Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

Theorie der Informatik zwischen den Stühlen

Gegensätze in der Informatik durchmustern und füreinander fruchtbar machen

Dirk Siefkes
Technische Universität Berlin
Fakultät Elektrotechnik und Informatik

Bericht-Nr. 2007 – 21

ISSN 1436-9915

Theorie der Informatik zwischen den Stühlen

Gegensätze in der Informatik durchmustern und füreinander fruchtbar machen

Dirk Siefkes, Technische Universität Berlin, Fak. Elektrotechnik und Informatik

Zusammenfassung: Eine Theorie eines Gebiets ist eine Position außerhalb, die von innen getragen wird und so eine kundige Sicht auf Gebiet und Umgebung, auf Abgrenzungen und Beziehungen erlaubt. Eine allgemeine Theorie der Informatik kann uns helfen, die Entwicklung der Disziplin und insbesondere des eigenen Fachgebietes besser zu verstehen und zu beeinflussen. Ansätze dazu gibt es in der Informatik in jedem Fachgebiet, da wir immer – Praktiker wie Theoretiker, bewusst oder nicht – beim Arbeiten Theorie entwickeln und verwenden. Ein Gerüst von außen liefern z.B. Psychologie und Soziologie, Semiotik und Linguistik, Pädagogik und Philosophie und Geschichtswissenschaft. Die Theoretische Informatik wird durch eine allgemeine Theorie der Informatik nicht beeinträchtigt, sondern kann wertvolle Beiträge liefern. Ich arbeite auf eine solche Theorie der Informatik hin, indem ich die gegensätzlichen Denk- und Arbeitsmuster, die die Disziplin durchziehen und umringen, analysiere und zeige, wie sie füreinander fruchtbar sein können. Dabei baue ich auf der Theoriedebatte auf, die seit fast 20 Jahren geführt wird.

Die Informatik ist von Gegensätzen durchzogen und umringt, die sie prägen: Mensch und Computer, Wissenschaft und Technik, Entwickler und Nutzer, Theorie und Praxis, Theorie und Ethik – um nur einige zu nennen. Gegensätze können zu Fronten werden, Kämpfe auslösen, die allen schaden; dann hemmen sie die Entwicklung. Gegensätze können aber auch in Beziehung gesetzt werden; manchmal zeigt sich dann, dass sie sich ergänzen und sogar aufeinander angewiesen sind. Dann kann man die Probleme, die sie hervorrufen, von beiden Seiten her angehen und so die Entwicklung fördern. Solche Gegensätze nenne ich *komplementäre Paare* (Sie95): Auf den ersten Blick widersprechen sie sich, schließen sich aus; auf den zweiten brauchen sie sich, verstärken sich gegenseitig; und drittens schließlich führen sie zu Veränderung, wenn sie angenommen und genutzt werden. Dann entsteht nämlich ein *Spannungsfeld*, in dem die Beteiligten sich entwickeln. Komplementäre Paare können zur *Entwicklung* beitragen.

Dazu sucht oder schafft man zwischen den Polen des jeweiligen Spannungsfeldes eine vermittelnde Position, eine *Vermittlung*, die von beiden Seiten getragen wird und eine Sicht auf beide Seiten erlaubt. Eine solche Position außerhalb, von der aus Kundige auf einen Problembereich oder ein ganzes Gebiet schauen, heißt traditionellerweise *Theorie*. Eine Theorie eines Faches, die von Fachfremden kommt, nützt wenig, weil ihnen die Fachkenntnis fehlt. Eine Theorie, die von den Fachvertretern selber erstellt wird, taugt wenig, wenn sie die Bezüge nach außen nicht einbeziehen und so nicht den Abstand finden, die Sicht von außen.

Will man die Entwicklung der Informatik verstehen, beeinflussen, fördern, kann man also die Gegensätze nutzen, die ihr inhärent sind, statt sie zu ignorieren, zu versöhnen, zu bekämpfen oder zu beklagen. Die Vermittler – „Theoretiker“ – beschönigen die Gegensätze nicht, sondern klären sie auf, sodass Unterschiedliches und Gemeinsames sichtbar wird. In diesem Sinn bemühen wir uns – eine Gruppe von Informatikern, unterstützt von anderen – seit fast 20 Jahren um eine *Theorie der Informatik* (TdI 92ff).

Einen ähnlichen Ansatz verfolgt Cecile Crutzen (Cru00, 01, C&H01, 07, Crutzen und Hein in TdI02): „Oppositionen“, z.B. zwischen Entwicklern und Nutzern, „dekonstruieren“, indem man die Bewertung der Pole umkehrt und in dem dadurch möglichen Dialog neue Zugänge „konstruiert“; vgl. Abschnitte 2,3,5,7. Auch bei dem Pädagogen Werner Sesink spielt die Vermittlung zwischen Gegensätzen eine zentrale Rolle: Gute Lehrer vermitteln zwischen den Fähigkeiten der Schüler und den Angeboten und Anforderungen der Gesellschaft (Ses01). Aus dem Gegensatz zwischen Technik und Bildung macht er so eine Verwandtschaft, die er in (TdI03, Ses04) zu einer Analogie zwischen Pädagogik und Informatik ausbaut: Informatiker sollten zwischen Entwicklern und Nutzern vermitteln; vgl. Abschnitte 2,4,5,7.

Eine Theorie der Informatik gewinnen wir aus ihrer Entwicklung und benutzen sie, um diese zu verändern. Eine gute Theorie der Informatik kann die Entwicklung der Disziplin vorantreiben und die Richtung positiv beeinflussen. Eine schlechte kann die Entwicklung hemmen oder in eine falsche Richtung lenken, so dass sie sich negativ für die Menschen auswirkt. Entwicklung setzt Bekanntes fort, aber nicht immer gleich, sondern den sich ändernden Situationen angemessen. Entwicklung stellt also *Muster* dar: Folgen von ähnlichen Phänomenen, variierend gemäß den jeweils beteiligten Menschen und Umgebungen. Eine Theorie der Informatik sollte also nach Mustern in der Informatik suchen, um Gegensätze zu hinterfragen, Probleme und Ansätze zu ihrer Lösung zu finden und so zu helfen, die Disziplin positiv weiterzuentwickeln (Sie05ff).

Die Arbeit an einer Theorie der Informatik begann mit einem BMFT-Diskursprojekt von Wolfgang Coy 1988-92 (vgl. TdI92 und andere Publikationen der Projektträger). Weitergeführt haben sie Frieder Nake, Arno Rolf und ich mit drei Arbeitstagungen 2001-03 (TdI01ff). Immer hat dabei die Spannung zwischen der Wissenschaft Informatik und ihren Anwendungen eine wichtige Rolle gespielt – wie in der Informatik selbst. In (TdI92) beziehen sich die Abschnitte ‚Grundlagen…’ und ‚Kultur…’ auf die Wissenschaft, die Abschnitte ‚Arbeit…’ und ‚Ethik…’ auf die Anwendungen. Im Titel des Buches wird der Gegensatz als verschiedene „Sichtweisen“ verharmlost und nicht thematisiert, geschweige denn bearbeitet. Auf der Theorietagung 2001 begann die Diskussion darum; dabei standen sich Teilnehmer der beiden Arbeitsgruppen „Kultur…“ und „Semiotik...“ einerseits und „Gestaltung…“ andererseits gegenüber (TdI01). Auf den Tagungen 2002 und 2003 prägte der Gegensatz Struktur und Verlauf, mit „Theorie der Anwendungen…“ als einer von drei bzw. zwei Arbeitsgruppen (TdI02, 03). Der Hintergrund ist, dass die Spannung sich speist aus dem Gegensatz zwischen Entwicklern und Nutzern, der den Kern der Probleme beim Einsatz von IT-Systemen bildet (s. Abschnitt 5).

Tatsächlich bilden Informatik und ihre Anwendungen ein komplementäres Paar: Jede Seite lebt von der anderen – so stark wie in wenigen Wissenschaften, den Gegensatz gibt es ja immer –, auch wenn das oft nicht deutlich ist. Auch sind nicht wenige Beteiligte auf beiden Seiten aktiv. Ein Ziel einer Theorie der Informatik sollte also die Vermittlung zwischen der Wissenschaft und ihren Anwendungen sein – nicht das einzige, es gibt genug andere Gegensätze, aber ein zentrales. Explizit aufgegriffen haben das Ziel Arno Rolf und ich 2006 auf der Tagung „Mensch-Maschine-Kommunikation“ mit der Arbeitsgruppe „IT-Gestaltung im Labyrinth der Organisation“ (MMK06). Wie schon auf den Theorietagungen wurden vielversprechende Ansätze deutlich, aber das Ziel wurde nicht erreicht. Eine Theorie der Informatik ist in dieser wichtigen Frage erst im Werden. Mehr dazu in Abschnitt 5.

Ein eng verwandtes Spannungsfeld, innerhalb der Wissenschaft selbst, ist das zwischen Kerninformatik – mit Theoretischer und Technischer Informatik, Softwaretechnik und anderen Gebieten – und Angewandter oder Praktischer Informatik – von Systemanalyse bis zu ‚Informatik und Gesellschaft’. Bezeichnungen, Einteilungen, Vorstellungen variieren, die Unterschiede verwischen sich oft. Aber der Gegensatz ist in den Köpfen vorhanden und daher wirksam: In der Kerninformatik geht es vor allem um die Informationstechnik (IT) selbst, in der Angewandten um deren praktische Verwendung. Und die Spannung zwischen den Gegenständen erzeugt ein Abbild in der Wissenschaft, das seinerseits ein komplementäres Paar bildet und so für Bewegung sorgt.

Da Computer mit Zeichen losgelöst von ihrer Bedeutung, also formal, arbeiten, geht der Weg vom Menschen zur Maschine immer über Formalismen (s. Abschnitt 2). Deswegen verstehen sich Vertreter der Theoretischen Informatik gern als Mittler zwischen Kern- und Angewandter Informatik. Tatsächlich ist die Theoretische Informatik, soweit sie sich um solche Formalismen bemüht, ein unerlässlicher Bestandteil der Informatik, auch wenn das von den Praktikern manchmal bestritten wird. Umgekehrt lebt die Theoretische Informatik von diesem Verständnis, als rein mathematisches Gebiet wäre sie in der Informatik ein Fremdkörper, auch wenn das ihre Vertreter manchmal anders sehen. Die Theoretische Informatik könnte z.B. mit der Technischen ein komplementäres Paar bilden; Vermittlungen sehe ich aber nicht. Am stärksten sind wohl die Beziehungen zur Softwaretechnik und verwandten Gebieten; aber komplementäre Paare kenne ich auch hier nicht. Eine allgemeine Theorie der Informatik steht also nicht in Konkurrenz zur Theoretischen Informatik, wie das von Fachvertretern oft aufgefasst wird (Sie07b). Sie könnte aber der Theoretischen Informatik neue Anstöße zur Entwicklung geben, wenn sich Theoretische Informatiker an der Theoriearbeit beteiligen (s. Abschnitt 8). Ich selber habe das versucht (vgl. Sie90), bin aber darüber ganz zur Theorie der Informatik übergewechselt (Sie92). Umgekehrt kann so die kritische Beschäftigung mit der Theoreti-

schen Informatik zur Arbeit an einer allgemeineren Theorie führen. Auch bei Anderen hat die gemeinsame Arbeit an einer Theorie der Informatik solche Wurzeln.

Informatiker, die an einer *Theorie der Informatik* arbeiten, sitzen also immer *zwischen den Stühlen*. Als Informatiker sind sie in einem Fachgebiet beheimatet, als Theoretiker suchen sie nach einer Position außerhalb – angezogen von einem anderen Fachgebiet, einer anderen Disziplin oder gar von einem Bereich außerhalb der Wissenschaft, mit dem sie hoffen in Austausch zu treten, um die eigene Arbeit und die Wechselwirkungen besser verstehen und bewerten zu können. Theoriearbeit ist also nichtwissenschaftlich – nicht un- oder gar anti-wissenschaftlich, denn sie geht aus wissenschaftlicher Arbeit hervor, aber auch nicht wissenschaftlich, denn Wissenschaft wird herkömmlicherweise in Disziplinen und dort in Fachgebieten betrieben, nicht als Vermittlung zwischen komplementären Positionen. Darin liegt der Reiz und die Schwierigkeit der Arbeit an einer Theorie der Informatik: den Anspruch gegenüber Beteiligten und „Betroffenen“ (und damit gegenüber sich selbst) zu legitimieren und verständlich zu machen.

Eine solche vermittelnde Position zwischen den Stühlen suche ich in diesem Aufsatz. Und so sieht die Suche aus: In Abschnitt 1 diskutiere ich *Muster und Entwicklung* und zeige, wie beide zusammenhängen; dabei spielt *Ähnlichkeit* eine wichtige Rolle. Damit kann ich in Abschnitt 2 den berüchtigten Gegensatz zwischen *Mensch und Computer* aufdröseln und verständlich machen, wie lebendige Menschen mit starrer IT umgehen. Dafür ist der Begriff *Hybridisierung* wichtig. In Abschnitt 3 betrachte ich drei Gegensatzpaare, die *menschliche Kultur* ausmachen: Regeln und Verhalten (Organisation), Zeichen und Bedeutung (Kommunikation), Schemata und Phänomene (Menschen). In Abschnitt 4 gehe ich den ehrwürdigen Gegensatz von *Individuum und Gesellschaft* mit diesen Mitteln an, um besser zu verstehen, wie Kulturen sich gegenseitig beeinflussen und sich dabei entwickeln. Diese *kulturelle Theorie* wende ich im Hauptteil der Arbeit auf die Informatik an und gewinne so eine Theorie der Informatik: In Abschnitt 5 analysiere ich, was geschieht, wenn IT-Systeme in geregelte Umgebungen eingeführt werden (*Destruktion, Instruktion, Konstruktion*). In Abschnitt 6 gehe ich der *Entwicklung der Wissenschaft Informatik* nach: Ich benenne komplementäre Paare, die sie bei ihrer Entstehung in der BRD geprägt haben, und zeige an der Entwicklung der Programmierung, wie die Prägung bis heute wirksam geblieben ist. Informatik ist danach keine wissenschaftliche Disziplin in irgendeinem geläufigen Sinn; ich charakterisiere sie deswegen in Abschnitt 7 als *kulturelle Entwicklung*. Daraus ergibt sich eine *Aufgabe der Informatik*, die den historisch gegebenen technisch-formalen Anteil mit dem aktuell dringlichen psychisch-sozial-politischen zusammenbringt. Mit dem Blick auf diese Aufgabe diskutiere ich in Ab-

schnitt 8 einige Gebiete der Informatik und ihrer Umgebung – alte, neue, entstehende und solche, die sich nicht als Gebiet fassen lassen – sowie ihre Verhältnisse zueinander. In Abschnitt 9 füge ich das alles zu einer *Theorie der Informatik* zusammen, die aus den klassischen Spannungen zwischen Theorie und Praxis, Ethik oder Politik Folgerungen für Forschung und Studium zieht.

Der Aufsatz spiegelt meine eigene Beteiligung an der Theoriearbeit wider. Im ersten Teil (1-5) lehnt er sich eng an die gleichzeitig entstandene Publikation (Sie07a) an, im zweiten Teil (6-9) schöpft er aus der Dokumentation der gemeinsamen Arbeit, insbesondere aus dem Bericht (TIK02) zu meiner Arbeitsgruppe in (TdI02) und den Unterlagen (MMK06). Ich habe versucht, die verschiedenen Aspekte der Theoriearbeit seit 2001 zu erfassen, will und kann aber kein umfassendes Bild zeichnen. Ich danke allen, die so wissentlich oder unwissentlich an dem Aufsatz beteiligt sind, insbesondere meinen Theoriemitarbeitern Frieder Nake und Arno Rolf (TdI01ff), unserem Vorarbeiter Wolfgang Coy (TdI92) sowie Jochen Ludewig, dessen Stimme in dem erwähnten Bericht unüberhörbar ist.

1. Muster und Entwicklung

Muster spielen in der Informatik als *Entwurfsmuster* eine wichtige Rolle (Gam95): Gute Lösungen für Softwareprobleme werden gesammelt und nach einem festen Schema beschrieben; die Beschreibungen dienen als Vorbilder in ähnlichen Fällen. Da jede Situation anders ist, werden die Vorbilder dabei passend abgewandelt – anders als bei Softwarebibliotheken, aus denen man fertige Programme auswählt. Damit haben wir eine erste Definition: *Muster* sind Wiederholungen in Abwandlung, angeregt von und angepasst an die jeweilige Situation (Bat72,79, Ale77,79, R&Z96, Sie05a,b). Stoff- und Tapetenmuster, an die wir bei dem Wort zuerst denken mögen, sind extreme Formen: Sie werden heute maschinell gefertigt, wiederholen daher ein Motiv immer gleich. Ich nenne solche Muster *starr* oder *gleichförmig*. Lebendiger sind handgefertigte Stücke, die ihren Reiz durch kleine Abwandlungen im Motiv erhalten. Arabische Teppiche mussten solche „Fehler“ haben, da nur Allah perfekte Lösungen schafft. Auch beim Abschreiben von Handschriften waren Abwandlungen üblich, die den Schreiber oder die Bedeutung der Stelle erkennen ließen. Menschliches Verhalten – „typische“ Bewegungen oder Gesten, Handlungen oder Ausdrücke, Reaktionen im Denken und Fühlen – bildet in der Regel lebendige Muster; starre Verhaltensmuster empfinden wir als krankhaft. Ein Extrem in die andere Richtung sind *chaotische* Muster, bei denen das Phänomen sich beliebig ändert, ein Muster eigentlich nicht erkennbar ist. Wir nennen Menschen chaotisch, wenn ihre Verhaltensmuster es sind. Muster, die nicht so extrem sind, nenne ich *lebendig* – in Anlehnung an Brödner, Seim, Wohland in (TdI02ff).

Die Entwurfsmuster von Gamma et al. sind also selbst keine Muster. Die kodifizierten Beschreibungen der vorbildlichen Lösungen erzeugen Muster, nämlich die den Situationen angepassten Nachbildungen. So wird jedes Muster von einem *Generator* erzeugt, der oft selber als Muster bezeichnet wird. Diese Identifizierung findet sich schon bei dem Architekten Christopher Alexander, der vorbildliche Lösungen für Wohn- und Siedlungssituationen gesammelt und in einheitlicher Form beschrieben hat, damit sie als Vorbilder für lebendige Architektur dienen (Ale77, 79). Er nennt die Lösungen ‚Muster', und von ihm sind Idee und Bezeichnung in die Softwaretechnik übernommen worden.

Umgekehrt entstehen die Vorbilder selber aus Mustern: Wir nehmen etwas wiederholt wahr, nie dasselbe, aber ähnlich, angeregt von und angepasst an gewisse Situationen – bis wir es als Phänomen identifizieren, ihm einen Namen geben und damit begrifflich fassen und einordnen. Jedes Muster lässt sich also auf einen *Kern* zurückführen, das Gemeinsame der immer ähnlich wiederkehrenden Phänomene. Kern und Generator sind dasselbe, aus verschiedener Sicht. Im allgemeinen sind die beiden Phasen des Wahrnehmens und des Erzeugens, des Abbild- und Vorbildseins nicht getrennt. Während wir das Muster entdecken und nach seinem Kern suchen, erzeugt der Kern im Entstehen seinerseits das Muster, wird zum Generator. Muster dienen daher für den Kybernetiker Gregory Bateson dem Erzeugen und Übertragen von *Information* (Bat72, 79): Lebewesen jeder Art *entwickeln sich* in Schritten, geradlinig oder spiralig, räumlich wie zeitlich, ontogenetisch wie phylogenetisch. „Gewusstes" oder „Geleistetes" wird von Segment zu Segment, von Stufe zu Stufe, von Generation zu Generation „in kodierter Form" transportiert und jeweils entsprechend angepasst. Ebenso wird Information über Grenzen beliebiger Art hinweg weitergegeben und erzeugt, z.B. bei der Wahrnehmung von „draußen" nach „drinnen", beim Handeln in umgekehrter Richtung (wobei die beiden Vorgänge selten getrennt ablaufen). Sich entwickeln heißt, „Gekonntes" weitertragen, Information aufnehmen und weitergeben. *Entwicklung* ist Musterbildung.

Nur bei der bewussten Herstellung trennen wir die beiden Phasen, erstellen ein *Modell*, von dem wir dann Abbilder herstellen. Auf diese Weise kondensieren wir Modelle jeder Art – gedanklich oder gegenständlich, mathematisch, informatisch, technisch, architektonisch, literarisch, philosophisch – aus wiederholten Beobachtungen, also als Abbilder, um sie als Vorbilder in weiteren Tätigkeiten zu verwenden. Modelle sind Kerne und Generatoren von Mustern im jeweiligen Bereich.

Lebendige Muster verändern sich oder bilden sich neu, während wir sie beobachten oder erzeugen; langsam verändert sich so der Generator bzw. Kern, während wir ihn verwenden oder entdecken; und gleichzeitig verändert sich dabei unser Blick auf den Bereich und der Bereich selbst. Muster zeigen und tragen so Entwicklung. Wenn sie starr oder chaotisch wer-

den, können sie Information nicht mehr aufnehmen und weitergeben, die Entwicklung stagniert bzw. zerfällt. Wenn sie sich mit anderen Mustern überlagern, ändert die Entwicklung ihren Verlauf oder ihren Wert.

Die *Kultur* einer menschlichen Gemeinschaft besteht aus den Mustern des Denkens und Fühlens, des Wahrnehmens und Verhaltens und der Beziehungen ihrer Mitglieder. Wenn wir eine Kultur durch ihre religiösen und politischen, ihre sozialen und individuellen Muster beschreiben, halten wir ihre Entwicklung in Händen. Die „Kultur" einer Gruppe – wann was wie und wobei getan und gedacht, gefühlt und geredet wird oder nicht, gemeinsam oder einzeln, schwatzend oder schweigend, fröhlich oder mürrisch – macht aus, was die Gruppe leistet und leisten kann, was sie „ist" und wohin sie sich entwickelt. Die „Arbeitskultur" einer Firma – das Verhältnis der Mitarbeiter zu ihrer Arbeit – ändert sich mit dem Verhältnis der Mitarbeiter zueinander. So entwickelt sich die Firma in ihren Arbeits- und Mitarbeitsmustern. Die „Streitkultur" einer Gesellschaft – Offenheit füreinander oder Verhärtung der Fronten, Verachtung oder Bewunderung – kann entscheiden, ob und wohin es mit der Gesellschaft weitergeht. In Kultur ist Entwicklung inhärent.

Ähnlichkeiten erkennen scheint eine grundlegende Fähigkeit alles Lebendigen zu sein – von Pflanzen und niederen Tieren, die auf ähnliche Reize ähnlich reagieren, bis zu den Menschen, die Ähnliches körperlich und darüber hinaus geistig einordnen und hervorbringen können. Ob John von Neumann das im Kopf hatte, als er auf der ersten Kybernetikkonferenz 1948 diskutierte, welche Fähigkeiten des menschlichen Gehirns man mit Automaten nachbilden könne? Als letzten Test wählt er ‚Ähnlichkeit' und stellt schließlich – resigniert oder augenzwinkernd? – fest: „Die kürzeste Beschreibung eines solchen Netzwerks könnte sehr wohl das Gehirn selber sein." (SGI97ff) Netzwerke sind Darstellungen von Automaten in einem logischen Formalismus, Nervenzellen sind für von Neumann logische Schaltungen. In heutige Technik und Ausdrucksweise übertragen hieße das: Menschliches Denken kann man nur mit dem Gehirn – gedanklich –, nicht mit Computern – technisch – nachbilden. Darüber kann man streiten. Aber unterschiedliche Denk- und Arbeitsmuster im Umgang mit Technischem und Geistigem aufspüren scheint mir ein gangbarer Weg, um zwischen Verfechtern und Anfechtern der harten KI-These zu vermitteln; mehr dazu in Abschnitt 2 und 8.

