# Grundlagenforschung in der Informatik? Perspektiven der Informatik und ihre Erkenntnisziele

**Lorenz Hilty***

Grundlagenforschung bedeutet, die grundlegenden Theorien und Konzepte eines Faches weiterzuentwickeln. Es sind also die Fundamente einer Wissenschaft, die durch Grundlagenforschung erweitert oder erneuert werden. Wo liegen die Fundamente der noch relativ jungen Wissenschaft Informatik? Diese Frage wird seit Jahrzehnten kontrovers diskutiert, gerade weil die Informatik eine so vielfältige Wissenschaft ist. Dabei gehen die Auffassungen schon über die Rolle des Computers in der Informatik auseinander. Ist der Computer der Erkenntnisgegenstand der Informatik oder geht es in der Informatik «genauso wenig um Computer wie in der Astronomie um Teleskope», wie der holländische Informatiker Edsger W. Dijkstra es ausdrückte? Ich möchte Sie zu einem Rundgang einladen, auf dem wir die Informatik aus vier verschiedenen Perspektiven betrachten. Dabei werde ich auf die Erkenntnisziele eingehen, die sich aus der jeweiligen Perspektive stellen.

«Ich will weniger Chemie schlucken», «In unserem Team stimmt die Psychologie nicht», «Die Informatik in meiner Firma ist veraltet»: Was haben diese drei Sätze gemeinsam? Den falschen Gebrauch des Namens einer Wissenschaftsdisziplin. Um es vorwegzuschicken: «Informatik» ist kein Sammelbegriff für digitale elektronische Geräte und Infrastrukturen, hierfür gibt es die Bezeichnung «Informations- und Kommunikationstechnik», kurz «IKT».[1] Schauen wir uns nun die Informatik aus verschiedenen Perspektiven an, die den verschiedenen Positionen im Diskurs um das Selbstverständnis dieser Wissenschaft entsprechen.

## Perspektive 1: Automatisierte Informationsverarbeitung

Gemäss dieser historisch ersten Sichtweise ist der zentrale Gegenstand der Informatik die *Automatisierung von Prozessen der Informationsverarbeitung mit Hilfe von Computern.* Dies erklärt auch die Herkunft des Wortes «Informatik», das 1957 vom Computerpionier Karl Steinbuch durch Zusammenziehung der Wörter «Information» und «Automatik» geprägt wurde. Im angelsächsischen Raum wurde das gleiche Fachgebiet zunächst als «Computer Science», seit 1989 auch als «Science of Computing» bezeichnet (Denning et al., 1989).[2] In dieser Perspektive ist die Informatik ein Forschungsprogramm, das – nach Jahrhunderten der Automatisierung körperlicher Tätigkeiten mit Hilfe mechanischer Maschinen – nun auch geistige Tätigkeiten durch einen neuen Typ von Maschinen automatisieren soll. Diese neuen Maschinen wurden «Computer» genannt. Zuvor war «Computer» übrigens eine Berufsbezeichnung für Menschen, die Berechnungen ausführten. Genau genommen müssten unsere heutigen Computer also «künstliche Computer» heissen.

Als Geburtsstunde der neuen Automatisierungswissenschaft Informatik gilt die Formulierung eines theoretischen Automatenmodells durch den britischen Mathematiker Alan Turing im Jahr 1936. Ich muss hier auf einige seiner Überlegungen eingehen, da diese für das Verständnis der ersten Perspektive notwendig sind. Turing beschrieb einen Automaten, der zu einer gegebenen Eingabe in Form einer Zeichenkette eine Ausgabe erzeugt, die der Eingabe eindeutig zugeordnet ist. Dieser Automat, der später «Turingmaschine» genannt wurde, tut somit nichts weiter, als eine mathematische Funktion zu berechnen. Die Turingmaschine wurde zum Modell für jede Form der Berechnung und damit jegliche Informationsverarbeitung. (Die Gleichsetzung von Berechnung, Informationsverarbeitung und geistigen Tätigkeiten ganz allgemein beruht auf einem Paradigma, auf das ich am Ende dieses Abschnitts zurückkommen werde.)

* Universität Zürich, Institut für Informatik, Binzmühlestrasse 14, 8050 Zürich.

