkreten Anwendungen zu schlagen.

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. 2), 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 Umsetzung von Konzepten

*Abstrakte Informatikkonzepte* wie beispielsweise die Algorithmisierung oder die strukturierte Zerlegung als fundamentale Ideen der Informatik oder der objektorientierte

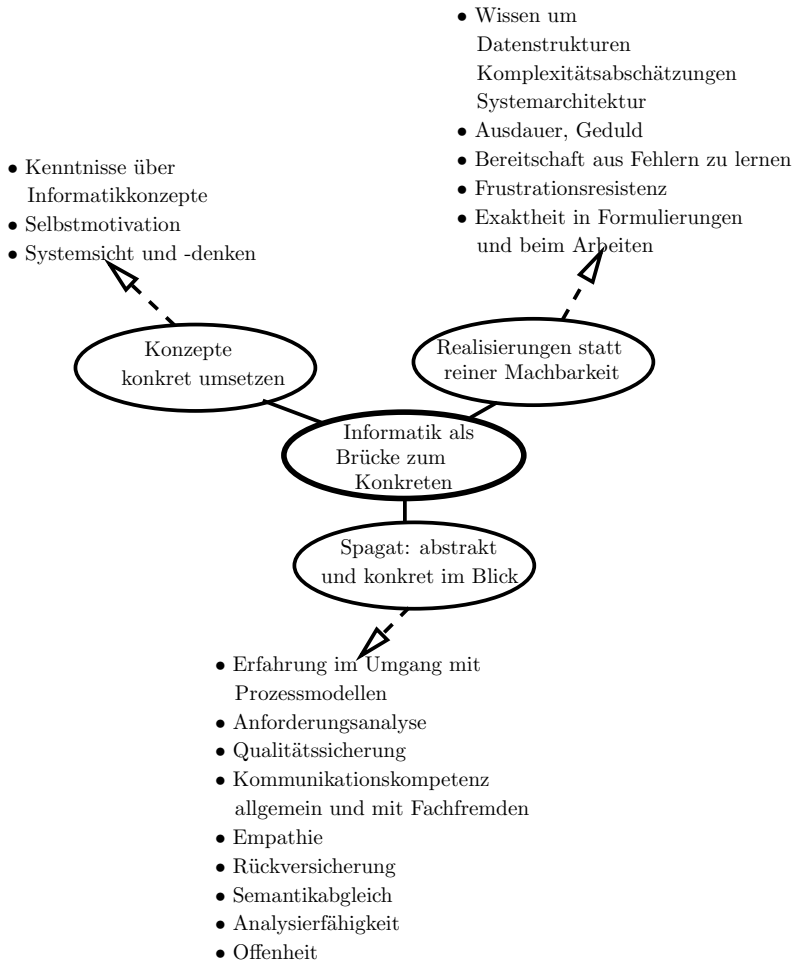


Abbildung 2: Durch die Rolle der Informatik als Brücke zwischen Abstraktem und Konkretem ergeben sich einige wünschenswerte Kompetenzen für Informatiker.

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.

## 1.3 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.

# 2 Lernziele und Lehrmethoden

## 2.1 Lernziele der Informatik

In Abbildung 3 sind die angestrebten Lernziele nach den vier Kompetenzbereichen aufgeschlüsselt zusammengetragen und als Ziele durchnummeriert (Z1 bis Z35). Die hervorgehobenen Ziele wurde von den TeilnehmerInnen der Fortbildung „Informatik-didaktischen Fortbildung für Lehrende“ für besonders wichtig im Informatikunterricht an den Schulen erachtet.

## 2.2 Lehrmethoden und Beispiele

**Beispiele aus der Praxis auf die Informatik übertragen** (M1) Mit dieser Methode werden die Ziele Z1, Z2, Z3, Z6, Z7, Z10, Z14, Z20 und Z28 angestrebt. Beispiele für die Umsetzung dieser Methode ist die *Konzeption und Realisierung von Datenbanken* sowie die *Umsetzung von mathematischen Datenstrukturen.*

**Simulation natürlicher Vorgänge aus der Biologie oder Physik** (M2) eine derartige Simulation kann theoretisch erfolgen und anschließend umgesetzt, sichtbar gemacht werden. Die Lernziele Z1, Z5, Z6, Z7, Z12, Z14, Z15 und Z18 können damit angestrebt werden.