Schon für Aristoteles war *Mimesis*, Nachbildung, ein zentraler Begriff. Ob in der Poesie oder auf der Theaterbühne, Menschen erzeugen einen Gegenstand, indem sie ihn darstellen, wobei Darsteller und Dargestelltes sich verändern, indem sie einander näher kommen, sich „anschmiegen", sich anähneln. Emil Auerbach arbeitet in seinem Buch „Mimesis" (Aue46) auf diese Weise „Dargestellte Wirklichkeit in der abendländischen Literatur" heraus: Darstel-

lungen sind wirklich, soweit sie wirken, und vermitteln so zwischen einer Kultur und ihrer Umgebung. Ähnlich versteht Anthony Giddens (Gid84) das Verhältnis zwischen den Strukturen einer „Gesellschaft“ und den Handlungen ihrer Mitglieder als Wechselwirkung, und genauso analysiert Günter Ortmann in (Ort03) die Entstehung, Befolgung und Verletzung von Regeln in einer Organisation; vgl. Abschnitt 3. Menschliche Gemeinschaften jeder Art entwickeln sich in der Entstehung, Verfestigung, Verwendung, Veränderung und Auflösung kultureller Muster.

Es sind gegensätzliche Kulturen, die Probleme mit der Technik machen. Und Probleme können zur Entwicklung der Kultur beitragen. So lösen Gegensätze – die „Widersprüche“ der Dialektik – Entwicklung aus. Das geschieht aber nicht von selbst; nötig sind Menschen – Beteiligte oder Engagierte, Betroffene –, die zwischen den Gegensätzen *vermitteln*. Es gibt viele Weisen, mit Gegensätzen umzugehen. Entweder macht man sie zu Kampfpositionen, schlägt sich auf die eine oder andere Seite, ignoriert oder vermeidet sie, redet sie klein, versucht sie abzutragen, zu versöhnen – auf keine dieser Weisen bringt man die beiden Seiten dazu, sich so miteinander zu befassen, dass die Beteiligten voneinander lernen könnten. Insbesondere ist der berühmte Goldene Mittelweg kein Weg, der weiterführte, denn er vermeidet beide Seiten. Auch die „Synthese“ der Dialektik darf nicht so aussehen, wenn die Entwicklung weitergehen soll. – Oder man nimmt beide Seiten ernst, sucht nach einer Position außerhalb, die beiden gerecht wird, nach einer *Theorie*, die die unterschiedlichen Entwicklungsmuster erkennen lässt; damit kann man zwischen den Gegensätzen zu vermitteln suchen: Mit Hilfe der Theorie können die Beteiligten die Muster so koppeln, dass sie sich aneinander reiben und aufeinander einstimmen können. Dabei können die Seiten sich nähern, vielleicht sogar verschmelzen. Die Entwicklung kann aber auch divergieren, die Gegensätze sich verschärfen. In jedem Fall macht das Miteinander unterschiedlicher Melodien schönere Musik als Sologesänge.

Vermitteln braucht aber Zeit. Entwicklung verläuft nicht nur in Sprüngen, durch Entladen von Gegensätzen, sondern dazwischen in friedlichen Phasen, in denen das Abgeladene eingesammelt und verwertet wird. Menschen leben nicht allein, sie brauchen Austausch, entwickeln sich nur, wenn sie sich miteinander verständigen können. Und Muster verändern sich langsam, haben wir gesehen. Die aufeinander folgenden Phänomene müssen ähnlich genug sein; sonst reißt das Muster ab. Vermittlung durch Theoriearbeit kann und muss beides – Entladung und Verarbeitung, Frieden und Revolution – in Gang bringen und halten. Widersprüche lösen sich nicht von selber auf; erst vermittelt bewirken sie Neues auf höherer Ebene. Entwicklung besteht in der langsamen Veränderung von Mustern, ausgelöst und getragen durch Gegensätze.

2. Mensch und Computer

Das Verhältnis zwischen Mensch und Computer wird unterschiedlich wahrgenommen; vgl. auch (Brö97). Die einen sehen einen Gegensatz: Computer können nicht, was Menschen wesentlich ist, z.B. denken oder fühlen. Und Computer können vieles, was Menschen nicht können, z.B. blitzschnell und fehlerfrei formale Aufgaben durchführen, wie rechnen oder Texte durchsuchen. Die anderen betrachten Mensch und Computer als ähnlich: Computer sind wie Menschen, extrem in eine Richtung entwickelt, aber nicht grundsätzlich verschieden. Daher kann man Menschen durch Computer ersetzen; die Probleme, die dabei auftreten, rühren von der Unvollkommenheit beider, sind prinzipiell vermeidbar. Und man kann Menschen durch Computer simulieren, etwa aus dem „Verhalten“ technischer „Agenten“ auf menschliches Verhalten schließen.

Nach meiner Auffassung sind beide Positionen falsch, aber enthalten Richtiges, das sich sinnvoll kombinieren lässt. Mensch und Computer sind grundsätzlich verschieden: Computer sind nicht lebendig, und Menschen mit maschinenhaften Zügen empfinden wir als krank, wenn nicht tot. Aber Mensch und Computer bilden kein komplementäres Paar: Menschen können gut ohne Computer leben, nur in unserer Kultur nicht mehr. Und Computer brauchen keine Menschen. Menschen müssen sie entwerfen, herstellen, bedienen; aber es sind die Menschen, die das wollen, nicht die Computer. (Und irgendwann können die Computer es selbst, sagen die Technikgläubigen.) Computer sind Menschen nachgebildet, sind das Produkt langer kultureller Entwicklung (Coy85), in der Menschen versuchen, das Chaos, das sie bedroht, in den Griff zu kriegen. Mensch und Computer sind weder gegensätzlich noch ähnlich; Computer sind eine Erstarrung menschlichen Wesens. Das lässt sich besser klarmachen, wenn wir menschliche und maschinelle Muster vergleichen.

Maschinenabläufe sind gleichförmig. Sie reagieren auf Bedienung und Einflüsse aus der Umgebung; aber wenn sie sich unvorgesehen ändern, ist die Maschine kaputt. Maschinelle Muster sind starr. Menschen dagegen, die nicht situationsgerecht reagieren, sind krank (s. oben und Abschnitt 3 unten). Menschliche Muster sind lebendig. Maschinen haben keine Kultur, aber verändern jede menschliche Kultur, in der sie verwendet werden. Ihre starren Muster können bei Herstellern und Nutzern höchst lebendige auslösen, z.B. Ärger über Nichtfunktionieren, Begeisterung an guter Funktion, Freude über Erleichterungen im Leben, Verzweiflung über Nichtverstehen oder Nichtbeherrschen oder Faszination am mächtigen Instrument, neue Arbeitsformen und Erfüllung alter Träume. Aber mit lebendigen Mustern können wir starre nicht zum Schwingen bringen, nicht lebendig machen. Der Satz „Mein Computer versteht

mich nicht“ birgt keine Hoffnung. IT ist ein mächtiges und wertvolles Werkzeug, aber kein Partner.

Mit klassischen Maschinen simulieren und ersetzen wir natürliche Abläufe und körperliche Tätigkeiten. Mit Computern erweitern wir das Vorgehen auf geistige Tätigkeiten, wir „maschinisieren Kopfarbeit“ (Nake in TdI92). Heute würde man sagen: Mit Hilfe von Computern können wir jede geregelte körperliche und geistige Tätigkeit organisieren und ausführen. Das ist nur scheinbar allgemeiner. Wenn man Arbeit als geregelte Tätigkeit definiert, fallen auf jeden Fall Spiele und andere Freizeitbeschäftigungen darunter. (Gibt es schon Computerspiele, bei denen die Maschine gegen sich selbst spielt statt gegen Menschen? Die müssten nur zuschauen – oder auch nicht – und hätten wirklich frei.) Da wir geregelte geistige Aktivität nur bei Menschen kennen, liegt es so nahe, Computer als Partner zu behandeln.

Allerdings ist der Computer in vielen Anwendungen längst kein Gegenüber mehr, er ist von der Maschine übers Werkzeug zum Medium geworden, das menschliche Kooperation und Kommunikation unterstützt, ohne selbst sichtbar zu sein. Aber es sind immer noch geregelte geistige Aktivitäten, die er uns „abnimmt“, wenn auch auf niedrigster Ebene und in unauffälliger Weise. Wie sich menschliche Muster durch diese Unterstützung verändern, ist strittig und noch wenig erforscht. Dass sie sich verändern, ist unbestreitbar: In den „virtuellen Welten“ der IT scheint alles machbar, alles Denkbare kann gleich ohne viel Aufwand und ohne Folgen erprobt werden; das liefert ungeahnte Möglichkeiten und Gefahren für menschliche Entwicklung. Vgl. dazu Heidi Schelhowe (Sce97, 07); schon die Thematik der Bücher („der Computer als“ vs. „Lernen mit IT“) zeigt den Wandel der letzten zehn Jahre.

Umso wichtiger ist es, sich über das Verhältnis des Menschen mit diesem „Medium“ klarzuwerden. In einem Interdisziplinären Forschungsprojekt „Sozialgeschichte der Informatik“ an der TU Berlin (SGI97ff, Sie99ff, Siefkes in Bau01, Hes03) haben wir die besondere Weise, in der Menschen mit IT umgehen, so beschrieben: Sie *hybridisieren* Mensch und Computer mit Hilfe formaler Notationen – Programme und/oder Graphiken oder andere formale Darstellungen auf Papier, Bildschirm oder Tafel. Das heißt, beim Entwerfen wie beim Benutzen überlagern sie in Gedanken die erwarteten oder tatsächlichen maschinellen Abläufe und die menschlichen Verhaltensmuster, denen diese nachgebildet sind. Sie können gar nicht anders, weil sie auf Umgang mit Lebewesen geprägt sind. Vor Augen oder in Händen haben sie aber beides nicht, sondern nur die formalen Gebilde. Da in unserer Kultur Darstellungen das Dargestellte vertreten, übernehmen die formalen Notationen die Rolle der Abwesenden und agieren als Mensch und Maschine gleichzeitig. Als solche sind sie mehrfach auffällig geworden; bei Michaela Reisin und Christiane Floyd heißen sie „autooperationale Form“ (Rei92, Flo97), bei Frieder Nake *algorithmische Zeichen* (Nake in Bau01, Hes03), bei uns *selbstagierende*

Notationen (SGI97ff). So wird der Computer zum „Partner“, und die Menschen können nicht verstehen, wenn er sich anders verhält als erwartet. Wenn sie sich des Vorgehens bewusst sind, ist Hybridisierung ein mächtiges Werkzeug. Sonst verführt es Hersteller wie Anwender dazu, Computer immer menschenähnlicher gestalten bzw. benutzen zu wollen, ohne zu merken, dass dabei auch umgekehrt die Menschen immer maschinenähnlicher werden.

Das gilt für die formalen Texte und Diagramme beim Entwickeln von IT, vom Maschinencode bis zu objektorientierten Programmen. Es gilt aber genauso für die weniger formal scheinenden Darstellungen beim Benutzen, ob wir nun mit Hilfe der IT Diagramme manipulieren oder in virtuellen Räumen spazieren. Wir müssen nur in der Definition von „hybridisieren“ „Mensch und Computer“ durch „Welt und IT“ ersetzen und die bunten Bilder der virtuellen Welt als formale Notationen (an)erkennen – sicher ungewohnt, aber heilsam. (Es gilt sogar für die automatisierte Welt, in die uns IT als Steuereinheit versetzt. Die formalen Notationen sind jetzt optische, akustische oder motorische Signale, wenn sie nicht ganz verschwunden sind.) Die verallgemeinerte Definition zeigt deutlicher, dass menschliche und technische Entwicklung wechselwirken. Die IT ist ja Teil der Welt; also muss sich menschliches Verhalten ändern, wenn sich IT allgegenwärtig ist.

Mensch und Computer bilden kein komplementäres Paar, haben wir oben gesehen; Welt und IT auch nicht. Beide Pole eines komplementären Paares sind eigenständige Bereiche, die sich entwickeln können. IT-Systeme sind in Logik und Mechanik erstarrte Abbilder der Wirklichkeit; sie können sich nicht über das hinaus verändern, was Menschen in das Bild hineingezeichnet haben. Dementsprechend ist Hybridisieren kein Vermitteln. Vermitteln können nur Beteiligte, die bereit und in der Lage sind, in gemeinsamer Arbeit Erfahrungen zwischen beiden Polen hin- und herzutragen, Sichtweisen auszutauschen. Das können Computer nicht. Durch Hybridisieren verschaffen sich Entwickler wie Benutzer Zugang zum Computer, indem sie mit Maschinenmustern wie mit menschlichen Verhaltensweisen umgehen. Der Computer ist also beteiligt, aber passiv, oder nur in den Gedanken der beteiligten Menschen.

Probleme mit IT entstehen also dadurch, dass Entwickler und Benutzer verschiedene Erwartungen an Computer heranbringen und daher, oft ohne es zu wissen, unterschiedliche Bilder und Vorstellungen hybridisieren. Cecile Crutzen und Hans-Werner Hein beschreiben das in (C&H01): Benutzer erwarten, dass das System „verlässlich“ ist, wie ein guter Partner. Das heißt, es tut nicht nur, was es soll, sondern findet auch in unvorhergesehenen Situationen einen Weg. Hersteller wollen aber das System „sicher“ machen, wie eine gute Maschine. Das heißt, es macht keine Fehler und stürzt nicht ab, in unvorgesehenen Situationen verweigert es den Dienst. (Aber in wirklich unvorhergesehenen verhält es sich irregulär, unvorhersehbar.)

Weil beide ihre Vorstellungen mit dem System hybridisieren, verhält es sich – dasselbe System – für beide so verschieden, als seien es zwei Systeme.

Crutzen und Hein zeigen in (C&H07), wie man das ungleiche Verhältnis und die dadurch produzierten Abhängigkeiten durch „Dekonstruktion“ offen legen und durch „Konstruktion“, z.B. durch Umkehren der Bewertungen und Dialog zwischen den Polen, so verändern kann, dass beide Seiten an Freiheit gewinnen. Den Ansatz, der aus der feministischen Theorie kommt, entwickelt Crutzen in (Cru00, 01, auch in TdI02). Mit (fast) denselben Begriffen analysiert Werner Sesink in (TdI03, s. auch Ses04) die Situation bei der Einführung von IT-Systemen: Wenn man wirklich etwas erreichen will, muss man die Arbeitsmuster vorher „de-(kon)struieren“ und hinterher „(re)konstruieren“; s. Abschnitt 5. Mit gleichem Ziel und ähnlichen Mitteln setzt sich Corinna Bath (Bah07) mit verschiedenen Ansätzen auseinander, beim Herstellen und Nutzen von IT-Systemen mit Emotionen umzugehen und dabei Ungleichheitsstrukturen – wie die von Entwickler und Benutzer – zu verfestigen oder zu lockern; vgl. auch (Grü00). Allgemein scheint mir „Dekonstruktion und Konstruktion“ ein guter Weg, auf dem man komplementäre Paare zum Laufen bringen kann. Vermitteln heißt nicht beschwichtigen, sondern Unruhe stiften, indem man beiden Seiten eine Stimme verleiht.

Für Fertig-Systeme kann der Dialog nicht direkt geschehen. Umso wichtiger ist es, dass Benutzer ihre Erfahrungen untereinander austauschen. Für die Nutzung des Internets durch Jugendliche betont Schelhowe das in (Sce07): In virtuellen Räumen können sie ihre Phantasien ungestört und ungestraft erproben und so wertvolle Erfahrungen machen, die sie in der realen Welt nicht (mehr) machen können. Sie müssen diese Erfahrungen aber mit denen Gleichaltriger und Erwachsener abgleichen, im Gespräch und gemeinsamen Tun. Sonst bleiben virtuelle und reale Welt getrennt, und der Effekt wird negativ: Das Ich entwickelt virtuelle Anteile, die mit dem realen Ich immer weniger zu tun haben und sich immer härter mit der realen Welt reiben. – Auch für Entwickler ist der Austausch untereinander wesentlich. Vor allem ist hier aber die Disziplin Informatik gefordert, in interdisziplinärer Zusammenarbeit Wissen über die Problematik in die Wissenschaft einzubringen; siehe dazu die Abschnitte 5,7 und 9. Um was es bei solchen Bewegungen über die Grenzen gehen könnte, zeige ich in den nächsten beiden Abschnitten.

3. Regeln und Verhalten, Zeichen und Bedeutung, Schemata und Phänomene

Klassische Maschinen operieren auf Materie; in Mechanik gegossene Anweisungen werden durch Bedienung und über Sensoren abgearbeitet. Computer operieren auf Daten; in Elektronik gegossene Anweisungen werden durch Bedienung und über Datenabfragen mit Hilfe einer klassischen Maschine abgearbeitet. Man kann auch sagen: Klassische Maschinen folgen Ge-

setzen der Mechanik, Computer folgen Regeln der Logik. (Vgl. auch Brö97 und Brödner, Seim, Wohland in TdI02ff). Computer können so andere Maschinen steuern.

Menschen haben schon immer eigenes und fremdes *Verhalten* durch Regeln zu steuern versucht. In *Regeln* werden vorhandene oder erträumte *Regelmäßigkeiten* des Verhaltens, Verhaltensmuster also (s.u.), explizit formuliert, mit dem Ziel, sie zu erzwingen oder auszuschließen. Die gesammelten Regeln oder die durch sie geregelte Gruppierung nennt man *Organisation*. Das Neue an Computern ist, dass sie Regeln einfach ausführen, während Menschen sie erst interpretieren müssen. Interpretieren gibt Spielraum; je nach Situation, Erinnerung, Ziel, Verständnis, Stimmung kommen andere Ergebnisse heraus. Menschen müssen Regeln Sinn verleihen, d.h. lebendige Muster finden, die mit den Regeln mehr oder weniger verträglich sind und zu ihren eigenen Mustern mehr oder weniger passen. Regeln erzeugen keine Verhaltensmuster, sie geben nur einige vor und schließen andere aus. Regeln sind keine Generatoren, nur Wegweiser (Gid84). Das bedenken Regler in der Regel nicht, sie setzen ihre Regeln mit den Regelmäßigkeiten gleich, die sie darin erfassen möchten. Hinzu kommt, dass Menschen Regeln jederzeit ändern können – sei es, dass sie als Regler die Macht dazu haben, dass sie als Geregelte sich widersetzen oder dass sie gemeinsam die Organisation reformieren.

Tatsächlich ist der Unterschied noch schärfer: Menschen können Regeln immer übertreten, Computer nie. Menschen handeln gegen Regeln – aus der Situation heraus oder lang geplant, in Begeisterung oder aus gestautem Ärger, im eigenen Interesse oder dem der Organisation. Diese von der Organisation nicht intendierten oder sogar nicht erwünschten Muster lassen die Beteiligten ihre Aufgaben erst erfüllen. Abweichendes Verhalten ist Anlass und Hintergrund für Veränderung, also für Entwicklung der Organisation. Allerdings ist das Verhältnis nicht stabil; es kann ihr auch schaden oder sie zerstören. Das ist der Kern von Giddens Strukturationstheorie (Gid84); mehr dazu in Abschnitt 4.

Menschliches Verhalten ist nicht völlig vorhersehbar, geschweige denn vorschreibbar. Das ist ein Glück: Keine Organisation würde sonst funktionieren. Dienst nach Vorschrift ruiniert die Organisation. Günther Ortmann (Ort03) formuliert das als *Paradox der Organisationstheorie*: Verhalten gegen die Organisation oder an ihr vorbei („fummeln, bummeln, schummeln“) kann dem Erhalt der Organisation dienen. Computer dagegen, die ihre Regeln aufgrund der Situation oder ihrer Ergebnisse übertreten oder verändern – „adaptive“ oder „lernende“ Programme –, sind dafür programmiert; Anpassung oder Lernen sind in Meta-Regeln „vorgesehen“, tatsächlich also vorgeschrieben. Computer machen immer Dienst nach Vorschrift; das müssen Entwickler bedenken, wenn sie eine Organisation oder Teile davon in IT „abbilden“. Mehr dazu in Abschnitt 5.

Regeln müssen wir interpretieren, bevor wir sie ausführen können, weil sie sprachlich oder mit Hilfe anderer Zeichen formuliert sind. *Zeichen* verweisen auf einen Sachverhalt, außerhalb oder in uns, den wir finden sollen. Das Suchen und sein Ergebnis hängen von der Situation ab, immer ähnlich, aber nicht gleich. Zeichen erzeugen lebendige Muster, Kommunikationsmuster im allgemeinen. Peirce (Pei86) nennt den Vorgang *Semiosis*; jeder Schritt führt das Muster weiter, verändert das Zeichen ein wenig, bringt neue Zeichen hervor. Das Muster ist die *Bedeutung* des Zeichens. Umgekehrt entstehen Zeichen aus Mustern, die wir wahrnehmen: Ein Phänomen wiederholt sich, immer ähnlich, bis es uns auffällt und wir ihm einen Namen geben, ein Zeichen für das Muster. Ein Zeichen ist für Menschen, die es kennen, Kern und Generator eines lebendigen Bedeutungsmusters.

Nicht zufällig ist die Situation ähnlich wie bei den Regeln. Wenn wir uns verständigen, setzen wir voraus, dass die benutzten Zeichen für alle Beteiligten dasselbe bedeuten. Die Bedeutungsmuster sind aber unterschiedlich, sie hängen ja von Person und Situation ab. Wenn wir Zeichen benutzen, passen wir sie in unsere Lebensgeschichte ein. Wir verständigen uns trotzdem, solange die Kerne der Muster, mit denen wir das Gemeinsame der sich wandelnden Wiederholungen erfassen, hinreichend ähnlich sind. (Zu den Prozessen, die das ermöglichen, vgl. Abschnitt 4.)

Wir haben also ein *Paradox der Semiotik*, das dem der Organisationstheorie zugrunde liegt: Menschen verständigen sich mit Zeichen, obwohl, nein, weil sie sie unterschiedlich interpretieren. Differenzen im Verständnis sind unvermeidlich, weil Menschen verschieden sind. Wie bei den Regeln bereiten sie Probleme, sind aber unerlässlich für die Entwicklung. Botschaften werden zu Information erst, wenn die Beteiligten sie verarbeiten und sich dadurch verändern; vgl. Abschnitt 1. Und wie bei jeder Entwicklung ist das Verhältnis nicht stabil; Differenzen können jederzeit der Kommunikation schaden oder sie zerstören. Daher gibt es ein Unglück, wenn wir Regeln einer Organisation direkt in IT umsetzen (s.u. und Abschnitt 5). Computer interpretieren die Zeichen nicht, mit denen die umgesetzten (also eindeutig gemachten) Regeln formuliert sind, sie führen sie aus; aus Zeichen werden im Computer *Signale* (Nake in Bau01, TdI01, Hes03).