E-mail: *hilty@ifi.uzh.ch*

**Lorenz M. Hilty,** Dr. rer. nat., ist Professor für Informatik und Nachhaltigkeit an der Universität Zürich und leitet die gleichnamige Forschungsgruppe der Empa in St. Gallen. Studium in Physik, Informatik und Psychologie an den Universitäten Zürich und Hamburg, Promotion 1991 und Habilitation 1996 in Informatik (Hamburg). Forschungsaufenthalte am Institut für Wirtschaft und Ökologie der Universität St. Gallen (1992) und am Forschungsinstitut für Anwendungsorientierte Wissensverarbeitung an der Universität Ulm (1996). Professur für Wirtschaftsinformatik an der Fachhochschule Nordwestschweiz (1998), Aufbau und Leitung der Abteilung Technologie und Gesellschaft der Empa (2004), Gastprofessur am Institut für Soziale Ökologie in Wien (2008), Professur an der Universität Zürich (2010) und Delegierter der Universitätsleitung für Nachhaltigkeit (2014). Im Nebenamt ist Lorenz Hilty Affiliated Professor an der KTH Stockholm, wo er regelmässig lehrt, ohne ein Flugzeug zu besteigen.

[1] auch «Informationstechnik», kurz «IT».

[2] Inzwischen gibt es daneben auch die Bezeichnung «Informatics», allerdings nicht immer bedeutungsgleich mit «Informatik».

Zu jedem Problem, das *überhaupt* durch Berechnung lösbar ist, gibt es eine Turingmaschine, die die entsprechende Funktion berechnet und somit das Problem löst. Entscheidend für die Entstehung der Informatik war nun aber, dass Turing zeigte, dass es unter den unendlich vielen möglichen Turingmaschinen eine gibt, die alle anderen ersetzen kann, «universelle Turingmaschine» genannt. Diese bekommt als Eingabe eine eindeutige Beschreibung des Verfahrens, das sie ausführen soll – genannt «Algorithmus» – neben der eigentlichen Eingabe für die Berechnung. Anhand der Beschreibung kann sie die Berechnung simulieren, also nachahmen, die die andere Turingmaschine ausführen würde. Weil das Ergebnis der simulierten Berechnung von dem Ergebnis, das die «echte» Turingmaschine berechnen würde, nicht unterscheidbar ist (lediglich der Aufwand für die Berechnung kann unterschiedlich sein), benötigt man die andere Maschine nicht. Man braucht also nur *eine* Turingmaschine, um potenziell *alle* berechenbaren Probleme lösen zu können. Vorausgesetzt natürlich, man kann für jedes Problem den passenden Algorithmus formulieren. Damit hatte die Idee des programmierbaren Computers ihr theoretisches Fundament. Was wir heute als die Hardware eines Computers kennen, ist äquivalent zu einer universellen Turingmaschine, wenn man von der Begrenztheit des Speichers absieht. Programme sind sprachliche Beschreibungen von Algorithmen.

Das Fazit aus diesen Überlegungen lautet: Automatisierung ist in der Informatik identisch mit der Formulierung von Algorithmen. Damit wird der *Algorithmus* – und nicht der Computer – zum zentralen Gegenstand der Informatik. Entsprechend dieser Sichtweise ist als Fundament eine *theoretische Informatik* entstanden, die sich hauptsächlich mit den Eigenschaften von Algorithmen, den sie beschreibenden formalen Sprachen und den sie ausführenden Automaten befasst. Von Interesse ist insbesondere, mit welchem Aufwand an Speicherplatz und Rechenschritten ein Algorithmus ausführbar ist. Danach werden Algorithmen in sog. Komplexitätsklassen eingeteilt.