**Spiel programmieren** (M3) Diese Methode eignet sich beispielsweise als Einstieg in Delphi. Beispiele sind „schwarzer Peter“, „Streichholzziehen“ und „Pentamino“ (Katamino). Die Ziele hier bei sind Z1, Z5, Z6, Z7, Z12 und Z14.

**Grenzen der Programmierung aufzeigen** (M4) Eine Behandlung des Halteproblems strebt folgende Ziele an: Z1, Z2, Z14, ZZ25, Z28 und Z35.

**Compilerbau** (M5) Mit einer Unterrichtseinheit zum Thema Compilerbau inklusive der praktischen Umsetzung zielt auf Z1, Z2, Z3, (Z7), Z8, Z10, Z13, Z14, Z15 und Z17.

**Homepage-Entwicklung** (M6) Wenn die Schüler und Schülerinnen eine Homepage entwickeln dürfen, zielt das auf Z2, Z4, Z11, Z14, Z15, Z27 und Z28. Wenn es sich dabei um eine Webseite wie beispielsweise die Schulhomepage handelt, werden zusätzlich Z13, Z19, Z20, Z21 und Z22 gefördert.

**Betriebsbesichtigung mit Führung** (M7) Der Besuch in einem Betrieb mit echten IT-Arbeitsplätzen kann die Ziele Z8, Z14, Z16, Z18, Z19, Z20, Z21, Z24, Z25 und Z28 unterstützen.

**Behandlung formaler Methoden der Informatik** (M8) Die Behandlung von

| **Kompetenzen**<br><br>**Charakteristika** | fachlich | methodisch | sozial | selbst |
|---|---|---|---|---|
| Immaterialität | **fachl. abstrahieren** Z1<br>formalisieren Z2<br>Idee: Sprache Z3<br>Regeln Z4 | **Exaktheit** Z10<br>formale Methoden Z11<br>**programmieren** Z12<br>**neue Sprachen lernen** Z13<br>**visualisieren** Z14 | Kommunikation allg. Z19<br>und mit Fachfremden Z20<br>Kompromissbereitschaft Z21<br>Durchsetzungsvermögen Z22 | Bewusstsein für<br>und Reflexion über Z25<br>Besonderheiten<br>Selbstvertrauen Z26<br>präsentieren Z27<br>**Systemsicht** Z28<br>**Systemdenken** |
| Informatik als<br>Brücke zum<br>Konkreten | **Algorithmisierung** Z5<br>strukturierte Z6<br>Zerlegung<br>**Datenstrukturen** Z7<br>Komplexität Z8<br>**Systemarchitektur** Z9 | Exaktheit<br>Prozessmodelle Z15<br>Analysierfähigkeit Z16<br>**Semantikabgleich** Z17<br>**Qualitätssicherung** Z18 | Kommunikation allg.<br>mit Fachfremden<br>Empathie Z23<br>**Teamfähigkeit** Z24 | Selbstmotivation Z29<br>Systemsicht<br>Systemdenken<br>**Ausdauer** Z30<br>**Geduld** Z31<br>**Umgang mit Fehlern** Z32<br>**Frustrations-**<br>**resistenz** Z33<br>Offenheit Z34<br>**Selbstkritik** Z35 |

Abbildung 3: Zusammenhang zwischen den Charakteristika der Informatik und möglichen Lernzielen

endlichen Automaten oder Petrinetzen kann auf der einen Seite durch die Modellierung und Formalisierung auch ohne konkrete Umsetzungen am Rechner viel zeigen. Andererseits können auch diese formalen Methoden konkret umgesetzt und visualisiert werden. Insgesamt werden durch diese Themen die Ziele Z1, Z2, Z6, Z10, Z11, Z14, Z15 und Z16 angestrebt.

# Literatur

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

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.