In psychologischer Terminologie sind es *Schemata*, die geistig-körperliche Verhaltensweisen erzeugen – *Phänomene*, die sich immer wieder in bestimmten Situationen zeigen, jeweils der Situation entsprechend. (Ste95, Joh87; der Psychologe Stern verweist auf Piaget, der Philosoph Johnson auf Kant.) Verhaltensweisen sind also Muster mit Schemata als Generatoren. Umgekehrt entstehen Schemata und wandeln sich in der Auseinandersetzung mit geistigen, körperlichen und sozialen Mustern, eigenen und fremden. Wir lernen durch Wiederholungen,

wenn sie Muster bilden, mit denen wir leben können. Starre oder chaotische geistige Muster sind Anzeichen psychischer Störungen. Schwer depressive Menschen z.B. zeigen starre Verhaltensweisen mit chaotischen Einsprengungen, bei schizophrenen ist es umgekehrt, autistische kommen dem maschinellen Idealbild noch näher. Dauernder Umgang mit IT kann starres, dauernde Probleme mit IT können chaotisches Verhalten auslösen. Wenn allerdings Computer selber chaotisch zu reagieren scheinen, liegt es meist an uns: Wir sind gewohnt, kleine Unterschiede zu ignorieren, Situationen gleichzusetzen, die es nicht sind. Funktionierende Computer laufen starr.

Noch unter den beiden betrachteten liegt also ein *Paradox der Lerntheorie*: Das Verhalten von Menschen ist durch Schemata aller Art bestimmt, die sie mitbringen und erlernen (zu den Prozessen, die das ermöglichen, vgl. Abschnitt 4); Menschen sind aber lebendig nur, soweit sie diese Schemata durchbrechen können, sobald es die Situation erfordert. Die Schemata *und* die Fähigkeit, sie zeitweilig zu verlassen und so langfristig zu verändern, machen menschliche Identität aus.

Die klassischen geistigen Schemata der Philosophie sind *Begriffe* und *Werte*; sie erzeugen Denk- bzw. Gefühlsmuster. Wenn wir ein neues Muster wahrnehmen und gedanklich einordnen, haben wir seine unterschiedlichen Phänomene unter einen Begriff gebracht, mit dem wir sie reproduzieren können. Verarbeiten wir es gefühlsmäßig, ordnen wir es in unser Wertesystem ein. Soziologen fassen Handeln und Wahrnehmung als Verhalten zusammen und sprechen von schematischem Verhalten als *Gewohnheiten* (habits). Ohne feste geistige und soziale Gewohnheiten könnten wir nicht leben; sie machen den größten Teil unseres Verhaltens aus. Werden wir aber gezwungen, sie zu ändern, kann das Existenzängste auslösen (Gid84). Starre Gewohnheiten andererseits, die sich nicht mehr ändern, verhindern Entwicklung. Crutzen und Hein nennen sie, Dewey folgend, *Routinen* (Cru00, C&H07). Bei anderen Menschen stoßen sie uns ab, bei uns selber bemerken wir sie oft nicht. In der Regel unterscheiden Soziologen Muster nicht von ihren Generatoren, kennen also keine Schemata.

Verhaltensmuster sind körperliche Muster, die von geistigen gesteuert werden; s. auch Anfang Abschnitt 1. Ich nenne sie *soziale* Muster, weil sie nach außen wirken. Kommunikationsmuster sind spezielle soziale Muster; aber alle menschlichen Muster – körperliche, sprachliche, bildliche – können der Kommunikation dienen. Geistige und soziale Muster kennzeichnen einen Menschen; die Sammlung der dazugehörigen Schemata nennen wir seinen *Charakter*.

Nach psychologischer Theorie kommen menschliche Werte in Paaren (Klu07, LeD01): Alles, was uns wertvoll ist, – z.B. Sicherheit, Anerkennung, Beziehung – hat einen "Schatten" – Unsicherheit, Ablehnung, Einsamkeit. Der hellen Seite widmen wir unsere ganze Aufmerk-

samkeit, von der dunklen entfernen wir uns oder ignorieren sie. Aber gerade dadurch spielt die Schattenseite eine wichtige Rolle in unserem Leben, sie beeinflusst unser Denken und Fühlen, unser Handeln und Wahrnehmen genauso wie die Sonnenseite. Wir tun gut daran, beide Seiten zu kennen. Dann können wir zwischen ihnen vermitteln und sie zu komplementären Paaren machen. Wie man eine solche Veränderung auf radikale Weise erreichen kann, zeigt Cecile Crutzen (l.c. Ende Abschnitt 2): Man kehrt die einseitige Bewertung der Pole um und macht so ein neues Verständnis beider Seiten möglich; genauer in Abschnitt 7.

Tatsächlich ist die Situation verwickelter. Betrachten wir die Paare zuverlässig - unzuverlässig, gehorsam - ungehorsam, selbständig - unselbständig. Sind zuverlässige Menschen nicht unselbständig? Ist nicht Ungehorsam manchmal Pflicht? Eigenständige Menschen gehorchen oder nicht, je nach Situation; sie haben ihren eigenen Stand. Bewertungen sind nicht nur in jeder Kultur, bei jedem Menschen anders, sie sind auch auf vertrackte Weise überkreuz verknüpft. Also mache ich aus den Paaren *Spektren* oder besser *Cluster*:

- von extremen – gefügig, abhängig, gehorsam,
- über mittlere – zuverlässig, selbständig, eigenständig, eigenwillig, eigensinnig
- zu anderen extremen Positionen – ungehorsam, unzugänglich, aufsässig.

Positiv bewerten wir die mittleren Positionen, aber nur, wenn sie flexibel gehalten werden. Zuverlässige Mitarbeiter missachten Regeln, wenn sie dadurch bessere Ergebnisse erzielen, und befolgen sie sklavisch, wenn sie nur so ein Unglück vermeiden können. Das Schema der Zuverlässigkeit erzeugt "in der Regel" die erwarteten mittleren Positionen aus dem Cluster, aber in besonderen Situationen eben auch die Extrema. Das ist wichtig für Giddens Strukturationstheorie und steckt hinter den drei Paradoxen oben.

Verhalten, Bedeutungen, Phänomene aller Art kommen für Menschen in Clustern – klar genug gezeichnet, um damit zu leben, aber mit Ausrutschern nach allen Seiten. Der Sehnsucht nach Beziehungen kann die Angst vor Einsamkeit oder die Lust an Abhängigkeit gegenüberstehen. Nach Jürgen Link (Lin97) kennzeichnet es die Moderne, solche Verteilungen zu „normalisieren“: mit Hilfe eines Maßes zu linearisieren und dann über der Geraden „normal“ zu verteilen, das „Gute“ in der Mitte, mit Abweichungen rechts und links. Auch in der Dialektik, von der meine komplementären Paare kommen, wird mit Hilfe von Widersprüchen linearisiert, dann aber nicht normalisiert, sondern Spannung aufgebaut. (Man könnte also Dialektik und komplementäre Paare in Richtung Clusterbildung verallgemeinern.)

Damit kann ich die Arbeiten von Crutzen und Hein weiterführen, die in (C&H01, 07) menschliche Zuverlässigkeit mit der "Dienlichkeit und Sicherheit von Softwaresystemen" kontrastieren; vgl. oben Abschnitt 2. Ein Computer darf und kann sich nicht wie ein Mensch „normal mit Abweichungen“ verhalten. Wenn wir menschliche Eigenschaften auf den Com-

puter übertragen, müssen wir uns für einen festen Wert entscheiden. Wir wählen immer das Extrem, das Kontrolle und Verfügbarkeit garantiert; anders geht es nicht. Vorhersehbare Abhängigkeiten programmieren wir ein, aber wirklich Neues ist nicht vorhersehbar. Maschinen sollen und können nicht selbständig sein; oder gar aufsässig oder chaotisch. In Menschen, die mit dem System umgehen, lösen die starren maschinellen Muster aber die Schemata mit dem vollen Cluster aus und schaffen damit Erwartungen, die das System nicht erfüllen kann. Das Leitbild der Informatik ist Beherrschbarkeit, konstatiert Ralf Klischewski (Kli96) und setzt als Ausgleich daneben das Gegenbild 'Anarchie'; vgl. Abschnitt 7.

Die verschiedenen Arten von Schemata – geistige und körperliche, Denk- und Gefühls-, Handlungs- und Wahrnehmungsmuster – sind stark gekoppelt, sodass sie sich immer gegenseitig beeinflussen: Wahrnehmungen und Gedanken lösen Gefühle aus; Gefühle werden gedanklich verarbeitet; wir lernen am besten, wenn Geist und Körper beteiligt sind; usw. (Ste95, Cio97). Dasselbe sagt in anderer Terminologie die Tätigkeitstheorie (Vyg78). Auch auf diese Weise können starre Muster, wie die von IT erzeugten, lebendige, ja leidenschaftliche Muster in Gang bringen, wenn wir mit ihnen in Berührung kommen. So kann die Einführung neuer IT-Systeme eine Organisation völlig verändern, auch wenn sie funktional dasselbe leisten wie vorher die Mitarbeiter; dem Problem gehe ich in Abschnitt 5 nach. Und so kann unreflektiertes Nutzen des Computers als Medium den Charakter verändern – umso mehr, je ähnlicher die virtuelle Welt der realen gemacht wird; s.o. (Sce07).

Damit haben wir das nötige Werkzeug beisammen, um Muster in Kulturen aufzuspüren und zu verstehen. Menschliches Tun und Denken, Fühlen und Wahrnehmen und alle davon berührten Phänomene verfestigen sich in Schemata, die Muster aller Art entstehen und erkennen lassen (*Schematisieren*: Entwicklung körperlicher und geistiger Fähigkeiten). Dabei entstehen Zeichensysteme, mit denen man sich über solche Muster verständigt (*Semiotisieren*: Entwicklung von Kommunikation), und Regelsysteme, mit denen man gemeinsames Handeln steuert (*Organisieren*: Entwicklung von Kooperation). Die Entwicklungen wechselwirken, und die zugehörigen „Phänomene“ – Organisationen, Gesten/Bilder/Sprachen, Menschen – bilden in ihrer Verschränkung menschliche Gesellschaften. Eine *Kultur* ist gekennzeichnet durch die Muster von Kooperation, Kommunikation und Charakter, die sie aufweist; vgl. oben Abschnitt 1. Kulturelle Muster unter den drei genannten Aspekten zu analysieren, könnte einen Rahmen liefern, in dem man die unzähligen Darstellungen und Theorien zur Entwicklung menschlicher Kulturen aufarbeiten und in einer Theorie vereinheitlichen kann. Als eine solche Vereinheitlichung verstehe ich die Strukturationstheorie von Giddens (Gid84; s. Abschnitt 4).

Ausführlicher behandele ich die Thematik, vor allem die Rolle von Schemata im Umgang mit IT, in (Sie99ff).

Probleme mit IT rühren daher, dass die verschiedenen Kulturen, die beim Umgang mit der Technik aufeinanderprallen, – Entwickler und Anwender etwa oder Manager und Mitarbeiter – ihre verschiedenen oder gar unverträglichen Muster in die Technik oder den Umgang mit ihr „einbauen". Wanda Orlikowski (Orl00) unterscheidet deswegen Technik von „Technik-in-Aktion": Derselbe Gegenstand kann je nach den Menschen, die am Umgang (Herstellung, Einsatz, Nutzung) beteiligt sind, ganz unterschiedlich wirken. Viele Menschen stehen IT mit Unverständnis gegenüber – fassungslos oder fasziniert. Tatsächlich verstehen sie (als Benutzer) diejenigen nicht, die ihnen das System gebaut, verkauft, aufgezwungen oder empfohlen haben, oder (als Hersteller) diejenigen, die es blind oder widerwillig oder gar nicht benutzen. Wenn wir auf dieses Unverständnis zwischen zwei Kulturen, die mit derselben Technik umgehen, die oben anvisierte allgemeine Theorie kultureller Entwicklung anwenden, gewinnen wir ein Stück der Theorie der Informatik, nach der wir suchen. Die Aufgabe gehen wir in Abschnitt 5 an; vorher müssen wir das Verhältnis von Entwicklern und Benutzern oder anderen zu ihren „Kulturen" klären.

4. Individuum und Gesellschaft

Nach allgemeiner soziologischer Sicht wird individuelles Handeln vom gesellschaftlichen Rahmen getragen. Die Gesellschaft stellt die Regeln und Ressourcen bereit, die menschliches Leben ermöglichen und einschränken. Die Gesellschaft gibt die Bedingungen vor, in die die Einzelnen sich einpassen müssen. Für viele stehen daher Individuum und Gesellschaft im Gegensatz. Wie aber entsteht der gesellschaftliche Rahmen? „Society constitutes itself through the activities of its members", sagt der Soziologe (und Psychologe) Anthony Giddens in (Gid84). Society ist für ihn jede Art menschlicher Gemeinschaft, von kleinen informellen Gruppen bis zu beliebig großen wirtschaftlichen, politischen und kulturellen Einheiten. Die Typen geistiger und sozialer Muster, die solche Gemeinschaften charakterisieren, nenne ich Kulturen (Abschnitte 1, 3).

Damit kann ich Giddens' Aussage so formulieren: Die geistigen und sozialen Muster einer Kultur lassen die gesellschaftlichen Regeln und Ressourcen erst entstehen, sich wandeln und vergehen. Eine Gesellschaft, in der zu viele Aktivitätsmuster den gemeinsamen Grundsätzen zuwiderlaufen, kann für eine Weile mit Anstrengung erhalten werden und zerfällt dann. Regeln und Ressourcen wirken also konkret („von unten") über das Wissen und die Motive der Beteiligten, nicht abstrakt („von oben") als gesellschaftliche Mittel und Zwänge. Die Gesellschaft liefert die Ressourcen, die menschliche Kommunikation und Kooperation möglich ma-

chen, und die Regeln, die dabei bestimmte Muster vorschreiben, andere ausschließen. Umgekehrt entstehen gesellschaftliche Strukturen als Sediment solcher Muster („Tradition“), oder sie werden dafür geschaffen („Gestaltung“). In Abschnitt 3 habe ich Regel- und Zeichensysteme, also organisatorische und kommunikative Strukturen, als Beispiele betrachtet. Herrschaftssysteme, die bei Giddens die zentrale Dimension liefern, fehlen bei mir; ich führe ‚Macht’ auf die Cluster ‚Kooperation’ und ‚Kommunikation’ mit den Extremen ‚beherrschen/unterwerfen’ bzw. ‚befehlen/gehorchen’ zurück (s. Abschnitt 3).

Giddens’ Strukturationstheorie findet sich im Kern schon bei Hannah Arendt (Are58), in härterer Form und historisch differenziert: Die „Fiktion“ einer einheitlichen Gesellschaft übt auf ihre Mitglieder ungeheuren Zwang aus, sie werden auf durchschnittliches Verhalten normiert; vgl. Links „Normalisierung“ (Lin97), Abschnitt 3. Der Zwang ist umso stärker, je größer die Gesellschaft ist; am grausamsten ist er in der heutigen Massengesellschaft unter der Herrschaft einer anonymen Bürokratie. Aber Alltägliches bezieht seinen Sinn aus den herausragenden Taten der Menschen und den Ereignissen der Geschichte, die es erst konstituieren (sic!). Nur durch menschliches Tun wird die Fiktion zur Wirklichkeit.

Das Paar ‚Individuum und Gesellschaft’ lässt sich nicht auf das Gegensatzpaar ‚Form und Prozess’ (Bat72, 79) zurückführen, so dass Gesellschaft aus Strukturen bestünde und Individuen aus Aktivitäten. Es gibt individuelle Strukturen wie Charakter (Abschnitt 3); sie werden aber nicht nur lokal geprägt, durch Familie und Freundschaft, sondern ebenso durch die in jeder Kultur wirksamen Bewertungen. Und es gibt gesellschaftliche Prozesse wie Technisierung oder Aufklärung, die aber von Individuen getragen werden. Gesellschaft wie Individuum entwickeln sich in unterschiedlichen Mustern, die sich wechselseitig beeinflussen. Bilden also Individuum und Gesellschaft ein komplementäres Paar?

Die Wechselwirkung wird deutlicher, wenn wir zwischen beiden Ebenen eine dritte einführen. Vermitteln kann man zwischen den Ebenen nur abstrakt, konkret zwischen individuellen Kulturen und den unterschiedlichen Kulturen gesellschaftlicher Gruppierungen, wie Erwachsene, Staat, Wirtschaft. Konkret läuft die dadurch angestoßene Entwicklung im Rahmen realer Organisationen wie Firmen, Schulen, Vereinen ab. Aber auch die treten selten direkt in Erscheinung. Menschen sind soziale Wesen. Sie handeln im Rahmen organisierter Gemeinschaften selten als Einzelpersonen, sondern meist in kleinen informellen Gruppen. Für Giddens ist es „co-presence“, die den Unterschied macht, „face-to-face communication and cooperation“. Das ist mir zu eng, weil es technikbasierte Beziehungen wie e-mail ausschließt, und zu weit, weil es gestörte Beziehungen wie Verachtung zulässt, die Kommunikation und Kooperation beeinträchtigen. In (Sie82, 92) definiere ich *kleine Systeme* als Gruppen, in denen nicht nur die sozialen Gegebenheiten (Kooperation und Kommunikation), sondern auch die

äußeren, wie Dinge und Regeln, und die geistigen, wie Wissen und Werte, von den Mitgliedern gestaltet werden können und daher die Entwicklung der Gruppe fördern. Das heißt nicht, dass es in kleinen Systemen keine Konflikte gibt; sie müssen aber von der Gruppe bearbeitet und gelöst werden können. Die Definition schließt große Gruppen aus, die auf technische Unterstützung und strikte Regeln angewiesen sind; aber zahlenmäßig kleine Gruppen müssen nicht kleine Systeme sein. Eine erkaltete Partnerschaft z.B. ist keins, weil beide nur noch nach starren Regeln miteinander verkehren. In einem kleinen System belastet keine der sechs „Bestimmenden" die Beziehungen durch Über- oder Untermaß; die Mitglieder gehen aber nicht den „goldenen Mittelweg", sondern bewegen sich gemeinsam zwischen den Extremen.

Der Charakter eines kleinen Systems ergibt sich aus den Eigenheiten der beteiligten Menschen und bestimmt umgekehrt, wie mentale und soziale Muster entstehen, wirken und vergehen. Und die gesellschaftlichen Gegensätze, die in Gruppen aufeinanderstoßen, führen zu Veränderungen, die sich in den Umgebungen bis hin zur ganzen Gesellschaft als menschliche, technische, organisatorische und kulturelle „Innovationen" niederschlagen können. Kleine Systeme bilden also mit den Individuen einerseits und mit den großen Organisationen andererseits komplementäre Paare. Kleine Systeme sind die Zentren, durch die hindurch individuelle und gesellschaftliche Entwicklung verläuft. Um den Austausch zwischen unterschiedlichen Kulturen zu fördern, müssen wir sie in kleinen Systemen zusammenbringen. Individuum und Gesellschaft bilden also nur indirekt ein komplementäres Paar; genauer ist es ein „Doppelstern" aus zwei komplementären Paaren mit den kleinen Systemen im Zentrum. Und „Gesellschaft" ist Summe oder Abstraktion aller Institutionen und anderer großer Gruppierungen, die sie ausmachen, wie Schule, Wirtschaft, Arbeitnehmer, Erwachsene.

Die Ebene der kleinen Systeme fehlt bei Hannah Arendt, Individuum und Gesellschaft stehen sich unvermittelt gegenüber. „Gesellschaft" entstand mit Beginn der Neuzeit, als das Arbeiten und Herstellen der Menschen aus dem privaten Raum in die Öffentlichkeit gezogen wurde. Damit verloren Familie und andere soziale Gruppierungen (kleine Systeme) an Bedeutung. Mir erscheinen aber die Veränderungen, die die IT mit sich gebracht hat, nicht verständlich, wenn man kleine Systeme nicht in die Betrachtung einbezieht. IT-Systeme verstärken dramatisch die Machtmittel von Bürokraten und Organisatoren, Globalisierung wäre ohne sie nicht möglich. Aber gleichzeitig verschaffen sie dem Einzelnen ungeahnte Möglichkeiten, sich außerhalb der vorgezeichneten „normalen Fahrten" (Lin97) zu bewegen und so der Einflussnahme von oben zu entziehen. Diese Möglichkeiten werden aber Wirklichkeit erst, wenn sie in Kommunikation und Kooperation, also in kleinen Systemen, mit der „großen" Welt verknüpft werden; vgl. Schelhowe (Sce07), Abschnitt 2.

Das Konzept der kleinen Systeme ist in den Sozialwissenschaften unüblich; es erscheint normativ und empirisch-analytisch nicht begründet. Vor- und Nachteile von IT beruhen aber gerade darauf, dass informelle Kommunikation und Kooperation durch IT-Einführung „eingefangen“ wird, dass also kleine Systeme „groß“ gemacht werden. Das neue Gebiet Computer Supported Cooperative Work (CSCW) beschäftigt sich genau mit dem Phänomen, ohne das Konzept einzuführen. Auch in pädagogischen Untersuchungen, zur Gruppendynamik etwa, wird das Konzept oft benutzt, ohne es zu präzisieren. Ich behaupte, dass es den meisten sozialwissenschaftlichen Untersuchungen individuellen Verhaltens unbenannt zugrundeliegt. „Akteure“ etwa sind in der Regel kleine Systeme; „Individuen“ sind für sich so wenig „handlungsfähig“ wie Organisationen. Das Konzept ist also empirisch wohlfundiert, auch wenn es zahlenmäßig nicht fassbar, also nicht messbar ist.

Die Theorie dieser beiden Abschnitte nenne ich „evolutionär“ oder „kulturell“, weil sie Kultur als Entwicklung betrachtet. Den Anstoß dazu hat Gregory Bateson gegeben (Bat72, 79), der individuelle Entwicklung als Evolution auffasst (Siefkes in TdI92) und Informationsübertragung als Kopplung von Mustern versteht (s.o.). Aufgrund der Erfahrungen mit dem Sozialgeschichteprojekt (SGI97ff; s. Abschnitt 2, Ende) habe ich sie auf kulturelle Entwicklung erweitert (Sie95ff). Ich spreche vom „Schmetterlingsmodell“, weil geistige und soziale Entwicklung analog sind. Sie bilden die Flügel des Schmetterlings, die durch die Schemata verbunden sind, die im Körper angelegt sind und so in kleinen Systemen direkt wirken können. Wesentliche Beiträge zu der Theorie kamen aus der Psychologie (Stern, Ciompi), Philosophie (Johnson), Semiotik (Nake), Pädagogik (Sesink) und den Sozialwissenschaften (Giddens, Ortmann, Rolf). Rolf fasst, Giddens folgend, Individuen und Gruppen als Akteure zusammen und nennt das (Mikro- und Makrosicht verbindende) Bild der Wechselwirkung zwischen Akteuren und ihren gesellschaftlichen Arenen „Mikropolismodell“ (Rol98, 07, Kra06, G&R07, Sie07c).

5. Entwickler und Benutzer

Informatiker und andere geben sich große Mühe, gute IT zu erstellen. Wesentliche Bereiche der Wissenschaft Informatik widmen sich direkt oder indirekt dem Vorgang. Trotzdem gibt es so viele Probleme beim Benutzen. Betrachten wir, was geschieht, wenn IT in eine Organisation eingeführt wird, in einen Betrieb, eine Verwaltung, eine Schule. Das Beispiel ist sehr allgemein, und das dazu Gesagte lässt sich auf andere Bereiche, z.B. Computerspiele oder andere interaktive Systeme, übertragen; vgl. dazu Abschnitt 2 („Computer als Medium“, Schelhowe).