Das bekannteste Beispiel für ein *ungelöstes* Problem der theoretischen Informatik ist die Frage, ob die beiden als «P» und «NP» bekannten Komplexitätsklassen identisch sind. Sollte diese Vermutung bewiesen werden, könnte das Auswirkungen auf die Sicherheit verschlüsselter Daten haben, denn diese beruht darauf, wie komplex das Verfahren ist, das ein Angreifer einsetzen müsste, um die Daten zu entschlüsseln. Ein Beispiel für ein *gelöstes* Problem der theoretischen Informatik ist die Frage, ob es einen Algorithmus gibt, der einen anderen Algorithmus als Eingabe bekommt und entscheiden kann, ob dieser eine bestimmte (nicht-triviale) Eigenschaft hat. Wäre das der Fall, könnte man Beweise für die Korrektheit von Programmen automatisieren. Nach dem Satz von Rice lautet die Antwort leider: «Nein». Als Konsequenz daraus ist es nicht möglich, die Verifikation von Programmen zu automatisieren. Unter anderem deshalb leben wir mit fehlerhaften Programmen und sind tagtäglich gezwungen, uns auf eine prinzipiell unzuverlässige digitale Infrastruktur zu verlassen. Soweit zwei Beispiele für fundamentale Fragen der Informatik in dieser ersten Perspektive.

Ich komme nun zurück auf das Paradigma, das der ersten Perspektive der Informatik zugrunde liegt. Es wurde von Allen Newell und Herbert Simon 1976 als «*Physical Symbol System Hypothesis*» (PSSH) formuliert und ist auch unter dem Namen «Informationsverarbeitungs-Paradigma» bekannt. Die PSSH besagt, dass der Umgang mit Symbolen nach gegebenen Regeln eine ausreichende Voraussetzung für intelligentes Handeln ist. Symbole werden dabei explizit als physische Entitäten definiert. Weil Symbole aber austauschbar sind und es nur auf ihre gegenseitige Unterscheidbarkeit und die Regeln zu ihrer Verarbeitung (eben die Algorithmen) ankommt, kann Intelligenz mit nahezu beliebigen materiellen Mitteln realisiert werden, insbesondere mit biologischer oder elektronischer «Hardware». Dieses Paradigma könnte man auch so formulieren: Intelligenz ist Informationsverarbeitung und Informationsverarbeitung ist Umgang mit physischen Symbolen nach Regeln, also Berechnung. Die PSSH hat in der analytischen Philosophie des Geistes zu einer anhaltenden Debatte geführt, insbesondere über die Möglichkeiten und Grenzen von Künstlicher Intelligenz. Wir werden als dritte Perspektive eine Auffassung von Informatik kennenlernen, die sog. kontextuelle Informatik, die ohne die PSSH auskommt.

**Perspektive 2: Gestaltung des Virtuellen**

Turings Idee der universellen Maschine macht uns den enormen Freiraum bewusst, den ein programmierbarer Computer anbietet. Ein Computer ist eine Maschine, die buchstäblich alles und nichts kann, nämlich *potenziell alles* (was berechenbar ist und wofür der Speicher ausreicht) und *vorläufig nichts*, solange man keinen Algorithmus formuliert hat, der eine spezielle Maschine aus ihm macht. So gesehen ist Informatik der Ort, *wo mit Sprache Maschinen konstruiert werden*, und zwar nahezu beliebige Maschinen. Diese Gestaltungsfreiheit ist faszinierend und herausfordernd zugleich. Ein Bildhauer, der vor einem unbehauenen Felsblock steht, braucht viel Imaginationsfähigkeit, um die Skulptur zu sehen, die er formen will, und zugleich sehr viel handwerkliches

Geschick, um sie zu realisieren. Und auch wenn beides zusammenkommt, ist noch lange nicht garantiert, dass ein Werk entsteht, das viele Menschen als Bereicherung empfinden – wie beispielsweise die Skulpturen von Auguste Rodin.