Nach Abschnitt 3 gibt es ein Unglück, wenn man bei der IT-Gestaltung die Regeln der Organisation einfach nachbildet. Aus einem Regelsystem, das die Bildung lebendiger Arbeitsmuster unterstützen soll, wird so ein technisches System, das nur starre Muster erzeugt und die Organisation erstickt (Brödner, Seim, Wohland in TdI02ff). Gefördert wird diese verbreitete Sünde durch Gleichsetzen von Formalismen und Maschinen (gelegentlich in der Logik), von menschlichem Denken und formalem Schließen (KI) oder von sozialen und technischen Systemen (z.B. Sozionik); s. dazu Abschnitte 2, 8. Einführen von IT in Organisationen verlangt nicht nur Gestaltung von IT, sondern ebenso Umgestaltung von Organisation (Rol98, 07, Kra06, G&R07, MMK06).

Ausgangspunkt für Softwareentwicklung ist heute meist eine genaue Beschreibung der Situation („Anforderungsdefinition“), die die zukünftigen Benutzer oder andere Experten liefern. Die Entwickler überführen die Beschreibung schrittweise in ein Softwaresystem: Sie eliminieren alle Bezüge nach außen und machen so die Bedeutung der Beschreibung eindeutig (*Formalisieren*); danach oder dabei formen sie die Beschreibung in Anweisungen um, sodass sie immer eindeutige Ergebnisse liefert (*Algorithmisieren*); schließlich erstellen sie daraus lauffähige Software (*Maschinisieren*). Viele vermeiden die mathematischen Ausdrücke „formal“ und „algorithmisch“ und den technischen Ausdruck „Maschine“ (warum? Siehe Abschnitte 2, 6) und erstellen nach der Beschreibung ein Modell (*Modellieren*), das sie direkt auf die Maschine bringen (*Implementieren*). Da Programme formal und algorithmisch sind, wird auch dabei formalisiert und algorithmisiert, nebenbei und schon vorher bei der Entwicklung des Modellsystems; und natürlich ist die Maschine das Ziel. Ich halte mich deswegen an die erste Aufteilung.

Früher war die Aufgabe der Entwickler damit beendet, sie übergaben das IT-System den Nutzern. Wenn es Probleme gab, wurde die Schuld hin- und hergeschoben: „Das System tut nicht, was es soll.“ „Die Beschreibung war schlecht.“ „Die Entwickler verstehen nur Formalismen und Technik, sie können anderes nicht lesen.“ „Die Benutzer können mit dem System nicht umgehen.“ Deswegen liefert die Informatik immer neue Ansätze: Neue Programmiersprachen und Modellierungstechniken werden entwickelt, die anschaulicher sind und mit Computern nichts zu tun zu haben scheinen. Mit zyklischem Vorgehen, Prototyping und Partizipation (Flo92) werden die Vorgänge überschaubarer gemacht und die Benutzer einbezogen. Im „Contextual Design“ (B&H97, 99) werden solche Ansätze zur Methode gemacht (s. u.). Auch auf der Benutzerseite geschieht viel: Die Mitarbeiter werden geschult, die Anforderungen von Mitarbeitern und Managern und vielleicht Entwicklern gemeinsam erstellt. Aber für die Schritte von der lebendigen Organisation zu den Anforderungen und zurück von der fertigen IT zur Organisation scheint es keine wissenschaftlich fundierten Vorgehensweisen zu

geben, die dem informatischen Dreischritt „formalisieren, algorithmisieren, maschinisieren“ entsprächen. Vielleicht sind sie und daher ihre Planung nicht genau fassbar und deswegen wissenschaftlich nicht greifbar. Wohl deswegen spricht man heute von IT-Gestaltung statt -Erstellung; um nutzbare IT zu erstellen, braucht es Kunst, nicht nur Wissenschaft.

In Abschnitt 3 haben wir gesehen, dass drei Weisen, Ordnung ins Chaos zu bringen, – „Schematisieren, Semiotisieren, Organisieren“ – jeder menschlichen Kultur zugrunde liegen, insbesondere den informatischen Dreischritt erst möglich machen. Wenn wir die Kunst der IT-Gestaltung wissenschaftlich zähmen wollen, sollten wir also nach Mustern suchen, die die Beteiligten im jeweiligen Einsatzbereich maschinisiert haben möchten, und fragen, wie sie und die Umgebung sich verändern, wenn wir sie durch IT-Muster ersetzen. Auf diese Weise verzahnen wir den allgemeinen theoretischen Rahmen mit informatischer Praxis, machen also einen weiteren Schritt auf dem Weg zu einer Theorie der Informatik.

Werner Sesink hat in (TdI03) die beiden Vorgänge vor und nach der informatischen Arbeit, das Erstellen der Anforderungen und das Umstellen auf die Benutzung, „Dekonstruktion“ und „Rekonstruktion“ genannt, oder härter und, wie ich meine, treffender „*Destruktion*“ und „*Konstruktion*“. Damit weist er auf die Defizite der bisherigen Darstellung hin. Entwickler können Kopfarbeit und andere menschliche Tätigkeiten nicht direkt maschinisieren, sondern müssen sie erst „maschinenhaft denken“ (Nake in TdI92), d.h. die lebendigen Muster erstarren lassen. Und die Benutzer müssen diese erstarrten Muster innerhalb der alten lebendigen Umgebung nutzen lernen. Bevor die Benutzer Anforderungen erstellen können, müssen sie also Vorgänge, die auf den Computer gebracht werden sollen (und können?), identifizieren und aus dem lebendigen Zusammenhang herauslösen. Dabei (zer)stören, destruieren, sie an den Stellen die vorhandene Organisation. Und um das fertige IT-System zu nutzen, müssen sie die Verbindungen wieder herstellen. Das können aber nicht dieselben sein, da an die Stelle von Maschinen Menschen getreten sind. Da sie die maschinellen Muster nicht verändern können – sie sind ja nur Benutzer –, müssen sie die Umgebungen verändern, neu konstruieren. „Changing the organization“ ist explizites Ziel aller Schritte im Contextual Design (s.o.). Und das Buch (Flo92), in dem viel über den zweiten Schritt steht, trägt den Titel „Software Development and Reality Construction“. Dort ist der Zugang der radikale Konstruktivismus (von Glasersfeld), bei mir eine evolutionäre Theorie. Die noch gefälligere Bezeichnung „De- und Rekontextualisierung“, die Sesink auch gebraucht, ist irreführend, weil sie suggeriert, es müssten nur wie im Texteditor Teile der Organisation herausgeschnitten und durch IT ersetzt werden. Tatsächlich muss der ganze Bereich neu etabliert werden. Das wird leicht übersehen oder unterschätzt, da die zu IT gewordenen Teile vorher schon mehr oder weniger schema-

tisch abliefen, sprachlich gefasst und organisatorisch geregelt waren. Deswegen müssen solche Vorgaben schon bei der Destruktion herausgearbeitet werden (s.u.).

In Anlehnung an Sesinks Bezeichnungen fasse ich die drei informatischen Schritte als *Instruktion* zusammen: Informatiker instruieren den Computer, und die Nutzer gleich mit. Tatsächlich betrachtet er als Pädagoge IT-Einführung als Lehr- und Lernvorgang (Ses01, 04, Sesink in TdI03). Ein guter Lehrer trichtert seinen Schülern nicht einfach etwas ein, sondern hilft ihnen dabei, alte Gewohnheiten des Denkens und Handelns abzulegen und neue zu gewinnen; nur so verarbeiten sie den Stoff, nur so lernen sie. IT-Entwickler sollten gute Lehrer in diesem Sinne sein; vgl. Abschnitt 9. – Ich beschreibe die drei Phasen jetzt genauer mit Hilfe von Mustern.

Menschliches Tun muss schematisch werden, bevor es fassbar ist (Abschnitt 3). Für ihre Anforderungen müssen die Benutzer also nach Regelmäßigkeiten suchen; sporadische oder chaotische Handlungen kann man weder beschreiben noch regeln. Für die Muster, die sie so finden oder nach ihren Vorstellungen entwickeln, müssen sie nach Regeln suchen, die es dafür in der Organisation oder im Kopf der Handelnden gibt, oder sie müssen selber welche aufstellen. Wenn Muster nicht den Regeln gehorchen, müssen sie Regeln oder Muster ändern. Schließlich müssen sie aus den Mustern Kerne gewinnen, die diese erzeugen; dazu brauchen sie sicher Hilfe von Psychologen, Semiotikern, Linguisten, Soziologen und anderen Experten. Diese Muster und ihre Kerne sowie Verbindungen zu anderen Mustern in der Organisation beschreiben sie und übergeben die Beschreibungen als Anforderungen den Entwicklern. Die Regeln sollten sie ihnen nicht verraten; sonst werden die programmiert, nach Abschnitt 3 eine Todsünde. Lieber sollten sie die Entwickler beim Aufstellen der Anforderungen beteiligen. – Das ist die *Destruktion*.

Als nächstes sind die Entwickler dran. Sie bringen die Anforderungen auf die Maschine, wobei sie Bezüge nach außen kappen oder durch Interaktion sichern (*Instruktion*). Dabei werden die Muster starr, das kann auch die Beteiligung von Benutzern nicht verhindern.

Das Paradox der Organisationstheorie (Abschnitt 3) kann Informatikern helfen, einzusehen, dass sie mit IT-Systemen menschliche Organisationen nicht abbilden können. Sie können versuchen, die Spielräume, in denen menschliches Zusammenleben sich bewegt, durch Einbau von situationsabhängigen Parametern zu simulieren, so wie Numeriker konvergierende Größen mit Hilfe von Intervallarithmetik berechnen. Im Fall der Numerik kann man sich dem Vorbild beliebig nähern; im Fall der Informatik ist der Unterschied aber ein qualitativer, der sich quantitativ nicht überwinden lässt.

Arno Rolf plädiert daher dafür, dass IT-Gestalter „Formalisierungslücken“ beachten: Bereiche der Organisation, in denen IT-Einsatz vorläufig – weil Technik oder Organisation nicht reif sind – oder grundsätzlich – weil Erstarrung schädlich wäre – vermieden werden muss (l.c., s.o.). Das eröffnet produktive Fragen: Wie kann man den Begriff für Informatiker wie Manager positiv besetzen, etwa „Freiraum“ statt „Lücke“? Wie geht man mit diesen Freiräumen so um, dass sie chaotische Muster nicht zulassen? Kann man absolute von relativen Freiräumen unterscheiden, in denen nur gewisse Arten von Software ausgeschlossen sind? Wie charakterisiert man solche Typen von Freiräumen und von Software?

Das fertige IT-System schließlich fügen Benutzer und Entwickler in die Organisation ein, am besten wieder gemeinsam (*Konstruktion*). Dabei können die Benutzer überprüfen, ob das System ihren Regeln genügt. Trotzdem wird die Anwendung zunächst Schwierigkeiten bereiten, weil die ursprünglichen Muster erstarrt sind und nicht mit den lebendigen Mustern der Umgebung zusammengehen. Die sozialen Muster, die das System in der Anwendung erzeugt, sind i.a. lebendig, aber auf jeden Fall anders als die technischen und anders als die sozialen, die sie ersetzen. IT-Systeme werden von Nutzern als Partner angesehen und behandelt; mit ihren starren Verhaltensmustern verändern sie die Organisation aber radikaler als neue Partner eine Gemeinschaft. Sie bringen technische Sicherheiten, können aber Sicherheiten der Beteiligten (zer)stören und damit Ängste auslösen (s.o. Abschnitt 2&3: Giddens und Crutzen/Hein).

IT-Systeme sind blinde Flecken in der Organisation: Nicht die automatisierten Abläufe, sondern die dadurch ausgelösten Arbeitsmuster reproduzieren die Organisation. Die Beteiligten müssen also im Wechsel Umgebungs- und Systemmuster ändern, bis die Verbindungen laufen. Wo das nicht geht, müssen die Benutzer ihre Anforderungen zurückschrauben oder die Entwickler das System neu konzipieren. Die Konstruktion ist also bei weitem der schwierigste und am wenigsten gesicherte Schritt des langen Weges der IT-Einführung. Sie wird erleichtert, wenn wie bei der allgemeinen Musterbildung, bei der die Phasen des Abbild- und Vorbildseins nicht getrennt sind (Abschnitt 1), die Phasen der Entstehung von IT-Mustern nicht nacheinander, sondern in Wechselwirkung ablaufen; das ist die Idee von Prototyping und zyklischem Vorgehen.

IT-Gestalter und -Nutzer müssen also gemeinsam herausfinden, jedes Mal neu, wie die Destruktion alter und die Konstruktion neuer organisatorischer Strukturen aussehen könnte. Modellierung hilft wie erwähnt nur den Informatikern; für Destruktion und Konstruktion gibt es keine methodische Unterstützung. Es bleibt nur eins: Beide Seiten müssen voneinander lernen, Lehrer wie Schüler sein (Sesink, s.o.); nur so lösen sie Veränderungen aus, bringen Entwicklung in Gang. Für die Wissenschaft Informatik bringt das eine neue Zielsetzung mit

sich, die stärkere interdisziplinäre Zusammenarbeit erfordert (Abschnitt 7); die Umorientierung verlangt Änderungen in der Ausbildung (Abschnitt 9).

Jetzt können wir den in den Abschnitten 3 und 4 aufgespannten theoretischen Rahmen und das Beispiel aus der Praxis von Abschnitt 5 nutzen, um mit Hilfe der komplementären Paare, die die Informatik prägen, ein neues Bild von ihr zu zeichnen.

6. Was ist, was war und was soll die Informatik?

Gegen Ende der 60er Jahre wurde in der Bundesrepublik Deutschland das Überregionale Forschungsprogramm Informatik beschlossen. Für Politik und Industrie versprach die Entwicklung elektronischer Rechenautomaten technischen und wirtschaftlichen Aufschwung: Die wissenschaftliche Disziplin wurde eingerichtet, um Fachkräfte auszubilden, die die neuen Maschinen beherrschen sollten. Den beteiligten Wissenschaftlern ging es mehr um Wissenschaftlichkeit. Einige von ihnen haben in den frühen 70er Jahren ihre Antworten auf die Frage „Was ist Informatik?“ publiziert. Für F.L. Bauer ist die Informatik eine „Ingenieur-Geisteswissenschaft oder, wenn Sie lieber wollen, eine Geistes-Ingenieurwissenschaft“ – wobei das Geistige in Mathematik und Logik liegt –, deren Eigenheit „die Programmierung“ ist, die aus dieser Kreuzung entspringt. Wilfried Brauer definiert die Disziplin als „Wissenschaft von der (vor allem maschinellen) Verarbeitung von Information“; die Informatik ist eine „Strukturwissenschaft“, es geht um Methoden, die Maschine kommt nur in Klammern vor. Im Unterschied zu den beiden Mathematikern definiert der Nachrichtentechniker Günter Giloi die Informatik ganz pragmatisch als eine bunte Sammlung von Fachgebieten, wie sie damals be- und entstanden. Walter Zemanek schließlich, ebenfalls Nachrichtentechniker, knüpft stark an die Ingenieurstradition an, setzt aber die Informatik durch den Vergleich mit der Architektur davon ab: Wie die Architekten sind die Informatiker nicht nur für das technische Funktionieren, sondern ebenso für die gute Verwendbarkeit ihrer Produkte zuständig. Damit meint Zemanek allerdings, wie die meisten Architekten, eher die fachliche als die moralische oder gar die soziale Verantwortung. – Für Details und Literaturangaben siehe (SGI97ff).

In dieser gerafften Geburtsgeschichte der Informatik wird ein ganzes Bündel komplementärer Paare sichtbar: Die drei Hebammen Politik, Wirtschaft und Technik helfen gemeinsam, das Kind ans Licht zu bringen, verfolgen dabei aber (wie auch sonst – sie sind selber paarweise komplementär) durchaus unterschiedliche Interessen. Der Wirtschaft geht es vor allem um die Ausbildung von Fachkräften, der Technik eher um die Erforschung und Entwicklung der neuen Maschinen, der Politik um beides: Die stagnierende Wirtschaft soll angekurbelt, der Technik- und Wissenschaftsstandort Deutschland gesichert werden. Die wissenschaftlichen Paten dagegen wünschen, wie oben zu hören, dem Kind vor allem Wissenschaftlichkeit. Aber

sie kommen selber aus unterschiedlichen Lagern: Entworfen und benutzt haben Rechenautomaten bis dahin vor allem Mathematiker, das technische knowhow liegt eher bei den Elektro- und Nachrichtentechnikern. Drittens schließlich gibt es an Universitäten und Forschungszentren – zumeist innerhalb der Ursprungsdisziplinen – schon Fachgebiete ganz unterschiedlicher Ausrichtung, die in die entstehende Informatik integriert werden sollen. Ein Neuankömmling, der seinen Platz sucht, muss aber wissen, was er will, und kann nicht alles haben wollen.

Gerade diese vielfältigen Spannungsfelder haben – wie nach der Theorie zu erwarten – die Entwicklung des Kindes geprägt. Dass die Informatik als wissenschaftliche Disziplin eingerichtet wurde und nicht bloß zur Ausbildung von Fachkräften, entsprach dem Selbstverständnis der beteiligten Wissenschaftler. Es war aber auch nötig, um sich gegen Einflussnahmen aus Politik, Wirtschaft und Technik abzusichern. Das Ideal reiner Wissenschaftlichkeit stand wiederum in Spannung zur Notwendigkeit eines irdischen Raums dafür: Einerseits musste man auf Eigenständigkeit pochen, um sich von den Mutterdisziplinen absetzen und so ihrem Zugriff zu entziehen. Auch dass nach vielen Diskussionen der Name „Informatik“ statt des pragmatischeren „Computer Science“ oder „Rechentechnik“ gewählt wurde, hat darin seinen Grund. Andererseits war der Bezug zu den Mutterdisziplinen nötig, um den Anspruch auf Wissenschaftlichkeit zu untermauern und in Frieden und mit vernünftiger Ausstattung arbeiten zu können. Auch die Fachgebiete, die manche Wissenschaftler als Mitgift einbrachten, erzeugten Spannung: Sie ermöglichten geregelte Arbeit von Anfang an, erschwerten aber die Profilbildung, die damals wie heute im universitären Feld nötig war. Und ihre bunte Vielfalt förderte noch einen Anspruch auf Universalität, den der universelle Gegenstand Computer, so sehr er verdrängt wurde, nahezulegen schien.

In diesen Spannungsfeldern bewegt sich die Informatik bis heute. Sie wird von öffentlicher Hand, Wirtschaft und Stiftungen finanziert, damit sie die Entwicklung effizienter Computersysteme vorantreibt; aber sie versteht sich ausdrücklich nicht als Computerwissenschaft. Computer sind überall; aber Informatiker vermeiden das Wort und sprechen von Informations- und Kommunikationstechnologie. Die wissenschaftliche Entwicklung wird zunehmend von der technischen und deren Anwendungen bestimmt, wie die Bezeichnung neuer Fachgebiete verrät; aber ob die Wissenschaft die Technik und ihre Verwendung voranbringt, ist zumindest umstritten – mehr als bei anderen Technikbereichen.

Der Computer wird als Werkzeug betrachtet, als persönlicher Assistent oder als intelligentes Gerät, heute meist als (digitales) Medium (Sce97, 07), aber nicht als Maschine. Das rückt die Informatik in die Nähe der unterschiedlichsten Geistes- und Sozialwissenschaften – manche sehen sie als Kulturwissenschaft –, aber entfernt sie von den Ingenieurswissenschaften.

Eine Orientierung auf gemeinsames Ziel hin – etwa „Entwicklung und Verwendung von Computersystemen“ – erscheint undenkbar.

Dementsprechend hat sich das Studium seit Beginn wenig verändert, obwohl die Technik sich so rasant entwickelt. Im Grundstudium werden Grundlagen aus Mathematik, Technik und Programmierung vermittelt, das Hauptstudium besteht aus sich rasch verändernden Spezialgebieten; es gibt weder einen Gegenstand noch Methoden, die Verbindungen und ein Profil des Faches herstellen könnten. Dass die Probleme mit dem Computer heute weniger in der Technik als in deren Verwendung liegen, wird höchstens in der Möglichkeit der Wahl eines Nebenfachs angedeutet. Die Studenten sitzen den ganzen Tag vorm Bildschirm; doch wenn ich von der „Maschine“ rede, wissen sie nicht, was ich meine. Aber für Verwandte und Bekannte sind sie Experten, die man um Hilfe am PC bittet, was sie belustigt bis entrüstet.

Kurz gefasst: Informatiker verändern mit IT immer stärker alle menschlichen Lebensräume, versuchen aber gleichzeitig diesen Einfluss zu verschleiern. Technik darf nicht als Technik auffallen; je menschlicher sie daherkommt, desto besser. Es scheint das höchste Ziel, die Technik immer menschlicher zu machen (Abschnitt 2). Damit stehlen Informatiker sich aber, ohne es zu merken, aus der Verantwortung für die Veränderungen, die sie bewirken.

Ich halte es für besser, die Gegensätze zwischen Mensch und Technik herauszuarbeiten. Nicht um die Technik als unmenschlich oder den Menschen als ungenügend abzuwerten, sondern um in den Unterschieden die Beziehungen besser zu sehen (s. Einleitung). Technik ist die maschinisierte Form regelmäßig ablaufender natürlicher Vorgänge und menschlicher Tätigkeiten. Technik ist also eine Form von Kultur, die bei uns in allen geistigen und sozialen Bereichen angelegt ist – auf unterschiedliche Weise. Wenn wir die Anlagen sehen, können wir eher klären, in welchen Bereichen welcher Einsatz von IT ökologische Entwicklung und menschliches Leben fördert oder behindert.

Es erscheint paradox, dass der Gegenstand Computer immer mächtiger und allgegenwärtiger und gleichzeitig immer unsichtbarer wird. Wird so die Informatik dafür gestraft, dass sie sich von Anfang an von diesem Merkmal ihrer Herkunft distanziert hat? Aber der US-amerikanisch-internationalen Computer Science geht es nicht anders. Sehen wir uns das Verhältnis zum anderen Merkmal an, den Formalismen, die aus der Mathematik stammen. (Zu den zugehörigen Fachgebieten, Technische und Theoretische Informatik, siehe Abschnitt 8.)

Die Informatik ist von Anfang bis heute durch das uralte Spannungsfeld zwischen Mathematik und Technik geprägt. Wolfgang Coy hat in dem schönen Büchlein (Coy85) die Wechselwirkung zwischen beiden Bereichen nachgezeichnet, die schließlich in der Entwicklung von Computern und dann in der Entstehung der Informatik einen neuen Bereich entste-

hen ließ. Das Spezifische der Informatik liegt nicht in der Technik und nicht in der Mathematik, sondern im Programmieren, hieß es oben. Man muss dafür mathematische Formalismen und maschinelle Bedingungen beherrschen, für die Mathematiker bzw. Elektro- und Nachrichtentechniker zuständig sind; aber erst die Zusammenführung beider macht die Wissenschaftlichkeit der Informatik aus. Dass Programmieren eine wissenschaftliche Tätigkeit sei, war nicht selbstverständlich, wie man ihrer langen Entwicklungsgeschichte sieht, die mit den ersten Computern beginnt; für Details und Literatur der folgenden kurzen Darstellung siehe (SGI97ff, auch Sie99ff).