Das gilt analog auch in der Softwareentwicklung, aber es gibt entscheidende Unterschiede. Wer Software entwickelt, braucht zwar Vorstellungskraft und handwerkliche Fähigkeiten zur Umsetzung seiner Vorstellung. Aber die Einschränkungen durch physikalische Gesetze gelten nicht, denn das Artefakt entsteht nicht in der physischen Realität, sondern im virtuellen Raum. Man kann sich die Bedeutung von «virtuell» in der Informatik durchaus aus dem Begriff des virtuellen Bildes der Physik ableiten. Ein virtuelles Bild ist das Bild eines Gegenstandes, der für den Betrachter hinter einem Spiegel zu liegen scheint. Der Versuch, nach dem Apfel im Spiegel zu greifen, scheitert aber, denn der Apfel ist nicht real. Er hat zwar einige Eigenschaften mit dem realen Apfel gemeinsam, aber nicht alle. In genau diesem Sinne ist all das, was einen Computer über seine berührbaren Teile (die Hardware) hinaus ausmacht, nur virtuell vorhanden. «Der Computer erlaubt es, rein gedankliche Welten mit eigenen Gesetzmässigkeiten zu schaffen – mit enormen realen Wirkungen. Informatik ist die Wissenschaft, welche die Gesetzmässigkeiten dieser virtuellen Welten erforscht und dem Menschen dienstbar macht. Die Erfindung des Computers führt aus einer mechanistisch geprägten Welt der Materie und der Naturgesetze zu einer informationsbestimmten Welt von Daten und Algorithmen.» (Gsell, 2013, S. 5) Die realen Wirkungen der virtuellen Artefakte entstehen dadurch, dass diese mit der realen Welt über Sensoren und Aktuatoren verbunden sind. Genau genommen ist eine Tastatur bereits eine Ansammlung von Sensoren, Bildschirme sind Ansammlungen von Aktuatoren, Touch-Screens beides in einem. Es gibt unzählige weitere Arten von Sensoren und Aktuatoren. Wenn wir an die vielen Geräte denken, die von eingebetteten Prozessoren gesteuert werden, vom Auto bis zum Roboter, dann wird deutlich, dass die Kopplung zwischen virtueller und realer Welt sehr eng werden kann.[3]

Betrachtet man die *Gestaltung des Virtuellen* als Hauptgegenstand der Informatik, hat dies zwei interessante Implikationen. Erstens wird die Möglichkeit, komplexe Ideen präzise auszudrücken, zu einem Hauptanliegen. Man könnte die Informatik geradezu als die «Wissenschaft vom Explizit-Machen» bezeichnen. Zu diesem Zweck wurden über Jahrzehnte viele künstliche Sprachen entwickelt und verfeinert, darunter Programmiersprachen, Spezifikationssprachen und Modellierungssprachen. Die zweite Implikation lautet: In der Informatik geht es um Gestaltungsprobleme. Im Gegensatz zu einem Bildhauer, einem Architekten oder einem Autodesigner ist aber der Softwareentwickler nicht durch die Gesetze der Physik eingeschränkt, kann jede seiner Entscheidungen ohne Materialverlust rückgängig machen und hat dadurch enorme Gestaltungsfreiheit. Diese ist nur begrenzt durch die eigene Phantasie, durch die Ausdrucksmöglichkeiten der Sprache und durch die begrenzten Hardwareressourcen, die zur Ausführung zur Verfügung stehen. Diese Freiheit hat enorme Vorteile, aber auch den Nachteil, dass sich Softwareentwickler fast zwangsläufig in der selbst geschaffenen Komplexität ihrer Schöpfungen verstricken. Programme wachsen schnell an, Millionen von Anweisungen sind keine Seltenheit. Stellen Sie sich vor, ein Romanautor würde Ihnen berichten, dass er an einem Krimi von mehreren zehntausend Seiten arbeitet, womöglich zusammen mit Koautoren, mit denen er sich nicht immer gut versteht. Würden Ihnen keine Zweifel kommen, ob die Handlung des Krimis wirklich schlüssig ist? Und ob jemand dieses Mammutwerk jemals nachvollziehen wird? Die Unüberschaubarkeit von Software führt dazu, dass massenhaft unbeherrschbare und damit unsichere Artefakte in die Praxis entlassen werden. Wären die Ergebnisse von Softwareentwicklung so deutlich sichtbar wie die Ergebnisse städtebaulicher Architektur, wir würden wohl überwiegend den Eindruck haben, in Slums zu leben.[4]

Das Gestaltungsproblem in der Informatik ist im Kern ein Problem der kognitiven Komplexität für den menschlichen Betrachter. Es geht hier also nicht mehr primär um Berechnungskomplexität wie in der ersten Perspektive, sondern um die Überschaubarkeit, die Beherrschbarkeit der virtuellen Artefakte. In dieser zweiten Perspektive lautet die Grundfrage der Informatik deshalb: *Wie können wir die Gestaltungsfreiheit in der virtuellen Welt so nutzen, dass unsere Artefakte beherrschbar bleiben und zu jedem Zeitpunkt die Anforderungen erfüllen, die wir in der realen Welt an sie stellen?*