Zunächst schien Programmieren kein Problem, da es nur um numerische Berechnungen ging. Mit den Formularen der Rechenbüros waren die mathematischen Verfahren leicht in Maschinenbefehle zu "kodieren", Computer wurden als direkte Nachbauten menschlicher Rechner an der Tafel oder an der Tischrechenmaschine gesehen (Zuse 1936, von Neumann 1945, Walther 1953). Turings theoretische Maschinen (1936) beeinflussten den Rechnerbau nicht, führten aber bei von Neumann (1948) und ihm selbst (1950) zu den ersten Phantasien von selbstreproduzierenden Automaten bzw. denkenden Maschinen. Bald wurde mehr Formalismus nötig, um die tiefer werdende Kluft zu überbrücken: Flussdiagramme und indirekte Adressierung erlaubten, komplexere Algorithmen effizienter auszuführen (Goldstine, von Neumann 1947/8). Kodierung wurde zur Geheimwissenschaft derer, die mit den Problemen mathematischer Optimierung und den Tücken der Maschine gleich gut umgehen konnten. Mit "automatischer Kodierung" wurde mehr und mehr von dieser Kunst der Maschine übertragen.

Mit der Entwicklung der ersten "Programmiersprachen" wurde schließlich ein eigener Formalismus eingeführt, den die Mathematiker lesen und schreiben und die Maschinen selber in ausführbaren Kode transformieren konnten (Fortran 1957). Schon Algol (1960) war als "rechnerunabhängig" konzipiert, es sollte vor allem die Kommunikation der Mathematiker über "rechnerausführbare" Verfahren erleichtern. Damit wurde die Sprach- von der Rechnerentwicklung getrennt, zunehmend wurden Formalismen aus Mathematik und Logik zur Programmierung und ihrer Unterstützung benutzt. Gleichzeitig traten Bilder neben Wörter als Mittel der "Mensch-Maschine-Kommunikation". So wie der Rechner (als Mensch *und* als Maschine) bei Turing Symbole (und nichts sonst) manipulierte, manipuliert er jetzt Ikons auf dem Bildschirm, um Aktionen auszulösen und zu steuern. Dass er im Hintergrund als Maschine läuft, die programmiert wurde, ist kaum noch bewusst. Visuelle und motorische Schemata werden direkt angeregt, ohne Umweg über das Kognitive; und regen direkt zum Handeln an.

Die Entwicklung kulminiert (vorläufig) in der objektorientierten Programmierung. Der Anwendungsbereich wird durch interagierende Objekte auf dem Bildschirm dargestellt, die

mathematisch beschreibbar sind, aber wie reale Objekte erscheinen. Hinter der graphischen Oberfläche verschwindet das mathematische Werkzeug der abstrakten Datenstrukturen; Programmieren scheint auf das Zeichnen hübscher Bildchen reduziert (Crutzen/Hein l.c. Einleitung; vgl. auch Abschnitt 8). Der Programmierung ergeht es also nicht besser als dem Computer: Sie, die anfangs zum Spezifikum der Disziplin erklärt wurde, entwickelt sich unter der allgemeinen Aufmerksamkeit so stark, dass sie jetzt in den schönen Kleidern (OO) kaum noch zu erkennen ist. Programmieren geht im Design von Oberflächen unter, wird aus einer Wissenschaft zur Kunst, die man sich, wenn überhaupt, nebenbei aneignet. Wozu braucht man da Informatiker?

Informatiker stellen daher in den letzten Jahren die Frage „Was ist Informatik?“ wieder dringlicher (z B. Wih96, Rec00, Des01). Informatiker sind mit schuld daran, dass sich in einer unkritischen Öffentlichkeit, insbesondere in Politik und Verwaltung, die Vorstellung festgesetzt hat, alle Probleme seien mit Computern lösbar. Dadurch wachsen die Anforderungen, mit denen sich Informatiker konfrontiert sehen, schneller als der Fundus von Möglichkeiten, damit umzugehen. Außerdem wird auch Fachfremden deutlicher, dass Erfolg und Misserfolg des Einsatzes von Informatiksystemen stark von der intimen Kenntnis des Einsatzbereichs abhängt (Abschnitt 5). Da Informatiker solche Kenntnis in der Regel weder mitbringen noch erarbeiten können, fordern Praktiker von der Wissenschaft statt immer neuen Bergen von Techniken, Methoden und Wissen zunehmend Ausbildung in „sozialer Kompetenz“. Sie soll Informatiker befähigen, Informatiksysteme zusammen mit den „Betroffenen“ des Einsatzbereichs zu entwickeln. Informatiker sind aber stark technisch sozialisiert, genauso wie Technik stark durch soziale Anforderungen geprägt ist. Man kann „das Soziale“ und „das Technische“ nicht trennen, also kann man es auch nicht getrennt lehren. Soweit Wissenschaft traditionell separiert betrieben wird, können Informatiker daher soziale Kompetenz von Geistes- oder Sozialwissenschaftlern so wenig lernen wie im eigenen Fach; das gilt für Wissenschaftler wie für Studenten. Dasselbe gilt natürlich für technische Kompetenz, auch wenn Informatiker das genauso wenig akzeptieren wie andere Ingenieure. So können sie die traurige Erfahrung machen, dass „Betroffene“ ihnen beim Umgang mit Computern ebenbürtig oder überlegen sind, aber keine Ahnung von den einfachsten Prinzipien des Systementwurfs haben, während ihnen selber, auch wenn sie sich in Kommunikation und Management auskennen, die Kultur des Anwendungsbereichs fremd bleibt.

Aber auch in der eigenen Wissenschaft fühlen Informatiker keinen festen Boden unter den Füßen. Die beiden ältesten Teile, Theoretische und Technische Informatik, sind ihren Ursprungsdisziplinen Mathematik und Nachrichtentechnik methodisch enger verbunden geblie-

ben als der Informatik selbst; zudem ist die diskrete Mathematik der Theoretischen Informatik getrennt von der kontinuierlichen der Ingenieurwissenschaften. Frühere Kernbereiche wie Betriebssysteme und Compilerbau sind längst in einer Vielzahl anderer für spezielle Systemtypen untergegangen, und Softwaretechnik hat sich als zentraler Bereich der Informatik etabliert. Sie ist ohne Bezug zu formalen Methoden und technischen Gegebenheiten nicht zu betreiben; der Wert rein mathematischer oder ingenieurswissenschaftlicher Methoden in der Softwaretechnik ist aber beschränkt, und beide wollen sich nicht zu einem informatischen Bereich zusammenfügen. Ein eigener Kanon von Inhalten und Methoden hat sich nicht herausgebildet.

Noch schlechter sieht es mit der Abgrenzung nach außen aus. Anwendungen nutzen den Computer als Medium, nicht als spezifisches Gerät; sie werden daher methodisch stark vom Anwendungsgebiet bestimmt. Mit jedem Bereich, in den Computer Einzug halten, erhebt sich daher neu die Frage, ob die zuständigen wissenschaftlichen Disziplinen einen entsprechenden Anteil an die Informatik abzugeben oder umgekehrt die Informatik die neu entstehenden Gebiete mit Bezug nach draußen abzuspalten habe; s. Abschnitt 8. Auf die Frage "Was ist Informatik?" zu antworten "Was Informatiker tun" bringt uns also nicht weiter. Ob das, was Informatiker (und andere) tun, Informatik sei, ist zunehmend unklar. Erst recht gibt es kein Einvernehmen über die "eigentlichen" Fragen: Wozu betreiben wir Informatik? Welchen Einfluss haben wir? Was richten wir damit an? Wohin entwickelt sich die Disziplin? Wie können wir die Entwicklung beeinflussen?

7. Informatik als kulturelle Entwicklung

Eine Organisation, die nicht untergehen will, muss Selbsterhaltungskräfte mobilisieren. Denn es gibt selten (oder nur sehr kurz) eine Umgebung, die ihr die Existenz gönnt. Folglich stellt sich auch für die Informatik täglich die Frage, wie sie sich als Institution verteidigen kann. Uns Theoriearbeitern liegt die Erhaltung der Informatik am Herzen – nicht nur, weil wir uns sonst einen neuen Beruf suchen müssten. Wir stehen unserer wissenschaftlichen Heimat teilweise sehr kritisch gegenüber; aber diese Kritik gilt anderen Disziplinen ebenso oder mehr. Manche halten zum Beispiel die Informatik für zu formalistisch oder technikzentriert. Wir können aber dem Schicksal keine Bedingungen stellen, etwa in der Art, dass wir eine Fortdauer der Informatik wünschen, falls sie diese oder jene Bedingungen erfüllt.

Man kann Disziplinen mit Staaten vergleichen. Solange sich alle einig sind, brauchen sie keine Waffen. Aber gegen Bedrohungen von außen oder innen müssen sie sich verteidigen. Und die Informatik ist vielfältig bedroht, heute in der Zeit der Sparzwänge mehr denn je. Der Kampf mit den Nachbardisziplinen um Ressourcen ist härter geworden, teilweise existenzge-

fährdend. In der Außensicht wird Informatik mit Programmieren gleichgesetzt; es scheint also zu genügen, wenn die anderen Disziplinen sich ihre Hausinformatiker halten, die Informatik als Disziplin ist damit überflüssig. Mit dem Hinweis auf Globalisierung wird das Diplom durch Abschlüsse wie Bachelor und Master ersetzt; die Konsequenzen kann man drastisch ausmalen: Die Unterschiede zwischen Universitäten, Technischen Hochschulen und Fachhochschulen werden nivelliert; massenhafte Ausbildung für die Praxis wird gefördert, die wissenschaftliche Ausbildung und damit die Forschung werden an einige (Privat-)Universitäten für die Elite verdrängt. Auf der anderen Seite baut die Wirtschaft in Akademien und Corporate Universities spezifische Studiengänge für den eigenen Bedarf auf und droht damit die staatliche Berufsausbildung in Informatik überflüssig zu machen. Die Lage wäre weniger bedrohlich, wenn die Informatik in sich geschlossen wäre. Wir haben aber gesehen, dass die Informatik weder inhaltlich noch methodisch zu charakterisieren ist. Sie hat keine klaren Grenzen, die sie verteidigen könnte. Die Metapher vom Staat Informatik ist für die Mobilisierung der Selbsterhaltungskräfte ungeeignet.

Wir können stattdessen die Informatik als Kultur betrachten (SGI97ff, Sie99ff; Siefkes in Fre97, Bau01, TdI01,02). Kulturen brauchen einen Lebensbereich, aber nicht notwendig ein eigenes Gebiet; immer leben sie zwischen und mit anderen Kulturen. Wenn sie sich nicht von ihren Nachbarn abgrenzen, gehen sie in ihnen auf; wenn sie sich gegen Angriffe nicht verteidigen, werden sie geschluckt. Aber wenn sie sich abschotten, um ihre Identität zu wahren, erstarren sie; wenn sie andere angreifen, um sich zu bereichern, verlieren sie ihre eigentlichen Ziele. Sie ziehen ihre Stärke und Entwicklungsfähigkeit nicht aus Aggression, sondern aus Eigenständigkeit und Kooperation. Sie gewinnen und erhalten ihre Identität aus Traditionen, die sich in Riten und Mythen, in Sitten und Festen äußern. Diese Traditionen werden mündlich und schriftlich weitergegeben, aber reproduzieren sich vor allem im Gebrauch; ohne praktizierende Menschen werden sie zu Museumsstücken. Daher sind Kulturen regional unterschiedlich ausgeprägt, abhängig von Umgebung und Geschichte; Teilgebiete können sich voneinander stärker unterscheiden als von denen benachbarter Kulturen. So entwickeln sich Kulturen, manchmal schnell, oft überraschend; so bleiben sie überlebensfähig.

In dieser Vorstellung braucht die Informatik einen anerkannten Bestand von Methoden, Inhalten und Lehrgegenständen, um überleben zu können. Diese Kanons sind aber nicht ewig, sondern reproduzieren sich im Gebrauch durch die beteiligten Menschen und verändern sich dabei. Es gibt wie in Kulturen keinen hochrangigen Kern, umgeben von Randgebieten, die weniger wichtig wären. Theoretische und Technische Informatik sind Teilbereiche der Informatik, die ihre Methoden aus den Ursprungsdisziplinen beziehen, aber ihre Identität aus der

Informatik. Löste man sie aus der Informatik heraus, verlören sie Antrieb und Richtung ihrer Entwicklung. Neue Teilgebiete wie wissensbasierte Systeme, Datenschutz, CSCW, Medienpädagogik, Computerlinguistik oder Sozionik sind dagegen anderen Disziplinen wie Psychologie, Jura, Arbeitswissenschaften, Pädagogik, Linguistik oder Soziologie ebenso nahe wie der Informatik. Sie leben aus der Verbindung zu jeweils mindestens zwei Disziplinen und können in verschiedenen Hochschulen der einen oder der anderen zugeordnet sein. Die Informatik entwickelt sich in solchen regionalen Kooperationen nachhaltiger, als durch die globale Akquisition neuer Kolonien; sie ist damit zukunftsfähiger. Auch Gebiete wie Informatikgeschichte, Mensch-Maschine-Kommunikation oder Theorie der Formalisierung können von Informatikern allein genauso wenig betrieben werden wie von Historikern, Psychologen, Mathematikern oder Philosophen. – Mit einigen der erwähnten Gebiete befasse ich mich näher in Abschnitt 8.

In der Diskussion um die Frage „Was ist Informatik?“ werden gern Landkarten von Themen und Fachgebieten entworfen. Dabei gibt es aber selten Einigung: Zuordnung und Abgrenzung, Nähe oder Abstand sind in vielen Fällen strittig, ebenso die wechselseitigen Bezüge. Die Einschätzung hängt von der eigenen Positionierung und von der Blickrichtung – z.B. Vergangenheit, Gegenwart oder Zukunft – ab. Die Unstimmigkeiten können wir lösen, wenn wir von der statischen zu einer dynamischen und von der Staat- zur Kulturmetapher übergehen. Informatik als kulturelle Entwicklung zu betrachten, heißt dreierlei: Die Grenzen der Disziplin nicht als zu befestigende Gräben, sondern als Verkehrsflächen auffassen; die Karte der Gebiete als veränderlich betrachten; und die Grenzgebiete als regional unterschiedlich zugehörig ansehen. Fachgebiete sind dabei keine Besitztümer, die den Wert der Informatik ausmachen. Sie sind generische Einheiten und damit so wichtig, wie sie bereit und in der Lage sind, die Entwicklung zu tragen und zu fördern. Grenzgebiete spielen dabei eine besondere Rolle. Sie ziehen viele junge Leute an, die sich nicht mit „der Informatik“ identifizieren würden. Der intensive Kontakt mit anderen Disziplinen bringt neue Ideen, manchmal Revolutionen. Ohne Gebiete, die „schon immer“ dazugehört haben, gäbe es „die Informatik“ nicht; aber auch sie müssen sich ständig fragen (lassen), was sie zur Entwicklung der Informatik beitragen.

Wenn die Informatik keine klassische Wissenschaft mit klaren Inhalten und Grenzen ist, was ist sie dann? Auf den Theorietagungen (TdI01-03) wurden Alternativen diskutiert: 1) Informatik ist eine Wissenschaft der Moderne. Informatiker vertreten eine universell einsetzbare Technologie; damit gehen sie viel problemloser in andere Gebiete hinein als umgekehrt und helfen, sie zu modernisieren. Aber diese Offenheit ist nur scheinbar. Es geht den Informati-

kern um den technischen Erfolg (das System muss laufen), nicht darum, das Fremde kennenzulernen. So kolonialisieren sie den Teil der anderen Disziplin, in dem sie Erfolg haben; der Rest ist in Gefahr, weil er kein Geld mehr bekommt. Informatiker reiten in die anderen Gebiete ein, soweit ihre Maschinen sie tragen und deren Programme sie weisen. 2) Informatik ist eine postmoderne Wissenschaft mit einer Flickenteppich-Identität. Sie ist nicht durch Inhalte und Methoden zu charakterisieren; man muss die Forschungsschwerpunkte untersuchen, um ein Bild zu bekommen. Aber die Postmoderne hat keine einheitliche Vision, ihre Positionen erscheinen beliebig. „Kulturelle Entwicklung“ heißt aber nicht Beliebigkeit. 3) Informatik ist eine postpostmoderne Wissenschaft. Sie will alle Lebensbereiche computerisieren, befördert die Globalisierung, bezieht daraus ihre wissenschaftliche Identität. Aber Kulturen sind nicht global, sie leben von Kooperation mit *anderen* Kulturen, also von regionaler Verschiedenheit und Vielfalt. Träume von uniformen Weltkulturen sind faschistisch: *Eine* Kultur, Weltanschauung, Religion, Wissenschaft soll alle anderen ersetzen, notfalls mit Gewalt, weil sie als die beste angesehen wird.

Eine Kultur lebt davon, dass es Menschen gibt, die ihr angehören, sich mit ihr identifizieren und diese Identität tragen und weitergeben. So bleibt auch die Informatik nur am Leben, wenn wir Informatiker eine Identität haben. Die speist sich nicht aus einem Kanon von Inhalten und Methoden und davon abgeleiteten Curricula. Wir brauchen eine gemeinsame Aufgabe, an der zu arbeiten uns sinnvoll erscheint und daher Freude macht und die sich von den Aufgaben anderer Disziplinen klar unterscheidet. Weil wir zum Arbeiten Methoden und andere Hilfe brauchen, ergeben sich Kanon und Curricula aus der disziplinären Arbeit, bestimmen ihre Richtung, aber ändern sich dabei, entwickeln sich weiter. Deswegen habe ich Informatik eine kulturelle Entwicklung genannt, nicht bloß eine Kultur.

Wenn wir Computer oder Programme oder computergestützte Systeme als Ziel unserer Arbeit betrachten, reduzieren wir Informatik auf Maschinenkonstruktionen oder Angewandte Mathematische Logik oder eine Kombination beider. Wenn wir den Computer als neutrales Medium betrachten und nur auf die Anwendungen schauen, machen wir Informatik zur Universalwissenschaft, die alle anderen verdrängt oder kolonialisiert. Die *Aufgabe der Informatik* liegt dazwischen: *Entwicklung von Computersystemen mit dem Ziel, Bedingungen im Einsatzbereich zu verbessern*. So haben wir es im Bericht (TIK02) zu meiner AG in (TdI02) formuliert. Dass und wie Informatiker und Benutzer bei der Einführung von IT-Systemen zusammenarbeiten müssen, haben wir in Abschnitt 5 diskutiert. Während der Instruktion haben die Informatiker die Führung, während Destruktion und Konstruktion die Betroffenen. Die Verantwortung dafür, dass sich wirklich etwas bessert wie gewünscht, haben aber immer beide Seiten.

Das gilt nicht nur für Anwendungssysteme. Auch Standardsoftware und Hardware, die im digitalen Medium verschwinden, und mathematische Theorien und formale Methoden der Informatik, die die Betroffenen nicht zu sehen bekommen, dienen letztlich dazu, „bestimmte Bedingungen des Einsatzbereichs zu verbessern“. In allen ihren Tätigkeiten hybridisieren Informatiker menschliche Muster (ihre eigenen wie die realer und gedachter, direkter oder indirekter Benutzer) mit maschinellen. Das dürfen sie weder erdrängen noch vertuschen noch auf andere abwälzen.

Entwickler sollen mit Anwendern oder gar beliebigen Nutzern zusammenarbeiten – das klingt utopisch. Es gibt vielversprechende Ansätze; aber keiner scheint das Problem wirklich zu lösen. Entwickler sind Informatiker oder gelten als solche; sie haben die Autorität einer Wissenschaft hinter sich: Systementwicklung ist ein formaler und technischer Vorgang, den nur die Experten beherrschen. So wie Unterrichten ein pädagogischer Vorgang ist, den die Lehrenden, nicht die Lernenden in der Hand haben. Es hat schon immer Pädagogen gegeben, die dieses fundamentale Prinzip umgestoßen haben; warum nicht auch Informatiker? Wenn Erwartungen und Nutzung für den Erfolg eines Systems ebenso wichtig sind wie die formale und technische Entwicklung, muss die Informatik die Nutzer ebenso unterstützen wie die Entwickler. Das kann sie nicht aus sich heraus; und wenn sie sich die relevanten humanwissenschaftlichen Fachgebiete einverleibt, werden die fruchtbaren Unterschiede nivelliert, und die Informatik wird zum Dinosaurier. Helfen könnte aber die Zusammenarbeit mit Disziplinen wie Soziologie und Psychologe, Semiotik und Pädagogik. Dann könnten Nutzer von IT-Systemen auch mit Experten dieser Art zusammenarbeiten, und das könnte die Verständigung zwischen Nutzern und Entwicklern erleichtern. Eine Theorie der Informatik sollte die Basis für solche Kooperationen mit anderen Disziplinen bilden.

Das ergänzt sich gut mit dem Ansatz von Ralf Klischewski (Kli96), *Anarchie* als ein Leitbild in die Informatik aufzunehmen. Das klingt nach Revolution; das Ziel ist aber gerade, das Ungleichgewicht zwischen Wissenschaftlern und Anwendern aufzuheben, damit sie fruchtbar zusammenarbeiten können. Die Wissenschaft Informatik entwickelt Methoden zur Entwicklung von IT-Systemen, mit deren Hilfe man die Vielfalt organisatorischer Praxis besser unter Kontrolle bringen kann; sie folgt daher dem Leitbild der Beherrschbarkeit (vgl. Abschnitt 3). Anwender müssen aber mit den Methoden frei umgehen können, weil sie zu neuen Situationen nie wirklich passen; aus wissenschaftlicher Sicht verhalten sie sich anarchisch. Informatiker müssen sich der Macht bewusst werden, die sie mit ihren Methoden ausüben (und den Benutzern weitergeben), und den Anwendern möglichst viel Freiraum lassen, also auch Unbeherrschbarkeit positiv fassen. Dazu müssen sie im Diskurs mit den Anwendern lernen, was ihre Methoden bringen und was nicht. Die wissenschaftliche Arbeit wird fruchtbarer und ihre

Ergebnisse werden praxisrelevanter sein, wenn Informatiker den Spielraum zwischen Beherrschbarkeit und Anarchie voll ausschöpfen.

In meiner Sprechweise: Informatiker können Wissenschaftler wie Anwender sein. (Unter Anwendern verstehe ich hier Entwickler, Manager und Benutzer.) Wenn die Informatik Beherrschbarkeit und Anarchie als komplementäres Paar in sich aufnimmt, kann sie eher zwischen beiden Seiten vermitteln und so auch das Paar Entwickler/Nutzer in Bewegung bringen. Auch die Anwender müssen in dem Diskurs lernen: sich vom Methodenzwang der Informatiker nicht beherrschen zu lassen; zu verstehen, was möglich und sinnvoll ist und was nicht. Allerdings darf der Diskurs nicht nur verbal sein, sondern muss Gefühle und Bewertungen einschließen. Dann könnten aus Wissenschaftlern und Anwendern und aus Entwicklern und Benutzern komplementäre Paare werden.

Einen Weg zu solchen Diskursen, die zwischen komplementären Paaren vermitteln können, beschreibt Cecile Crutzen mit der Methode „Dekonstruktion und Konstruktion“, die sie der feministischen Theorie entnimmt (s. auch Abschnitte 2, 3, 5): Zuerst die „Opposition“ entdecken und herausarbeiten; das kann schwierig sein, weil sie so tief in allen kulturellen Mustern steckt, dass wir sie nicht mehr wahrnehmen. Dann die festgefahrene Bewertung der Pole umkehren – nicht nur gedanklich „gut“ und „böse“, „oben“ und „unten“ vertauschen, sondern sehen, dass kein Pol ohne den anderen sein kann. Entwickler von IT-Systemen z.B. benutzen auch: einerseits wissenschaftliche Theorien, Modelle und Methoden, andererseits die Erfahrungen der Benutzer; und Benutzer entwickeln dauernd, nämlich neue Formen des Umgangs mit dem System, womit sie das System selber weiter entwickeln. Ähnliches gilt für Wissenschaftler und Anwender, gefangen zwischen den Polen Beherrschbarkeit und Anarchie; s. oben. Und wenn die Umkehrung beiden Seiten gelungen ist, können sie als Gleichberechtigte im Dialog voneinander lernen. Für Entwickler und Benutzer habe ich das in Anlehnung an Werner Sesink in Abschnitt 5 beschrieben; für die (mediale) Nutzung des Computers betont Heidi Schelhowe, dass erst der Austausch untereinander aus den Erlebnissen Erfahrungen macht (Abschnitt 2); für Wissenschaftler bedeutet das Zusammenarbeit mit Anwendern und mit Vertretern anderer Disziplinen (s. oben und Abschnitt 5). Wie so oft ist die Methode am fruchtbarsten, wenn man die Schritte nicht sklavisch nacheinander ausführt, sondern parallel und im Wechsel, sich gegenseitig verstärkend.