3 Der Begriff der «virtuellen Realität» wird vor diesem Hintergrund als (beabsichtigtes) Oxymoron erkennbar. Ziel der Schaffung «virtueller Realität» ist es, die Gestaltung des Virtuellen und seine Kopplung mit den menschlichen Sinnen so weit zu treiben, dass man dem Erleben von Realität möglichst nahe kommt – dass wir also wie Alice in den Romanen von Lewis Carroll in die Welt hinter dem Spiegel schlüpfen können.

4 Der Vergleich von Software mit Slums stammt vom Computerpionier Joseph Weizenbaum, der zeitlebens vor den Folgen einer unkritischen Überschätzung der Möglichkeiten des Computers gewarnt hat.

Trotz beachtlicher Fortschritte sind ausreichende Antworten auf diese Frage noch nicht gefunden. Es ist immer noch die Regel und nicht etwa die Ausnahme, dass Softwareprodukte fehlerhaft, unsicher und schwer änderbar sind, weil sie im Laufe ihrer Entwicklung unbeherrschbar komplex geworden sind. Softwareprodukte mit Millionen von Anweisungen, an denen hunderte von Entwicklern arbeiten, sind keine Seltenheit. Als wäre die reale Welt mit ihren Interessenkonflikten und Abhängigkeiten nicht schon komplex genug, erzeugen wir fahrlässig virtuelle Artefakte, die weitere unbeherrschbare Komplexität in die Welt tragen. Um es mit Edsger W. Dijkstra zu sagen: «*Computing's core challenge is how not to make a mess of it. [...] Because we are dealing with artifacts, all unmastered complexity is of our own making; there is no one else to blame and so we had better learn how not to introduce the complexity in the first place.*» (Dijkstra, 1996)

Software Engineering ist das Teilgebiet der Informatik, das dieses Problem zu lösen versucht. Über den richtigen Weg scheiden sich die Geister. Ist Softwareentwicklung eine Wissenschaft, eine Kunst oder eine Managementaufgabe? Im *ersten Fall* wären formale Methoden weiterzuentwickeln, die es erlauben, Softwareprodukte so solide wie mathematische Theorien zu konstruieren und ihre Korrektheit formal zu beweisen – oder, wo das nicht möglich ist, sie nach rigorosen wissenschaftlichen Standards empirisch zu testen (wissenschaftlicher Ansatz). Bei aller Anerkennung für die Fortschritte auf diesem Gebiet bestehen verbreitete Zweifel, ob dieser Ansatz genügen kann. Um in der Analogie zur gebauten Umwelt zu bleiben: Würde man der Welt einen Gefallen tun, wenn man Architektur auf Baustatik reduzieren würde? Betrachtet man Softwareentwicklung dagegen als Form von Kunst (*zweiter Fall*), wird das Fach schwer lehrbar und wir haben keine Antwort auf das Problem, dass die unendliche virtuelle Spielwiese mit ernsten Realitäten verknüpft ist: Mit Fabriken, Fahrzeugen, Flugzeugen, Operationssälen, Kriegsmaschinen, Finanzmärkten und neuerdings auch mit demokratischen Wahlen. Sieht man Softwareentwicklung, so der dritte Ansatz, primär als Managementproblem, so legt man den Fokus auf den Entwicklungsprozess, auf seine Organisation und seine Werkzeuge. Das ist zweifellos notwendig, denn Software wird im Team entwickelt, und es gibt weitere Beteiligte wie die Auftraggeber und die zukünftigen Benutzer. Ein erheblicher Teil des Gestaltungsproblems der Informatik hat seine Wurzeln deshalb in Kommunikationsproblemen zwischen Menschen. Denn diese müssen ja *gemeinsam* das oben erwähnte Problem des «Explizit-Machens» komplexer Ideen lösen. Der Schwierigkeitsgrad dieses Problems und das Risiko unbeherrschter Komplexität potenziert sich mit der Zahl der Beteiligten. Wie also soll man den Prozess organisieren, welche Sprachen und Werkzeuge soll man zur Kommunikation verwenden, um die Komplexität im Zaum zu halten? Heute sind so genannte *agile Methoden* als Gegenbewegung zu den traditionellen, als bürokratisch empfundenen Entwicklungsmethoden beliebt, die der Kreativität und der Kommunikation hohen Stellenwert einräumen.

Grundlagenforschung im Software Engineering müsste meines Erachtens das Problem lösen, den agilen mit dem wissenschaftlichen Ansatz zu versöhnen. Hierzu wäre Softwareentwicklung als abundante, durchaus kreative Produktion von Mikrotheorien aufzufassen, die kommunizierbar sind und – mit Unterstützung von einfach zu benutzenden Werkzeugen – systematisch intersubjektiv validiert werden. Eine solche diskursorientierte Grundlagenforschung würde weit über die Informatik hinausweisen. Sie würde das generelle Problem adressieren, wie subjektive Theorien effizient kommuniziert und im Diskurs besser genutzt werden können.

**Perspektive 3: Gestaltung soziotechnischer Systeme**

Der Technikphilosoph Günter Ropohl erläutert den Begriff des soziotechnischen Systems am Beispiel der Arbeit mit einem Computer: «Wenn Text geschrieben wird, tut das nicht allein der Mensch, aber es ist auch nicht allein der Computer, der den Text schreibt; erst die Arbeitseinheit von Mensch und Computer bringt die Textverarbeitung zuwege. Da freilich im benutzten Computer immer schon die Arbeit anderer Menschen verkörpert ist, da also die Mensch-Maschine-Einheit nicht nur durch den einzelnen Nutzer gebildet, sondern auch von anderen Menschen mitgeprägt wird, bezeichne ich sie als *soziotechnisches System*.» (Ropohl, 2009, 58f)

Schon in der ersten Perspektive war mit dem Begriff der Automatisierung die menschliche Arbeit implizit angesprochen. Ausgeblendet blieb aber die eigentlich naheliegende Frage, welche Arbeitsaufgaben im Zuge dieser Automatisierung durch Maschinen ersetzt oder unterstützt werden und mit welchen Methoden dies geschieht. Wie werden Arbeit und Organisationsstrukturen analysiert und wie werden sie durch diese Analyse und durch die darauf basierende Automatisierung verändert? In der zweiten Perspektive, die das Gestaltungsproblem der Informatik in den Fokus nahm, haben wir die humane und soziale Dimension zwar einbezogen, aber nur insoweit, als die Gestaltung des Virtuellen selbst eine menschliche Tätigkeit ist, die Kreativität und Komplexitätsbeherrschung erfordert. Nun erweitern wir die

Perspektive in einem dritten Schritt auf die Arbeitsprozesse und Organisationsstrukturen, die *für* die Gestaltung virtueller Artefakte analysiert und *durch* die Anwendung dieser Artefakte verändert werden. Dies entspricht, mit Steve Easterbrooks Worten, dem Übergang von «*computational thinking*» zu «*systems thinking*»: «*Computer professionals already have a conceptual toolkit for problem solving, sometimes known as computational thinking. However, computational thinking tends to see the world in terms a series of problems (or problem types) that have computational solutions (or solution types). Sustainability, on the other hand, demands a more systemic approach, to avoid technological solutionism, and to acknowledge that technology, human behaviour and environmental impacts are tightly inter-related. [...] I argue that systems thinking provides the necessary bridge from computational thinking to sustainability practice.*» (Easterbrook, 2014, S. 235)

Führt man sich das riesige Veränderungspotenzial der Digitalisierung vor Augen, das sich seit Mitte des letzten Jahrhunderts scheinbar unerschöpflich entfaltet, betrachtet man den resultierenden wirtschaftlichen Strukturwandel und die rasante Veränderung sozialer Praktiken, so muss eine Informatik *ohne* diese erweiterte Perspektive erscheinen wie Städtebau ohne Verständnis für Wohnen, Verkehr und alle anderen Aspekte menschlichen Lebens. Noch einmal Ropohl: «Neue technische Lösungen sind immer auch neue Handlungsmuster, von Menschen für Menschen entworfen und damit Kristallisationen gesellschaftlicher Verhältnisse.» (Ropohl 1991, S. 233)