So wie Informatiker bei der praktischen Arbeit mit Benutzern in einem Boot sitzen, müssen sie also wissenschaftlich mit Vertretern anderer Fachgebiete und Disziplinen zusammenarbeiten, zu denen ein Spannungsverhältnis besteht. Andere Ingenieurswissenschaften greifen nicht so tief in menschliche Lebensbereiche ein, sie verändern direkt nur Äußerliches; daher gilt das für sie nicht in demselben Maße. Man vergleiche dazu die Diskussion der Informatik

als "Gestaltungswissenschaft" im Sichtweisenband (Rolf, Siefkes, Volpert in TdI92) und in (Rol98). Natürlich ist die Informatik eine Ingenieurs-, keine Sozial-, Kultur- oder gar Geisteswissenschaft. Computer sind Maschinen, auch wenn man sie Medium oder Geisttechnik nennt. Aber „Verbesserung“ schließt Kulturelles ein und verlangt, dass die Informatik sich in ganz anderer Weise als klassische Ingenieurswissenschaften als komplementär und nicht nur als gegensätzlich zu anderen Disziplinen versteht. Vielleicht könnte ihr eines Tages gar die Rolle der Mittlerin zwischen Ingenieurs- und Kulturwissenschaften zufallen?

Aus der Metapher „Informatik als kulturelle Entwicklung“ ist mit der so formulierten Aufgabe eine Bestimmung der Disziplin geworden. Die Informatik entwickelt sich nicht nur *wie* eine Kultur. In die Entwicklung von Informatiksystemen fließen (über Hybridisierung) dauernd die Eigenheiten der Anwendungsbereiche ein, also unserer ganzen Kultur, ebenso wie wir mit unseren Systemen die ganze Kultur beeinflussen. Informatik entwickelt sich als Teil unserer Kultur und mit dieser, sie ist – gewollt oder ungewollt – prominenter Teil der kulturellen Entwicklung.

Computer und Informatik sind das (vorläufige) Ergebnis der wechselseitigen Entwicklung von Mathematik und Technik oder allgemeiner von rationalem Denken über und technischem Umgang mit Mensch und Natur (Coy85; Abschnitte 2, 6). Bei einem Workshop „Creative Writing“ habe ich eine Informatikerin unter Tränen sagen hören, sie könne keine Texte mehr schreiben, nur Listen führen. Andere Informatiker stellen fest, dass sie in Entscheidungsbäumen statt in „Geschichten“ denken, in workflows und Geschäftsprozessen statt in arbeitenden Menschen, d.h. in logischen statt in inhaltlichen Zusammenhängen. Die tägliche Beschäftigung mit Denk- und Arbeitsformen lässt immer besser damit umgehen, verstärkt entsprechende Schemata (Abschnitt 3). Umgekehrt bauen Informatiker diese Formen verstärkt in ihre Artefakte ein, in ihre Systeme und Theorien. „IT-technology-in-use“ (Orl00, Abschnitt 3) bildet einen sich selbst verstärkenden Zirkel mit „IT-ideology-in-use“. Denselben Einfluss hat IT aber auf die Denk- und Arbeitsformen der Benutzer, also auf die ganze Kultur. Und als aus dem kulturellen Hintergrund kommende Selbstverständlichkeiten – „Orientierungsmuster“ haben wir sie, soziologischer Terminologie folgend, in dem Sozialgeschichteprojekt genannt (SGI99ff) – fließen sie wieder ein in die informatische Arbeit. Natürlich haben Informatiker diesen circulus vitiosus (oder benevolens) nicht in Gang gesetzt. Sie schwimmen aber auch nicht nur mit im Strom der Wechselwirkungen zwischen formalem Denken und technischem Tun, sondern verbreiten, vertiefen, beschleunigen ihn gewaltig. So konkret ist Informatik kulturelle Entwicklung: Als Teil der Kultur entwickelt sie sich in ihr und mit ihr.

Was ich gerade über informatische Formalismen gesagt habe, gilt ebenso für Werkzeuge und Methoden wie UML, EPK, ERP, Petrinetze, MS-Office oder PowerPoint. Kundige behaupten, „PowerPoint revolutioniere das Management“, da Projektanträge und -berichte und andere Texte nur noch als PP-Folien durch die Hierarchien auf und ab getragen werden. Kann man Ähnliches für andere Organisationsmittel sagen? Wie verändert sich unser Verhältnis zur Wirklichkeit dadurch, dass man mit Computern „alles“ simulieren kann? Ist Simulation nicht längst Konstruktion von „Wirklichkeit“ geworden? (Workshop „Hyperkult 15: Modelling and Simulation“, Lüneburg 2006.)

8. Informatik – ihre Gebiete und Nachbarn

Hybridisierung ist kein Fluch, der auf der Informatik lastet, und keine Zauberkunst, die sie auszeichnet, sondern eine Herausforderung, die Informatik auf besondere Weise auf disziplinäre und interdisziplinäre Zusammenarbeit angewiesen sein lässt. Da im Umgang mit dem Computer gegensätzliche Welten hybridisiert werden, müssen in der wissenschaftlichen Arbeit unterschiedliche Gebiete zusammenkommen, insbesondere geistes- und sozialwissenschaftliche mit ingenieur- und naturwissenschaftlichen. Das verlangt gleichzeitig Offenheit, Bescheidenheit und kritische Distanz aller Beteiligten.

Informatiker müssen menschliche Bereiche mit dem Computer hybridisieren, aber nicht unterschiedliche wissenschaftliche Arbeitsweisen miteinander. Eine Theorie soll zwischen komplementären Paaren vermitteln und so Gegensätzliches füreinander fruchtbar machen. Diese Balance gelingt nur in der gemeinsamen Arbeit. Wenn die Beteiligten lernen, die Erfahrungen der anderen für sich nutzbar zu machen, können sich die unterschiedlichen Menschen und ihre Welten gemeinsam entwickeln. Sicher wird es dann weniger große Projekte geben, die die Welt ändern und unsere Karriere und die Disziplin voran bringen. Aber wir können kleinere durchführen, die wirklich helfen (vgl. den Schluss des meines Beitrages in TdI92). So kommen wir der im letzten Abschnitt formulierten Aufgabe der Informatik nach, Computersystemen mit dem Ziel zu entwickeln, Bedingungen im Einsatzbereich zu verbessern.

Die erste Aufgabe einer Theorie der Informatik ist also, die Fachgebiete der Informatik daraufhin zu befragen, wie sie dieser Aufgabe nachkommen. In der Einleitung habe ich ‚Theorie’ als eine Position außerhalb definiert, die von innen heraus getragen wird. Das Vorgehen wird also nur fruchtbar, wenn Mitglieder die Fragestellung in ihr Fachgebiet einbringen. Da die Fachgebiete ganz unterschiedliche Teile der Aufgabe berühren, wird zunächst eine Sammlung unterschiedlicher Teiltheorien entstehen. Eine Theorie der Informatik ist aber mehr als eine solche Sammlung. So wie die drei Phasen im Zyklus der Entwicklung und Verwendung von IT (Abschnitt 5) und die drei Schritte der Dekonstruktion von Gegensätzen (Abschnitt 7)

ihre Bedeutung erst als Teile im Zusammenhang und durch ihre wechselseitigen Abhängigkeiten erhalten, erhalten die zugehörigen Teiltheorien ihre Bedeutung erst durch die Beziehungen zueinander und zu einer ganzen Theorie der Informatik. Theoriearbeit besteht also weiterhin darin, die schwierigen, oft widerstreitenden oder gar widersprüchlichen Beziehungen zwischen den befragten Fachgebieten untereinander und mit Nachbardisziplinen sowie mit neu entstehenden Gebieten herauszuarbeiten. Die Beziehungen können als Kooperation oder Konflikt, als Unterstützung oder Konkurrenz zu Tage treten und müssen als solche aufgenommen werden. Es geht nicht darum, das Verbindende in den unterschiedlichen Gebieten zu finden, sie auf ein gemeinsames Ziel festzulegen oder gar anzugleichen. Die Kreativität einer Gruppe speist sich aus ihrer Vielfalt (Bro79); aber dazu muss sie sich als Gruppe wahrnehmen. Aufgabe einer Theorie der Informatik ist es, die Einheit in dieser Vielfalt deutlich zu machen. Sehen wir uns einige Fachgebiete und Disziplinen an, die dazu beitragen können!

Wir beginnen mit der *Theoretischen Informatik*. Sie liefert mathematische Theorien und Modelle fürs Spezifizieren und Berechnen und für den Computer, also für die drei Schritte Formalisierung, Algorithmisierung, Maschinisierung, und könnte daher mathematischer Teil einer allgemeinen Theorie der Informatik sein. Sie unterscheidet sich von entsprechenden Gebieten der Mathematik, soweit sie „Instruktion“ als eingebettet in Schritte von „Destruktion“ und „Konstruktion“ sieht (Abschnitt 5). Solche Untersuchungen können die Auswahl von Forschungsgegenständen und -methoden und damit die Forschungsrichtung der Theoretischen Informatik beeinflussen, so wie der Wunsch nach konkreten Ergebnissen von analytischen Lösungen zu numerischen Verfahren führt.

Zum Beispiel unterscheiden sich endliche Automaten und Turingmaschinen nicht nur in der Berechnungskraft; man kann mit ihnen unterschiedliche Berechnungsvorgänge und damit unterschiedliche Teile oder Aspekte von Rechnern modellieren. So werden sie in Softwaretechnik und Technischer Informatik schon lange benutzt, auch wenn sie historisch nicht dafür entstanden sind. Ähnlich kann man Algorithmen nicht nur auf ihre Effizienz und Komplexität hin untersuchen und klassifizieren, sondern ebenso daraufhin, welchen menschlichen Bedürfnissen sie entgegenkommen, in welche organisatorische Rahmen sie passen, welche Arbeitsformen sie fördern oder behindern; das letztere meint weit mehr als Verständlichkeit oder „Benutzerfreundlichkeit“. Ebenso kann man die Semantik von Berechnungsformalismen vom menschlichen Umgang her oder auf die maschinelle Ausführung hin definieren; auch wenn die Formalismen mathematisch äquivalent sind, fördern sie ganz verschiedene Weisen des Formalisierens. Die jeweils gegenübergestellten Vorgehensweisen kann man als widersprüchlich ansehen; sie können sich aber gegenseitig befruchten, wenn man sie als komplementäre

Paare angeht. Man kann solche ungewohnten Aufgaben als Herausforderung statt als lästige Forderung verstehen; vgl. ‚Theorie vs. Ethik' in Abschnitt 9.

„Objektorientierung" z.B. ist entstanden, als man entdeckte, dass das mächtige mathematische Werkzeug der abstrakten Datenstrukturen aus der Universellen Algebra ebenso leicht anschaulich wie lauffähig zu machen ist. Durch diese Verknüpfung verschiedener Fachsichten ist der Ansatz so erfolgreich in der Programmierung und in anderen Gebieten geworden. Aus Sicht der Theorie der Informatik erleichtern allerdings die graphischen Darstellungen nicht nur das Verständnis für Informatiker wie Anwender, sondern verschleiern auch die erreichte Abstraktion und die zugrunde liegenden Abhängigkeiten (siehe Crutzen/Hein l.c. Einleitung; vgl. auch Abschnitt 6). Hängt das mit dem mathematischen Formalismus zusammen? Was würde sich ändern, wenn man andere Spezifikationstechniken verwendete?

Allgemein kann man fragen: Welche Formalismen sind zum Formalisieren welcher Beschreibungen welcher Situationen geeignet? Um mit der Frage umgehen zu können, müssen Informatiker nicht nur Formalismen kennen, sondern Formalisieren lernen und die Folgen dessen verstehen. Eine solche Orientierung an Problemen der Informatik würde der Theoretischen Informatik ein besseres Heimatrecht in der Informatik verschaffen, ohne dass sie ihre mathematische Identität leugnen müsste, und würde sie in eine allgemeine Theorie der Informatik einbetten.

Für die *Technische Informatik* trifft dasselbe in anderer Form zu. Man kann den Computer als technisches Gerät auffassen, höchstens die direkten „Schnittstellen" zur Software als Steuerungsmittel und zum Einsatzbereich als Operationsgebiet in den Blick nehmen. Man kann Maschinisierung aber auch als Teil der „Instruktion" behandeln (Abschnitt 5). Dann wird deutlich, dass Implementierung die vorhergehende Modellierung reziprok widerspiegelt, die damit verbundenen Änderungen aber nicht aufhebt, sondern im Gegenteil in Realität umsetzt. Wenn man den Blick dann noch auf den ganzen Prozess der Entwicklung und des Einsatzes von Computersystemen erweitert, sieht man, wie die technische Arbeit an Prozessen gesellschaftlicher Veränderung Teil hat. Wie oben kann das technische Vorgehen mit den erweiterten in fruchtbarer Wechselwirkung stehen.

Softwaretechnik ist ein, wenn nicht *das* zentrale Gebiet der Informatik. Zu einer Theorie der Informatik könnte sie beitragen, wenn sie Fragestellungen einbezieht wie (Flo92, Grü00, Bah07, G&R07, Brödner, Seim, Wohland in TdI02ff): Wie kann man lebendige Muster so programmieren, dass die vom IT-System erzeugten starren wieder ähnliche oder wunschgemäß veränderte erzeugen? Wie kann man erreichen, dass Menschen durch solches Verändern von Mustern gefördert und nicht geschädigt werden? Nach welchen (nicht nur technischen) Maßstäben bewerten wir die Qualität von Software? Welche anderen können wir wählen?

Welche Rolle spielen Emotionen in Softwareprojekten? Welche Arten von Arbeitsvorgängen können durch welche Formen von Software(gestaltung) unterstützt oder ersetzt werden, welche werden eher behindert? Was ist der Grund für Softwarehavarien? Wie hängt die Gestaltung von Software mit der Gestaltung von Organisationen durch IT-Einsatz zusammen? Für welche Aspekte unserer Arbeit als Softwaretechniker können wir die Verantwortung übernehmen? Damit könnten auch die Beziehungen zur Softwareergonomie enger werden.

Im Fachgebiet *Informatik und Gesellschaft (I&G)* geht es um Wechselwirkungen zwischen beiden Bereichen; vgl. auch Abschnitt 9. Sieht man die Bereiche als statisch, reduzieren sich die Wechselwirkungen auf Kämpfe zwischen politisch-wirtschaftlichen und technisch-wissenschaftlichen Ideologien. Damit kommt man zu Analysen, die zutreffen mögen, aber nicht zur Entwicklung der Informatik beitragen. Z.B.: Im Kapitalismus wird IT zur Organisation von Geld- und Güterströmen gebraucht; IT ersetzt aber menschliche Arbeit, zerstört damit letztlich den Staat. Informatiker schaufeln also an vielen Gräbern, oft am eigenen. Man kann spekulieren, ob IT in anderen Kulturen hätte entstehen können. Ansätze wie Freie Software und Ökonux versuchen, zwischen den Gegensätzen zu vermitteln: Freude und Kritik an der Technik zu vereinbaren, die Vorzüge auszunutzen ohne die Nachteile in Kauf zu nehmen. Im Sozialismus wird IT zur Organisation der Planwirtschaft gebraucht; damit würgt IT menschliche Freiheit und Initiative ab, zerstört Wirtschaft und Privatleben. In der DDR und der SU hat der Staat aus Angst vor zu viel Freiheit und Einflüssen aus dem Westen die Entwicklung von IT be- bis verhindert. Vermittlungsversuche zwischen IT und sozialistischer Gesellschaft hat es daher wohl nicht gegeben. – Solche Analysen der „Gesellschaftlichen Implikationen der Informatik“ (ohne die Aspekte ‚Wechselwirkung’ und ‚Vermittlung’) haben dazu beigetragen, dass I&G als Fachgebiet in der Informatik durchgesetzt wurde, haben ihm aber in der Disziplin einen schlechten Ruf eingebracht. Bei vielen Informatikern wird es als "Laberfach" abgetan: I&G-ler reden über unser Fach (oder kritisieren es gar), statt darin zu arbeiten.

An die Stelle einer solchen allgemeinen Ausrichtung von I&G sind daher heute spezifische Forschungsfragen getreten (Sci95, Rol07). Ein Schwerpunkt ist derzeit die Verwendung des Computers als Medium (Sce97) in allen Bereichen der Kommunikation und Kooperation: Wie verändern sich Formen der Lehre, der wissenschaftlichen Zusammenarbeit und allgemein der Hochschule durch die Verwendung des Computers als Lehr- und Lernmedium? Wie beeinflusst Computerarbeit die menschliche Psyche, und welche Forderungen z.B. für Schnittstellen und Arbeitsverhältnisse ergeben sich daraus? Gibt es geistiges Eigentum noch und was ist es wert, wenn es, einmal in elektronischer Form, unbeschränkt und fast kostenfrei zu verteilen ist? Bei solchen Untersuchungen wird IT meist als gegeben angenommen, ihre Ent-

wicklung durch Informatiker nicht analysiert. (Siehe aber die Bemerkungen weiter unten zu den neuen Gebieten Kognitionswissenschaft, Medienpädagogik und CSCW.) Lohnend wäre eine stärkere Zusammenarbeit mit „Praktikern“, bei der diese Entwicklung ausgehend von Problemen des Einsatzes von IT in den verschiedenen Anwendungsgebieten untersucht wird (Abschnitt 5). Wie erwähnt, wird auch dabei meist übersehen, dass der Weg von der Gesellschaft zur IT nur übers Formale geht, dass wir formalisieren und algorithmisieren müssen, um Mensch und Maschine zu hybridisieren (Abschnitt 2). Es hätte erhebliche Folgen für das Fachgebiet – die vielleicht nicht alle als positiv ansähen –, wenn der formale Anteil der Informatik, der Mensch und Maschine und so Informatik und Gesellschaft verknüpft, stärker einbezogen würde.

I&G ist dasjenige Fachgebiet der Informatik, das am stärksten mit der hier betrachteten Theorie in Wechselwirkung steht. Unser Unternehmen ist durch starke Impulse aus I&G ins Leben gerufen worden, und viele „Theoriearbeiter“ sind ihm in irgendeiner Form verbunden, sehen es allerdings gerade daher oft kritisch. Eine Umorientierung des Fachgebietes in Richtung „Theorie der Informatik“ könnte ihm ein klareres Profil geben und seinen Einfluss auf die Informatik verstärken. Eine Umbenennung von I&G in „Informatik als kulturelles Projekt“ wäre radikal, würde aber betonen, dass Informatik und Gesellschaft in einem dynamischen Verhältnis, in rasanter Wechselwirkung stehen. IT wird gern als "Spinne im Netz" gesehen – ein Bild, in dem wir Ängste vor den modernen Informations- und Kommunikationstechniken mit dem Glauben an ihre Macht mischen. Tatsächlich sind die Abhängigkeiten und Einflussmöglichkeiten vielfältiger, die Spinne reagiert auf das Zittern der Fäden ebenso empfindlich wie ihre Opfer, wir alle sind Teil des kulturellen Netzes. Ein Fachgebiet I&G mit der Spinne im Wappen könnte zu einem Zentrum informatischer Reflexion werden, deren Fäden die Disziplin zusammenhalten. Es würde mehr zu einer Theorie der Informatik beitragen.

Die meisten dieser und anderer Fachgebiete weisen mehr oder weniger enge Beziehungen zu Nachbardisziplinen oder -gebieten der Informatik auf. Solche Verknüpfungen habe ich im Verlauf der Arbeit immer wieder explizit oder implizit angesprochen (Abschnittsnummern in Klammern): Mathematik (0,6), Nachrichtentechnik (6), Psychologie (3), Soziologie (3,4), Organisationstheorie (3,5), Semiotik (3), Pädagogik (Einleitung, 2,3,5). Wenn man Informatik und solche Nachbardisziplinen als komplementär auffasst, kann das die Entwicklung beider fördern; betont man die Gegensätze oder verflacht sie durch Angleichung, schadet man beiden. Das ist eine zentrale Aussage der Theorie der Informatik, die hier entwickelt wird (Einleitung, Abschnitt 7). Ich betrachte jetzt einige Gebiete, die erst im Rahmen solcher Verknüpfungen entstanden sind.

Vorstellungen, dass Rechenmaschinen und menschliches Gehirn sich in irgendeiner Weise ähnlich sind, sind uralt – viel älter als der Computer, sogar älter als die Psychologie (vgl. Schmidt-Brücken in SGI98b). Sie haben wesentlichen Einfluss auf die Entwicklung des Computers gehabt (SGI01) und in beiden Disziplinen, Informatik und Psychologie, eigene Fachgebiete hervorgebracht. Im Gebiet *Künstliche Intelligenz (KI)* versucht man, menschliches Denken in Computern und Programmen nachzubilden; vgl. Ses04, Sik07. In der *Kognitionswissenschaft (KW)* benutzt man umgekehrt den Computer als Modell fürs Denken und Lernen und entwickelt entsprechende Theorien. Als Fachgebiete liefern sie ihren Heimatdisziplinen Modelle und Theorien für spezielle technische Vorhaben bzw. psychische Vorgänge. Darüber hinaus können sie gerade durch ihre Orientierung nach außen ganz neue Entwicklungen anstoßen; für KW siehe dazu das schöne Buch (VTR91), für KI die Literaturgattung Science Fiction, die allerdings die KI stärker beeinflusst hat als umgekehrt. Aber Versuche, die neuen Gebiete zu ihren Ursprüngen zurückzuholen, also KW für die Analyse von IT-Problemen und KI für die Untersuchung psychischer Phänomene einzusetzen, halte ich für sinnlos und bedenklich; das spezielle Werkzeug gibt nichts dafür her, und die darauf beschränkte Disziplin verliert ihren Wert als Partner der jeweils anderen. Die Entwicklungspsychologie und mit anderer Akzentuierung die Tätigkeitstheorie liefern andere Modelle fürs Denken und Lernen, die ich in Abschnitt 3 als Wechselwirkung zwischen geistigen und körperlichen Schemata und Phänomenen beschrieben habe. Sie machen die Vorteile und Probleme, die der Umgang mit Computern mit sich bringt, besser verständlich (Grü00; zum Lernen vgl. auch Sce07, Ses01, 04).