Spätestens seit den 1990er Jahren gibt es die Forderung nach einer um die humane und soziale Dimension erweiterten Informatik und einer theoretischen Fundierung der Informatik, die diesem Aspekt Rechnung trägt und sich damit grundlegend von dem bisher als theoretische Informatik definierten Gebiet unterscheidet (Coy et al., 1992). Für eine derart um den Anwendungskontext erweiterte Informatik hat Dieter Engbring (2003) die Bezeichnung «kontextuelle Informatik» vorgeschlagen. In der kontextuellen Informatik geht es nicht allein um «Artefakte», sondern auch um «Kognifakte» und «Soziofakte». Diese Terminologie hat sich zwar nicht allgemein durchgesetzt, ich verwende «kontextuelle Informatik» jedoch als Sammelbegriff für Ansätze in der Informatik, die eine um humane und soziale Phänomene erweiterte Perspektive einnehmen.

Zu diesen Ansätzen gehört das Fachgebiet der Mensch-Computer-Interaktion (*Human-Computer Interaction, HCI*), das ursprünglich die Interaktion zwischen Mensch und Computer unter dem Aspekt der Ergonomie oder Benutzbarkeit erforschte, seine Perspektive aber seither stark erweitert hat. Dazu gehört zweitens ein erweitertes Verständnis von Softwareentwicklung, etwa der partizipative Entwurf oder die immer systematischere Einbeziehung der zukünftigen Nutzenden in den Prozess der Anforderungsermittlung (*Requirements Engineering*). Dazu gehört drittens die Sozialinformatik (*Social Informatics*), die die Wechselwirkung zwischen IKT und sozialen Beziehungen zu ihrem zentralen Gegenstand macht. Ihr Begründer Rob Kling definierte sie als «*the interdisciplinary study of the design, uses and consequences of information technologies that takes into account their interaction with institutional and cultural contexts.*» (Kling 2007, S. 205)

Vor allem gehören aber alle anwendungsorientierten Gebiete der Informatik zur kontextuellen Informatik, die sich zu sog. Schnittstellen- oder Brückendisziplinen zum jeweiligen Anwendungsgebiet entwickelt haben. Die bekanntesten dieser Brückendisziplinen sind die Wirtschaftsinformatik und die Medizininformatik. Betrachten wir die Wirtschaftsinformatik etwas näher: Ihr Gegenstand ist das Informationssystem (IS). Nach Frank et al. sind IS «keine rein technischen, sondern sozio-technische Systeme, d. h. sie arrangieren Menschen und technische Komponenten im Hinblick auf die Lösung bestimmter wirtschaftlicher Aufgabenstellungen» (Frank et al., S. 75). Damit ist kein rein technisches, sondern ein *soziotechnisches* System Objekt der Analyse und Gestaltung. Die Wirtschaftsinformatik wird im angelsächsischen Raum übrigens meist nach ihrem Gegenstand als «Information Systems» bezeichnet. Dieses Gebiet hat sich unabhängig von «Computer Science» entwickelt und ist überwiegend an Business Schools etabliert. Im Gegensatz zu «Information Systems» blieb die Wirtschaftsinformatik in der Informatik verankert und zeichnet sich durch die Verbindung der technischen und nicht-technischen Aspekte ihres Gegenstandes aus.

Die Idee einer kontextuellen Informatik mit soziotechnischen Systemen als Erkenntnisgegenstand (unsere dritte Perspektive), hat eine Reihe von Implikationen. Erstens könnte man in einer so verstandenen Informatik ohne die PSSH auskommen. Man würde Informationsverarbeitung und Kommunikation als Tätigkeiten von Menschen definieren, die durch virtuelle Artefakte lediglich unterstützt werden. Am Beispiel der Wirtschaftsinformatik:

– Das technische System verarbeitet keine Informationen, sondern Daten (Strukturen aus physischen Symbolen), die nur dadurch zu Informationen werden, dass Menschen sie interpretieren. Daten sind also *potenzielle Information*.

– Menschen kommunizieren nicht mit Maschinen, sondern *mit Hilfe* von Maschinen mit anderen Menschen.