Ganz Entsprechendes gilt für das Gebiet *Sozionik*, das derzeit zwischen Informatik und Soziologie entsteht. Kooperation mit Soziologen ist unerlässlich für die Erforschung von Entwicklung und Einsatz von IT-Technologie (Abschnitte 5 und 7). Aber umgekehrt Internetagentensysteme zur Untersuchung sozialen Verhaltens zu benutzen, erscheint mir höchst fragwürdig. Auch beim „Methodentransfer“ von der Informatik in die Soziologie (Valk in TdI02, V&M07) habe ich Bedenken. Zugegeben, Argumentationen von Geistes- und Sozialwissenschaftlern erscheinen Mathematikern und Naturwissenschaftlern oft diffus. Aber klares methodisches Denken ist nur von Vorteil, wenn der Gegenstand klar strukturiert werden kann und die Methoden dafür geeignet sind. Soziale Situationen werden arg vereinfacht, wenn man sie mit Hilfe von Petrinetzen zu klären versucht. Fruchtbarer erscheint mir, die Spannungsfelder zwischen beiden Bereichen als komplementär zu nutzen (Abschnitte 2, 5, 7).

Anders sieht es bei den folgenden neuen Gebieten aus: Aus dem Interesse von Linguisten und Informatikern an Methoden und Techniken der jeweils anderen Disziplin ist die *Computerlinguistik* als eigenständiges Gebiet entstanden, das ohne intensive Beziehungen zu den

Heimatdisziplinen nicht gedeihen kann. Institutionell kann sie da oder dort angesiedelt sein und davon unabhängig wichtige Beiträge für beide Disziplinen und für eine Theorie der Informatik liefern. Die *Medienpädagogik* entstand, als IT-Systeme zunehmend als Medien der Wissensvermittlung eingesetzt wurden. Es stellte sich bald heraus, dass solcher Einsatz höchst fragwürdig ist, wenn er nicht von informatischer *und* pädagogischer Kompetenz getragen wird. Die resultierende enge Zusammenarbeit und intensive Beschäftigung mit der jeweils anderen Disziplin kann zu einem ganz neuen Verständnis der eigenen führen (Wik00, Sci01, Ses04, Sce07, Sesink in TdI03, 07, Wilkens in TdI02, 03; s. Abschnitte 2,3,5). Mit ganz anderer Zielsetzung ist das Gebiet *Computer Supported Cooperative Work (CSCW)* zwischen den Disziplinen Informatik, Soziologie, Psychologie und Pädagogik entstanden und entwickelt sich eigenständig.

Wieder anders steht es mit Untersuchungen über Technik allgemein, die man auf die Informatik anwenden will. Technikphilosophie, -soziologie und -geschichte sind in den jeweiligen Disziplinen lange etabliert, aber eine Philosophie, Soziologie oder Geschichte der Informatik wird es als Fachgebiet der Informatik kaum geben, obwohl es teilweise intensive Bemühungen um sie gibt (zur *Philosophie der Informatik* siehe Hes03, Hesse in TdI01 und Bau01; zur Geschichte Hel03 und den nächsten Absatz). Und entsprechende Theorien *über* die Informatik blieben für die Informatik ziemlich folgenlos (s. Einleitung). Denn für Maschinen und Formalismen gilt in besonderem Maß, was allgemein fürs Lernen anerkannt ist: Verstehen erwächst nicht allein aus dem Umgang mit der Sache oder der Reflexion über sie, sondern nur aus der Wechselwirkung von beidem. Entsprechende Theorien *der* Informatik werden daher am ersten von Informatikern zusammen mit Vertretern der anderen Disziplinen erarbeitet werden. Nur dann können sie als Teile einer allgemeinen Theorie der Informatik unsere Disziplin befruchten.

Als Beispiel betrachte ich die *Geschichte der Informatik*. Die Publikationen (SGI97ff, Sie99ff, Siefkes in Fre97, Bau01, TdI01ff) zeigen, wie Informatiker in Zusammenarbeit mit Historikern eine Sozial- und Kulturgeschichte der Informatik schreiben können, die wesentlich zur Theorie der Informatik beiträgt. Historiker ohne Erfahrung in der Informatik können Informatik-Quellen nicht auf diese Weise auswerten. Die Art der Darstellung stößt daher bei vielen Historikern auf dasselbe Unverständnis wie bei Informatikern mit einer klassischen Vorstellung von Geschichte.

Man kann die Geschichte der Informatik als eine Allee zeichnen, gesäumt von den Denkmälern ihrer Pioniere und Artefakte – Geistmaschinen, Programmiersprachen, Formalismen. Weiter führt es, wenn man die Denkmäler und die Texte, auf denen sie stehen, auf die „Orien-

tierungsmuster“ hin (s.o. Abschnitt 7) befragt, die Informatiker bei ihrer Arbeit geleitet und zu den Denkmälern geführt haben. Aus den Geschichten, die man so findet und erfindet, erwächst eine Geschichte, die nicht nur die kulturellen Wurzeln der Informatik zeigt und so Anlass zu neuen Geschichten gibt, sondern auch die kulturellen Blüten und Früchte, die sie treibt und trägt. Eine solche Geschichte gibt Anlass zu neuen Geschichten und spinnt sich so fort. Und sie lässt die Gegensätze erkennen, an denen Informatiker sich immer wieder gerieben haben, sowie die Entscheidungen, die gefallen sind, zum Guten oder Bösen. Sie zeigt „Informatik als kulturelle Entwicklung“ (Abschnitt 7): Wo wir herkommen, wo wir jetzt stehen und wohin es gehen könnte und sollte. So ist sie Teil einer Theorie der Informatik, die Wissenschaft mit Ethik wie mit Praxis vermittelt und auch so Informatiker bei ihrer Arbeit unterstützt.

Als letztes Beispiel interdisziplinärer Verflechtungen, in denen aus der hier entwickelten Sicht die Informatik hängt, betrachte ich ein Gebiet, das auf ersten Blick keinerlei Beziehungen zur Informatik erkennen lässt. *Geschlechterforschung* ist inzwischen ein anerkanntes wissenschaftliches Feld, meist irgendwo zwischen den Sozial- und Geisteswissenschaften angesiedelt. Seine Bedeutung für die Informatik leuchtet auf, wenn wir an den letzten Abschnitt zurückdenken: Informatik als kulturelle Entwicklung. Unsere Disziplin entwickelt sich nicht nur wie eine Kultur, sondern in ihr, mit ihr verquickt in teils erquickender, teils unerquicklicher Wechselwirkung. Besonders tief in jeder Kultur sind aber die Beziehungen der Geschlechter verankert. Geschlechterverhältnisse sind nicht einfach natur- oder gottgegeben, sondern beruhen auf „Zuschreibungen“ – Vorstellungen von Frauen und Männern, die sich mit der Kultur, wenn auch langsam, verändern. Ohne solche Zuschreibungen gäbe es keine Kultur. Je bewusster wir uns ihrer sind, desto besser verstehen wir ihre Entwicklung.

Wenn uns Informatik und Geschlechterverhältnisse am Herzen liegen, sollten wir also nach geschlechtlichen Zuschreibungen in der Informatik und nach informatischen Zuschreibungen in den Geschlechterverhältnissen suchen und ihre Beziehungen aufklären. Es gibt keine männliche und weibliche Informatik, der wir das Geschlecht austreiben müssten – so wenig, wie wir Männern oder Frauen ihre Liebe zur Informatik aus- oder eintreiben müssen. Aber die Informatik konserviert kulturelle Zuschreibungen, je mehr wir unsere Augen davor schließen. Und unbewusste Einschreibungen konservieren die Informatik.

Zum Beispiel werden in unserer Kultur Technik und Gesetze als männlich und überlegen angesehen, die soziale Sphäre mit ihren ungeregelten Beziehungen als weiblich und untergeordnet. Mit der unbewussten geschlechtlichen Konnotation machen wir aus Softwareentwicklung einen Kriegsfeldzug (Klischewski, Abschnitt 7): Informatiker besiegen die nach Hilfe rufende und gleichwohl widerständige Praxis, wie Männer die hilflosen und gleichwohl wi-

derspenstigen Frauen bezwingen. Sehen wir Frauen und Männer als Partner an, die mit ihren Stärken und Schwächen aufeinander angewiesen sind, verstehen wir auch die Wechselwirkungen zwischen Informatiksystemen und sozialen Situationen besser, können besser mit ihnen umgehen. Soft skills, die z.B. für Partizipation und Teamarbeit in der Systementwicklung benötigt werden, sind dann nicht "wei(bli)che" Fähigkeiten, sondern selbstverständliche Grundlage und Ausdruck einer fruchtbaren Arbeitskultur.

Geschlechterforschung ist also nichts anderes als eine Form von Theoriearbeit mit einem besonderen Anliegen. Auf die Informatik angewendet hat sie bei dem Unternehmen ‚Theorie der Informatik' eine wichtige Rolle gespielt; siehe die Arbeitsgruppe „Kulturelle und Geschlechterperspektiven auf die Informatik" von Corinna Bath und Jutta Weber in (TdI02) und die Arbeiten (Bah07, BSW07), die aus solchen Aktivitäten hervorgegangen sind, sowie (Sie94, 99b). Cecile Crutzen hat das Verfahren der Dekonstruktion von Gegensätzen aus der feministischen Theorie übernommen und auf Softwareentwicklung angewendet (Abschnitte 2, 3, 5, 7; so ist sie zur Theoriearbeit gekommen (TdI02). Wie bei Geschichte und Philosophie (und bei der Theorie selbst) halte ich es für besser, Geschlechterforschung aus informatischer Arbeit heraus und wieder zurück zu betreiben, damit Anliegen und Ergebnisse relevant für die Disziplin werden. Ein Fachgebiet „Geschlecht der Informatik" wäre der Sache nicht dienlich.

9. Theorie der Informatik zwischen Stühlen

Haben wir damit den vier Sichtweisen des Bandes (TdI92) eine fünfte hinzugefügt oder gar bloß die Sichtweise „Kultur" von dort übernommen? Der Titel „Sichtweisen der Informatik" wird gern als Eingeständnis einer Niederlage angesehen: Da sind ein paar, die mit der Informatik unzufrieden waren, aufgebrochen, um eine Theorie zu finden, die zeigt, was Informatik wirklich ist oder sein sollte. Weil sie die Theorie nicht finden konnten, haben sie sich damit begnügt, die Aussichtstürme zu beschreiben, auf die sie bei ihrer Suche gestiegen sind.

Diese Einschätzung resultiert aus einer bestimmten Vorstellung von Theorie: Die Theorie eines Gegenstandes ist eine Beschreibung, die den Gegenstand erklärt. Eine Theorie besteht also aus Aussagen *über* den Gegenstand. Der Gegenstand liegt vor uns, die Theorie steht *darüber*, wohl getrennt. Deswegen kommen Theorien von oben: für das Leben aus der Philosophie, für das Sterben aus der Religion, für das Glück aus der Astrologie. Für die Informatik aus der Mathematik.

Eine Theorie ist der Blick von einer Position außerhalb, habe ich selber in der Einleitung gesagt. Aber diese Position muss von innen heraus aufgebaut werden – also für die Informatik nicht aus der Mathematik –, und sie muss zwischen Gegensätzen vermitteln. So verlangt die in Abschnitt 7 formulierte Aufgabe der Informatik theoretische Arbeit: Der Einsatzbereich ist

nicht die Informatik selbst, also müssen wir uns aus ihr herausbewegen. Auch die Kriterien für „besser – schlechter“ dürfen nicht nur informatische sein; sie müssen aus der Praxis, aus der Ethik, aus der Geschichte kommen – um nur drei zu nennen. Aber Blick auf den Einsatzbereich und Bewertung bleiben „theoretisch“ im schlechten Sinn, wenn sie nicht von solider Kenntnis der informatischen Tätigkeit getragen werden. Diese Kenntnis kann nur die eigene Tätigkeit erfassen, nicht die Informatik als ganze. So kann ich als Vertreter der Theoretischen Informatik nicht die „Entwicklung von Computersystemen“ beurteilen, sondern nur die benutzten Formalismen und Algorithmen. Trotzdem arbeite ich damit, wenn ich es gut mache, an einer Theorie der Informatik, nicht nur an einer Theorie der Theoretischen Informatik: Die genannte Aufgabe ist so formuliert, dass sie die Zusammenarbeit mit Vertretern anderer Fachgebiete und anderer Disziplinen – als Vermittlung zwischen unterschiedlichen bis gegensätzlichen Sichtweisen – erzwingt, aber auch ermöglicht.

An einer Theorie der Informatik arbeiten heißt Informatikkultur entwickeln. Kulturen müssen vielfältig sein, wenn sie sich entwickeln wollen, habe ich oben gesagt (Bro79). Daher wird die Theorie wie eine Kultur nicht einheitlich sein; verschiedene Theoriearbeiter werden Unterschiedliches entwickeln wollen, sich dabei zu verständigen suchen, aber nicht einer Meinung sein. Wenn sie sich aber in der Aufgabe der Informatik einig sind, werden sie sich auch im Ziel der Theoriearbeit einig sein: Den Sinn ihrer Arbeit zu erfassen. Dann kann aus lauter kleinen Theorieteilen *eine* Theorie der Informatik erwachsen. Wir brauchen eine Theorie der Informatik, wenn wir unsere Disziplin entwickeln und erhalten und zu einem nach außen wie innen überzeugenden Selbstverständnis bringen wollen (Abschnitt 7). Das kann nur eine Theorie leisten, die aus der Informatik heraus entsteht, nicht eine philosophische, psychologische, soziologische oder sonstige Theorie von außen über die Informatik.

In vielen Diskussionen über die Informatik steht das komplementäre Paar der Wissenschaft und ihrer Anwendungen im Vordergrund. Wir sind ihm in diesem Aufsatz in verschiedenen Formen begegnet: Wissenschaft vs. Anwendungen, Kern- vs. Angewandte Informatik, Hersteller vs. Nutzer. In seiner allgemeinen Form heißt das Paar *Theorie vs. Praxis*. Das macht ein Problem deutlich, das immer entsteht, wenn Informatiker sich mit einer „Theorie der Informatik“ konfrontiert sehen. Theorie machen wir selber, sagen sie misstrauisch; wozu noch eine von außen? Ich präsentiere deswegen im folgenden das Paar noch einmal allgemein. Wenn wir dabei ‚Theorie’ gleichzeitig als ‚Wissenschaft Informatik’ und als ‚Theorie dieser Wissenschaft’ lesen, sehen wir hoffentlich, dass das Misstrauen auf einem Missverständnis beruht.

Üblicherweise werden Theorie und Praxis als gegensätzlich angesehen: Theoretiker sehen Dinge von außen, sie sind Zuschauer (Plato), „zurückgelehnt“ (Thoreau), um nicht vom Strom der Ereignisse fortgerissen zu werden, sie wollen verstehen und höchstens durchs Reden verändern. Praktiker dagegen handeln, sie sind mittendrin, lassen sich mittragen, wollen durch Handeln verändern und, wenn überhaupt, dadurch lernen. Anders formuliert: Praxis setzt Muster fort, ist in Bewegung; Theorie sucht nach den Kernen in Mustern, strebt nach Ruhe. Also sind beide füreinander unentbehrlich: Praxis ohne Theorie sieht die Muster nicht, geht daher leicht in die Irre oder tritt auf der Stelle, stört oder zerstört Entwicklung. Theorie ohne Praxis verändert die Muster nicht, macht sie starr, um sie besser zu fassen, behindert oder versteckt Entwicklung. Theorie und Praxis sind ein komplementäres Paar. Wer Entwicklung verstehen will, muss seine Theorie aus der Praxis herleiten und seine Praxis von der Theorie leiten lassen.

Tatsächlich lassen wir uns bei jedem Handeln von Theorie leiten, meist ohne es zu wissen (Fab71). Diese Theorien sind Gespinste aus Vorstellungen, Hoffnungen, Befürchtungen, Visionen, wilden Ideen, die wir irgendwann, irgendwo aufgesammelt haben. Sie tragen unser Handeln von innen genauso wie die "Umstände" von außen. Man kann das abstreiten oder ignorieren; dann bleibt man ein Opfer seiner Vergangenheit. Oder man kann versuchen, etwas über seine Theorien herauszufinden. Dann kann man sie vielleicht ändern, sich mit anderen darüber austauschen, Gemeinsamkeiten und Unterschiede feststellen. Während man so nach Theorie sucht, tritt man aus seiner Arbeit heraus, steht neben ihr, also neben sich, bleibt aber in dem Gebiet. Theorie ist eine Position außerhalb, die von innen her kommt, habe ich definiert. Theoriearbeit heißt, die verborgenen Theorien (eigene und die anderer) aufzuspüren, zu formulieren, zu vergleichen, um besser arbeiten zu können.

Eine Theorie der Informatik zerfällt also nicht in eine Theorie der wissenschaftlichen Disziplin und eine Theorie ihrer Anwendungen (Brödner, Seim, Wohland in TdI02ff); sie wirft aber auch nicht Wissenschaft und Praxis in einen Topf oder ersetzt gar die Wissenschaft. Sie macht die Verbindungen klar: Technik zu entwickeln und zu verwenden, ohne das kreative Potential der Wissenschaft dafür zu nutzen, wäre dumm. Umgekehrt erhält wissenschaftliche Arbeit wichtige Impulse und ihren Sinn aus der praktischen Verwendung, ohne dass sie sich mit Verwendbarkeit ihrer Produkte rechtfertigen müsste. Eine solche Theorie der Informatik vermittelt zwischen der Wissenschaft und ihren Anwendungen und damit zwischen Herstellern und Nutzern (Abschnitte 2, 3, 5, 7).

Als noch schärfer gegensätzlich werden üblicherweise *Theorie und Ethik* angesehen: Theoretiker wollen die Welt verstehen; sie glauben, dass Menschen sich dann in ihr „richtig“ verhalten können. Wenn Theorie zur Lehre wird, schaltet sie Gefühle als „subjektiv“ und äußere

Umstände als „zufällig“ aus, will dadurch „das Richtige“ gewinnen und in ewig gültige „Gesetze“ gießen. Theorie als Lehre ist „rational“. Ethiker wollen die Welt so erklären, dass Menschen sich in ihr „gut“ verhalten können. Wenn Ethik zur Lehre wird, sieht sie von Menschen als „egoistisch“ und von äußeren Umständen als überwindbar ab, will dadurch „das Gute“ gewinnen und in ewig gültige „Normen“ gießen. Ethik als Lehre ist „fundamental“.

Tatsächlich sind Theorie und Ethik eng verwandt. Nur was wir für gut oder schlecht halten, beeinflusst unser Handeln. Die (zumeist unbewussten) Theorien, die unser Tun bestimmen, bestehen aus Bewertungen so gut wie aus Erklärungen. Theoriearbeit heißt also auch, die Werte zu finden, die uns und andere leiten, und zu einem Ethikgebäude zu fügen. Ethikarbeit besteht darin, die Werte zu verwirklichen; nicht das Gebäude zu entwerfen, sondern es zu realisieren. Theorie macht Ethik verständlich, Ethik macht Theorie wahr. Theorie und Ethik sind keine Gegensätze, sondern Zwillinge von sehr unterschiedlicher Natur. Wenn man die beiden trennt, sich nur um eines kümmert, verkümmert das andere, und man selbst; s. Werte und ihre Schatten, Abschnitt 3.

So ist es zu verstehen, dass Ethik wie Theorie üblicherweise als gegensätzlich zur Praxis angesehen wird. Auch Ethiker sehen die Dinge von außen, sie sitzen nur nicht zurückgelehnt, sondern vorgebeugt, höchst engagiert. Sie wollen die Welt nicht erklären, sondern unser Tun in bestimmte Bahnen lenken. Tatsächlich sind also auch Ethik und Praxis füreinander unentbehrlich: Praxis ohne Ethik produziert beliebige Muster, ist daher leicht irrelevant oder richtet Schaden an. Ethik ohne Praxis urteilt nach Maßstäben, die nicht von dieser Welt sind, schließt Überraschungen aus, be- oder verhindert Entwicklung. Ethik und Praxis sind ein komplementäres Paar. Wer Entwicklung fördern will, muss seine Ethik mit der Praxis abstimmen und seine Praxis von der Theorie bestimmen lassen.

Technik zu entwickeln und zu verwenden, ohne die Veränderungen, die sich daraus in allen Bereichen ergeben, wissenschaftlich zu untersuchen, wäre unverantwortlich (Sie05b, Woe05). Es wäre auch töricht; denn solche Untersuchungen können ebenso neue Entwicklungen in Gang bringen wie sie verhindern. Wir sollten Verantwortung als Herausforderung für wissenschaftliche und technische Arbeit betrachten, nicht als lästige Bürde. Wenn wir Theorie und Ethik versöhnen wollen, dürfen wir Menschen nicht auf Werte oder Begriffe reduzieren; zwischen Schemata kann man nicht vermitteln (Abschnitte 1,3). Wir müssen menschliche Verhaltensmuster von beiden Seiten her anschauen und untersuchen, wie sich gedankliche und gefühlsmäßige Muster in ihnen überlagern; vgl. (Grü00, Bah07). Dann können beide Bereiche sich gegenseitig befruchten: Ethik liefert Motive für wissenschaftliche Untersuchungen, Theorie Kriterien für ethische Entscheidungen. Wie oben können wir ‚Theorie’ dabei ebenso als ‚Theorie der Informatik’ wie als ‚Theorie in der Informatik’ oder ‚Informatik als

Theorie' lesen. Dann ergibt sich eine Wechselwirkung zwischen Theorie der Informatik und Ethik der Informatik; vgl. meine Arbeit „Ohne eine Theorie der Informatik keine Ethik für Informatiker, ohne eine Ethik der Informatik keine Theorie für Informatiker" in (Sie93).

Wir arbeiten alle mit Theorien und Ethiken, die ineinander versponnen sind. Und für die Ethik gilt genauso wie für die Theorie: Sie muss aus unserer Arbeit kommen. Eine Ethik für Informatiker, von irgend welchen Leuten, vielleicht gar Ethikern, erarbeitet und für alle verbindlich, ist ebenso ein Unding wie eine Theorie der Informatik, von irgend welchen Leuten, vielleicht gar Philosophen, erarbeitet und für alle gültig.

Wir werden das Verhältnis zu unserer Arbeit ändern müssen. Wir sind gewohnt, Verantwortung für unser Tun abzulehnen, weil Programme für alles benutzt werden können: Die beste Idee kann zur schlimmsten Anwendung führen, weil wir ihre Folgen nicht absehen. Das ist aber kein Grund, sich verantwortungslos zu verhalten. Gegensätze können nur im Dialog zu komplementären Paaren werden (Crutzen, Einleitung, Abschnitt 7). Wenn wir jederzeit – mit Kollegen und "Betroffenen", mit Gegnern und Freunden – über unsere Arbeit reden und wohin sie führen könnte, verstehen wir besser, was wir wirklich wollen, was wir durch Hybridisierung bezwecken und verstecken wollen (letzter Aufsatz in Sie92, Siefkes in TdI92).

Christopher Alexander hat „zeitlose" Muster des Bauens gesammelt, die Lebens- und Bewegungsmuster der Bewohner unterstützen, um eine „Sprache" guter Architektur aufzubauen. Davon angeregt sammeln Informatiker „Entwurfsmuster" als eine Sprache guten Softwareentwurfs (Abschnitt 1). In (Sie05a,b) habe ich angeregt und begonnen, allgemeiner „Informatikmuster" als eine Sprache guter informatischer Arbeit auszumachen. Wie kann man damit „Typen" von Informatikern bzw. von informatischem Arbeiten unterscheiden und bewerten? In Hinblick auf Abschnitt 5 wäre ein erster Schritt, Entwurfsmuster so verallgemeinern, dass sie die Benutzersicht einbeziehen.

Ähnliches gilt für das dritte Paar *Theorie vs. Politik*. Beide Bereiche werden zumindest strikt getrennt gehalten: Wissenschaft darf durch Politik so wenig kontaminiert werden wie durch Ethik. „Wir liefern die Technik, Politiker und Öffentlichkeit entscheiden über Förderung und Verwendung." Offensichtlich ist aber heute kein gesellschaftlicher Bereich ohne IT-Einsatz mehr denkbar; IT durchsetzt und verändert alle „Kulturen", von den lokalen kleiner Betriebe bis zu den globalen der „Weltgesellschaft". Welche Folgerungen ergeben sich für Kern- und Angewandte Informatik und für das Management aus der Tatsache, dass informatisches Denken auf Universalität hinzielt und Globalisierung in ihrer heutigen Form ohne IT nicht denkbar ist? Müsste eine Theorie der Informatik nicht stärker ethisch und politisch argumentieren? Dann könnte eine Theorie der Informatik mit einer „Politik der Informatik" ebenso in Wechselwirkung treten wie mit einer „Ethik der Informatik"; vgl. auch Abschnitt 8.

Aus einer Theorie der Informatik ergeben sich auch für das *Informatikstudium* erhebliche Konsequenzen. Neben Technik und Mathematik brauchen Informatiker Kenntnisse und Fähigkeiten aus den Geistes- und Sozialwissenschaften, nicht als Schmuck, sondern als drittes Standbein. Dafür müssen sie – Studenten wie Dozenten – keine Universalgenies werden. Wenn Dozenten herausfinden wollen, welche Rolle ihr Spezialgebiet im Prozess von De-, In- und Konstruktion spielt (Abschnitt 5), müssen und können sie die engen Grenzen der Informatik überschreiten und lernen, fremde Wege zu gehen, und das den Studenten beibringen. Das ist nicht schwerer oder leichter, als im eigenen Gebiet immer tiefer zu bohren, erfordert aber ein Umdenken – auch derjenigen, die Wissenschaft weiter wie bisher betreiben. Unsere Curricula sind mit der Disziplin gewachsen und haben zu viel Material angesammelt, das nur historischen Wert hat. Wenn wir sie auf das beschränken, was im Licht einer Theorie der Informatik wichtig ist, ist genug Raum für Neues. Informatikgeschichte z.B., wenn sie wie oben betrieben und beschrieben wird, ist dabei nicht zusätzlicher lästiger Stoff, sondern hilft den „Stoff“ zu verdauen. Der Grund, den wir jetzt im Grundstudium legen, behindert eine wissenschaftliche Ausbildung, statt sie zu begründen. Ein Studiengang ist kein Tempel, der auf Betonfundamenten steht; Wissen wächst nur auf offenem Boden (Sie93).

„Geschichte als Zugang zur Informatik“ wurde deswegen in einem Studienreformprojekt an der TUB im Anschluss an das Sozialgeschichteprojekt erarbeitet (GZI02; s. auch Sie01, 02b). Geschichte als Generationenfolge von immer schnelleren Rechnern oder Hierarchien von Programmiersprachen langweilt; Geschichten von Pionieren erfreuen, aber belehren nicht. In Texten von Informatikern nach kulturellen Orientierungen zu suchen, kann Studenten wie Dozenten, Wissenschaftlern wie Anwendern einen neuen Zugang zur Informatik eröffnen (s.o.): Eine Ingenieurdisziplin, die nicht nur Probleme löst oder schafft, sondern Teil der Kultur ist, in der sie selber leben. Eine Wissenschaft, die ihre Verantwortung nicht leugnet oder verdrängt, sondern als Herausforderung annimmt.

In den großen Zyklen des Grundstudiums werden den Studenten Methoden, Praktiken und Theorien der Informatik als Fakten präsentiert. "Das ist die Informatik, auf diese Felsen müsst ihr bauen." Diese erdrückende Faktizität kann man aufbrechen, wenn man die Studenten in historischen Texten sehen lässt, wie Fakten entstehen und vergehen, wie sie sich widersprechen oder unterstützen oder jede Beziehung verweigern. Sie lernen, nach Gründen und Alternativen zu fragen. Sie erkennen, dass die Entwicklung in die Zukunft nicht so überraschend, aber auch nicht so unausweichlich ist. Sie entdecken, dass „Geschichte“ als globale Dokumentation des Gewesenen und „Geschichten“ als lokale Darstellung von Erlebtem komplementär sind (s.o.). Vielleicht motiviert sie das sogar, ihre Programme gut zu dokumentieren.

Die Kunst des Verbreitens von Fähigkeiten („Lehren“) besteht darin, in den Lernenden vorhandene Muster so anzuregen, dass die gewünschten Muster durch Resonanz entstehen; vgl. (Ses01, 04). Das macht Lehren so aufregend: Wir wissen nie, welche Position aus dem Cluster des gelehrten Generators angenommen wird (Abschnitt 3). Die Kunst des Aneignens von Fähigkeiten („Lernen“) besteht darin, sich den fremden Mustern zu öffnen, ohne die eigenen dabei zu verlieren. Das macht Lernen so spannend: Wir wissen nie, wohin es uns führen wird. Angst und Faszination, Hochmut und Unterwürfigkeit sind Gift fürs Lehren wie fürs Lernen. Fürs Verwenden und Verbreiten von IT gilt das in besonderem Maße, weil sie, wie erwähnt, überall und nirgends zu finden ist. Schon während des Studiums Erfahrungen im Umgang mit Organisationen („Praxis“) und mit anderen Disziplinen („Theorie“) zu sammeln, wäre die beste Vorbereitung für Informatiker. Faszination und Ängste vor IT-Technik abzubauen, scheint der erste Schritt für Entwickler, Manager und Nutzer. Danach bleibt nur das gemeinsame Arbeiten.

Sicher müssen Studenten Programmieren lernen, damit sie mit ihrer Arbeit zurechtkommen. In der Praxis kommen auf jeden Informatiker ca. drei Leute ohne solche Ausbildung. Wer soll denen Programmieren beibringen, wenn es die Ausgebildeten nicht können? Gerade deswegen dürfen aber in der Ausbildung Programmiersprachen nur ein Hilfsmittel sein, nicht Selbstzweck. Genauso sollen Studenten in den mathematischen Lehrveranstaltungen Formalisieren lernen, nicht Formalismen (s.o.). Und den Zugang zu den Geistes- und Sozialwissenschaften können sie nicht über lauter Zweitstudien gewinnen. Wir sollten ihnen Programmieren als Hybridisierungsaufgabe stellen. Dann lernen sie nicht nur, korrekte Programme zu schreiben, sondern auch ihren Wert zu beurteilen. Damit können sie solide Fähigkeiten in der Informatik gewinnen, die durch Fähigkeiten in den anderen Bereichen gestärkt und nicht nur ergänzt werden (Sie 92 ff, Siefkes in TdI92ff, Fre97, Bau01, Hes03).

Eine Theorie der Informatik ist für viele Informatiker ein Ärgernis. Die einen oder anderen sagen: Theorie ist bloß

- Ideologie: *Wir* tun etwas; *die* stellen Regeln auf, wie es sein soll.
- Utopie: *Wir* verfolgen Ziele; *die* schwärmen davon, wohin es führen könnte.
- Phantasie: *Wir* wissen Bescheid; *die* reden darüber und haben keine Ahnung.

Wenn man die Tuer aber fragt: Wozu tut Ihr das? Was wollt ihr damit? Welchen Sinn ergibt es für Euch und andere? Dann werden sie je nach Eigenart redselig oder schweigsam. Dann entwickeln sie, in Gedanken oder im Reden, Theorie. Theoria war im alten Griechenland der Bericht derer, die zu den Heiligen Spielen gesandt worden waren. Theoretiker waren Enthusiasten. Sie wollten ihre Begeisterung mit denen teilen, die zu Hause saßen, träge oder

frustriert oder zu beschäftigt mit den täglichen Dingen. Erst die griechischen Philosophen haben sich aus Enthusiasten in Zuschauer verwandelt, haben aus der Theorie Kontemplation gemacht (Are58).

Diesen Schritt gehen wir Theoretiker nicht mit. Wir sind Enthusiasten, keine Philosophen. Die Praxis ist kein fernes Spiel, sondern unsere eigene Arbeit als Informatiker, die wir tun wie die zu Hause Gebliebenen. Über die Arbeit anderer reden wir, soweit sie sich mit der unseren berührt, im Dialog mit den anderen und mit anderen Theoretikern. Wir suchen nach einer Theorie der Informatik, weil wir Sinn in unserer Arbeit sehen wollen und die Begeisterung und den Ärger, den wir daraus ziehen, verstehen und kommunizieren wollen.

Theorie soll Sicherheit geben, aber nicht unverrückbare von außen wie eine Ideologie, sondern Gewissheit über den Sinn unseres Tuns. Theorie soll Richtungen geben, aber nicht eindeutige oder ins Jenseits führende wie eine Utopie, sondern vielfältige hin zu anderen Menschen und Disziplinen. Theorie soll uns Fragen stellen lassen, aber nicht irgendwelche phantastischen, sondern konkrete, die neue Wege öffnen. So soll unsere Theorie zwischen Ideologie, Utopie und Phantasie hin und her pendeln: ein bisschen von jedem, aber keines ganz.

Literatur

(Ale77) Christopher Alexander: A Pattern Language: Towns, Buildings, Construction. New York: Oxford UP.

(Ale79) -"- : The Timeless Way of Building. New York: Oxford UP.

(Are58) Hannah Arendt: The Human Condition. University of Chicago Press. – Deutsch: Vita Activa. München: Piper 1981.

(Aue46) Erich Auerbach: Mimesis - Dargestellte Wirklichkeit in der Abendländischen Literatur. Bern: Francke; 2. Aufl. 1959.

(Bah07) Corinna Bath: Künstliche Emotionen und ihre Gegenbewegungen. Erscheint in (TdI07).

(BSW07) -"-, Ralf Streibl, Ulrike Wilkens : Informatik im Kontext. Wie wird Identität konstruiert. Erscheint in (TdI07).

(Bat72) Gregory Bateson: Steps to an Ecology of Mind. Ballantine Books.

(Bat79) -"- : Mind and Nature - a Necessary Unity. Bantam Books. Deutsch: Geist und Natur - eine notwendige Einheit. Suhrkamp 1982.

(Bau01) K. Bauknecht et al. (Hg.): Informatik 2001, Jahrestagung GI & OCG. Workshop "Erkenntnistheorie - Semiotik - Ontologie". Österreich. Computergesellschaft, Wien.

(Brö97) Peter Brödner: Der überlistete Odysseus. Über das zerrüttete Verhältnis von Menschen und Maschinen. Berlin: edition sigma.

(Bro79) Urie Bronfenbrenner: The ecology of human development. Harvard University Press.

(B&H97) Hugh Beyer, Karen Holtzblatt,: Contextual Design: Defining Customer-Centered Systems. San Francisco: Morgan Kaufmann.

(B&H99) -"- : Contextual Design. In: Interactions 1+2/99, pp. 32-42.

(Cio97) Luc Ciompi: Die emotionalen Grundlagen des Denkens. Entwurf einer fraktalen Affektlogik. Vandenhoeck.

(Coy85) Wolfgang Coy: Industrieroboter. Rotbuch, Berlin.

(Cru00) Cecile Crutzen: Interactie, een wereld van verschillen. Een visie op informatica vanuit genderstudies. Dissertation, Open Universiteit Nederland, Heerlen.
(Cru01) -"- : Dekonstruktion, Konstruktion und Inspiration. FifF Kommunikation 3/01, S. 47-52.
(C&H01) -"- , Hans-Werner Hein: Die bedenkliche Dienlichkeit und Sicherheit von Softwaresystemen und die erlebte Verlässlichkeit. In (Bau01), S. 782-787.
(C&H01) -"- : Dekonstruktion und Konstruktion. Erscheint in (TdI07).
(Des01) Jörg Desel: Das ist Informatik. Springer.
(Fab71) Karl-Georg Faber: Theorie der Geschichtswissenschaft. Beck.
(Flo92) Christiane Floyd et al. (eds.): Software Development and Reality Construction. Springer.
(Flo97) Christiane Floyd: Autooperationale Form und situiertes Handeln. In C. Hubig (Hrsg.): Cognitio humana - Dynamik des Wissens und der Werte. XVII. Deutscher Kongress für Philosophie, Vorträge und Kolloquien, S. 237-252.
(Fre97) Christian Freksa et al. (eds.): Foundations of Computer Science – Potential, Theory, Cognition. Springer.
(Gam95) Erich Gamma et al.: Design Patterns. Elements of reusable object-oriented software. Addison-Wesley.
(Gid84) Anthony Giddens: The Constitution of Society. Outline of the Theory of Structuration. Berkeley. – Deutsch: Die Konstitution der Gesellschaft. Campus 1988.
(Grü00) Barbara Grüter: e-motion - über elektronische Formen der Bewegung und die Gestaltung von Interaktionssystemen. MMI-Interaktiv, Nr.4, S. 1-16, ISSN 1439-7854. http://www.mmi-interaktiv.de/ausgaben/
(GZI02) Korb, Joachim et al.: Geschichte als Zugang zur Informatik. TU Berlin, FB Informatik, Bericht 02-15.
(G&R07) Dorina Gumm, Arno Rolf: Anforderungsmanagement im Spannungsfeld zwischen De- und Re-Kontextualisierung. Erscheint in (TdI07).
(Hel03) Hans Dieter Hellige: Geschichten der Informatik. Springer.
(Hes03) Wolfgang Hesse (Hg.): Objekt-, subjekt- oder handlungs-orientiert? - Perspektiven der Informatik. Interdisziplinäres Symposium, Marburg, Juli 2001. EMISA Forum 23, GI-FG „Entwicklungsmethoden für Informationssysteme und deren Anwendungen".
(Joh87) Mark Johnson: The Body in the Mind. University of Chicago Press.
(Kli96) Ralf Klischewski: Anarchie – ein Leitbild für die Informatik. Von den Grundlagen der Beherrschbarkeit zur selbstbestimmten Systementwicklung. Frankfurt/Main: Peter Lang.
(Klu07) Johann W. Kluczny: Der Wert und sein Schatten – Der Wert des Schattens. Manuskript.
(Kra06) Detlev Krause, Arno Rolf et al.: „Wissen, wie alles zusammenhängt". Das Mikropolis-Modell als Orientierungswerkzeug für die Gestaltung von IT in Organisation und Gesellschaft. Informatik-Spektrum Heft 06/4, S. 263-273.
(Kre07) Hans-Jörg Kreowski (Hg.): Informatik und Gesellschaft - Verflechtungen und Perspektiven. FIfF-Reihe *Kritische Informatik,* LIT.
(LeD01) Joseph LeDoux: Das Netz der Gefühle. Wie Emotionen entstehen. Dtv.
(Lin97) Jürgen Link: Versuch über den Normalismus. Wie Normalität produziert wird. Westdeutscher Verlag: Opladen.
(MMK06) Arno Rolf, Dirk Siefkes: IT-Gestaltung im Labyrinth der Organisation. Verbreiten, Verwenden, Verstehen. Moderatoren- und Hintergrundpapier und Bericht, AG der Arbeitstagung Mensch-Maschine-Kommunikation 2006. wiw.f4.fhtw-berlin.de/nullmeier/index_mmk.htm
(Orl00) Wanda Orlikowski: Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. Organization Science 11-4, pp. 404-428.
(Ort03) Günther Ortmann: Regel und Ausnahme. Paradoxien sozialer Ordnung. Suhrkamp.
(Pei86) Max H. Fisch: Peirce, Semiotic, and Pragmatism. Indiana University Press 1986.

(Rec00) Rechenberg, Peter: Was ist Informatik? Hanser 1991, 3. Aufl. 2000.
(Rei92) Fanny-Michaela Reisin, Kooperative Gestaltung in partizipativen Softwareprojekten. Peter Lang: Frankfurt/Main.
(Rol98) Arno Rolf: Grundlagen der Organisations- und Wirtschaftsinformatik. Springer.
(Rol07) -"- : Reiseführer für Informatik und Gesellschaft. Erscheint in (Kre07).
(R&Z96) Dirk Riehle, Heinz Züllighoven: Understanding and Using Patterns in Software Development. In Karl Lieberherr et al. (eds.): Theory and Practice of Object Systems. Special Issue Patterns. Vol. 2/1, pp. 3-13.
(Sce97) Heidi Schelhowe: Das Medium aus der Maschine. Campus.
(Sce07) -"- : Technologie, Imagination und Lernen. Grundlagen für Bildungsprozesse mit Digitalen Medien. Waxmann.
(Sci95) Schinzel, Britta: Schnittstellen zwischen Informatik und Gesellschaft. Vieweg.
(Sci01) -"-, Johannes Busse, Dirk Siefkes: Bildung und Computer. Themenschwerpunkt FIfF-Kommunikation 1/01.
(Ses01) Werner Sesink: Einführung in die Pädagogik. Münster: LIT.
(Ses04) -"- : In-formatio: Die Einbildung des Computers. Münster.
(Ses07) -"- : Zur bildungstheoretischen Bedeutung des Diskurses zwischen Pädagogik und Informatik. Erscheint in (TdI07).
(SGI97a) Peter Eulenhöfer et al.: Informatics as Cultural Development. TU Berlin, FB Informatik, Bericht 97-2.
(SGI97b) -"- : Die Rekonstruktion von Orientierungsmustern in Fachtexten aus der Informatik. TU Berlin, FB Informatik, Bericht 97-3.
(SGI97c) -"- : Die Konstruktion von Hybridobjekten als Orientierungsmuster in der Informatik. TU Berlin, FB Informatik, Bericht 97-23.
(SGI98a) -"- : Sozialgeschichte der Informatik. FIfF-Kommunikation 2/98, S. 3-4, 28-48.
(SGI98b) Dirk Siefkes et al. (Hg.): Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen. Deutscher Universitätsverlag.
(SGI99a) -"- : Pioniere der Informatik. Interviews mit F.L.Bauer, C.Floyd, J.Weizenbaum, N.Wirth, H.Zemanek. Springer.
(SGI99b) Peter Eulenhöfer: Die formale Orientierung der Informatik. Zur mathematischen Tradition der Disziplin in der Bundesrepublik Deutschland. Dissertation, TU Berlin.
(SGI01) Heike Stach: Zwischen Organismus und Notation. Zur kulturellen Konstruktion des Computer-Programms. Deutscher Universitätsverlag.
(Sie82) Dirk Siefkes: Kleine Systeme. TU Berlin, FB Informatik, Bericht 82-14. Engl.: Small Systems. Purdue University, Computer Science, CSD-TR 435, 1983.
(Sie90) -"- : Formalisieren und Beweisen. Logik für Informatiker. Vieweg, 2. Aufl. 1992.
(Sie92) -"- : Formale Methoden und kleine Systeme. Lernen, leben und arbeiten in formalen Umgebungen. Vieweg.
(Sie93) -"- : Evolutionäre Modelle in der Informatik. TU Berlin, FB Informatik, Bericht 93-15.
(Sie94) -"- : Das wirkliche Geschlecht der Informatik. In *Ulrike Erb et al.* (Hrsg.): Erfahrung und Abstraktion - Frauensichten auf die Informatik. Universität Hamburg, FB Informatik, FBI-HH-M233/1993, 155-160.
(Sie95) -"- : Ökologische Modelle geistiger und sozialer Entwicklung. Beginn eines Diskurses zur Sozialgeschichte der Informatik. Wissenschaftszentrum Berlin für Sozialforschung, Bericht FS II 95-102.
(Sie99a) -"- : Die Rolle von Schemata in der Informatik als kultureller Entwicklung. TU Berlin, FB Informatik, Bericht 99-6.
(Sie99b) -"- : Abspalten oder verbinden? Oder beides? Feministische Kritik und ökologische Wissenschaft. In C. von Braunmühl (Hg.): Der blockierte Dialog. Zur Rezeption feministischer Theorie-Impulse im Wissenschaftsbetrieb. Berlin-Verlag, S. 65-89.
(Sie01) -"- : Schreiben und Geschichte als Zugang zur Informatik. FIfF-Kommunikation 18/4, S. 11-13.

(Sie02) -"- : Sozialgeschichte und kulturelle Theorie der Informatik. TU Berlin, Fak. Elektrotechnik & Informatik, Bericht 02-16.
(Sie05a) -"- : Informatikmuster als Grundstock für eine Theorie der Informatik. Manuskript.
(Sie05b) -"- : Theorie der Informatik und Verantwortung von Informatikern. Wie sich informatische und kulturelle Entwicklung in Informatikmustern mischt. Erscheint in (Kre07).
(Sie07a) -"- : Muster im Umgang mit Informationstechnik. Erscheint in Dorina Gumm et al.: Immer Ärger mit der Technik - Transdisziplinäre Analysen typischer Probleme im alltäglichen Umgang mit Informationstechnologien. LIT.
(Sie07b) -"- : Theoretische Informatik und Theorie der Informatik. Was kann eine allgemeine Theorie der Informatik bringen? Erscheint in (TdI07).
(Sie07c) -"- : Theorie der Informatik im Widerspruch. Manuskript.
(Sik07) Jörg Siekmann: KI … und sich zum Bilde schuf er sie. Erscheint in (TdI07).
(Ste95) Daniel N. Stern: The Motherhood Constellation. Basic Books. Deutsch: Die Mutterschaftskonstellation. Klett-Cotta 1998.
(TdI92) Wolfgang Coy, Frieder Nake, Jörg Pflüger, Arno Rolf, Jürgen Seetzen, Dirk Siefkes, Reinhard Stransfeld (Hg.): Sichtweisen der Informatik. Vieweg.
(TdI01) Frieder Nake, Arno Rolf, Dirk Siefkes (Hg.): Informatik - Aufregung zu einer Disziplin. Tagung zur Theorie der Informatik 2001. Uni Hamburg, FB Informatik, Bericht 235.
(TdI02) -"- : Wozu Informatik? Theorie zwischen Ideologie, Utopie, Phantasie. Tagung zur Theorie der Informatik 2002. TU Berlin, Fak. Elektrotechnik & Informatik, Bericht 02-25.
(TdI03) -"- : Informatik zwischen Konstruktion und Verwertung. Tagung zur Theorie der Informatik 2003. Uni Bremen, FB Mathematik & Informatik, Bericht 1/04.
(TdI07) Andreas Möller, -"- : Beiträge zu einer Theorie der Informatik. Zum kritischen Selbstverständnis einer Disziplin. Erscheint im E-Journal Communication, Cooperation, Participation.
(TIK02) Dirk Siefkes, Johannes Busse, Jochen Ludewig, Rüdiger Valk, Ulrike Wilkens: Informatik im interdisziplinären Kontext. Erhaltung und Entwicklung durch Kooperation. Bericht zur AG „Informatik als Hybridwissenschaft". In (TdI02), S. 112-126.
(VTR91) Francisco Varela, Evan Thompson, Eleanor Rosch: The Embodied Mind. Cognitive Science and Human Experience. MIT Press.
(Vyg78) Lev Vygotski: Mind in Society. The Development of Higher Psychological Processes. Harvard University Press.
(V&M07) Rüdiger Valk, Daniel Moldt: Über die Informatik und ihre osmotische Brandmauer. Erscheint in (TdI07).
(Wih96) Rudolf Wilhelm: Informatik. Beck.
(Wik00) Ulrike Wilkens: Das allmähliche Verschwinden der informationstechnischen Grundbildung. Zum Verhältnis von Informatik und Allgemeinbildung, Aachen: Shaker.
(Woe05) Martin Woesler (Hg.): Ethik der Informationsgesellschaft. Bochum: Europäischer Universitätsverlag

(Zu meinen Publikationen siehe auch http://tal.cs.tu-berlin.de/siefkes)

8.11.2007