Das Begriffssystem der Informatik würde dann kompatibel mit einem wesentlichen Teil der Geistes- und Sozialwissenschaften, was der interdisziplinären Zusammenarbeit nicht abträglich sein müsste.

Eine zweite Implikation ist die Einsicht in die Unvermeidbarkeit einer Entwicklung, die ich das «Ende der Reversibilität» nenne. Softwareentwickler sind es gewohnt, bei der Gestaltung ihrer virtuellen Artefakte jede Entscheidung rückgängig machen zu können. Die Erweiterung der Perspektive auf soziotechnische Systeme bedeutet – bedauerlicherweise – die Vertreibung aus diesem «Undo-Paradies». Für die Entlassung von Mitarbeitenden, für die versehentliche Veröffentlichung vertraulicher Daten oder die Fehlsteuerung eines Röntgengeräts gibt es keinen Undo-Button. Sobald die virtuellen Artefakte mit sozialen Realitäten verknüpft sind, wird aus der virtuellen Spielwiese, die in der zweiten Perspektive als Ort des reinen intellektuellen Vergnügens erschien, ein Schlachtfeld der ökonomischen Interessen – und im Extremfall (Cyberkrieg) sogar ein Schlachtfeld im ursprünglichen, militärischen Wortsinn.

Eine dritte Implikation der Perspektiverweiterung ist die Unklarheit, welche Grundlagen eine kontextuelle Informatik benötigt, um die (Mit-)Gestaltung soziotechnischer Systeme systematisch zu betreiben. Kontextuelle Informatik ist hier mit einem Dilemma konfrontiert: Soziotechnische Systeme sind soziale Phänomene; die theoretischen Fundamente zu ihrer Analyse liegen folglich in den Sozialwissenschaften. Eine kontextuelle Informatik kann zwar Methoden zur Analyse und Gestaltung solcher Systeme unter dem Aspekt der Information und Kommunikation entwickeln, hätte hierfür aber keine eigene theoretische Grundlage. Sie bliebe eine Art Handwerk auf der Basis halbwegs rezipierter sozialwissenschaftlicher Theorien und Positionen. Alternativ könnte man eine kontextuelle Informatik nicht als Disziplin, sondern als inter- oder transdisziplinäres Konglomerat definieren, in das die «Kerninformatik» (d.h. die Zusammenfassung der Perspektiven 1 und 2) nur die vorhandenen Grundlagen einbringt. Dieses Konglomerat hätte dann aber das Problem, dass es sich auf keine gemeinsamen und nicht einmal auf untereinander konsistente theoretische Grundlagen stützen könnte. Als Ausweg aus diesem Dilemma bietet sich der Weg an, die Methoden der Gestaltung soziotechnischer Systeme als Theorien aufzufassen: Diese Methoden und ihre intendierte Anwendung sind präzise definierbar, ihre Wirksamkeit nach gängigen wissenschaftlichen Standards empirisch überprüfbar. Steffen Greiffenberg hat diesen Ansatz für die Wirtschaftsinformatik formuliert: «Die Wirtschaftsinformatik kann, wenn sie ihre Methoden zur Gestaltung von Informationssystemen als Theorien auffasst, zum einen durch die rückwirkenden Anforderungen an Theorien an Wissenschaftlichkeit gewinnen und zum anderen die Brücke zwischen Real- und Formalwissenschaft schlagen.» (Greiffenberg, 2003, S. 964) Der Ansatz erscheint über die Wirtschaftsinformatik hinaus verallgemeinerbar für jede Ausprägung der kontextuellen Informatik. Grundlagenforschung in der kontextuellen Informatik besteht also darin, *Methoden der Gestaltung soziotechnischer Systeme als wissenschaftliche Theorien zu formulieren und weiterzuentwickeln.*

Zum Abschluss der Diskussion der dritten Perspektive möchte ich einige Argumente anführen, die für die systematische Weiterentwicklung der kontextuellen Informatik sprechen. Die Aufgabe, die Zukunft von Arbeit und Kommunikation zu gestalten, ist kulturell viel zu wichtig, als dass man sie in die Hände von Softwareentwicklern legen sollte. Denn hierfür sind sie weder ausgebildet noch legitimiert. Bereits zur Jahrtausendwende hat der US-amerikanische Verfassungsrechtler Lawrence Lessig mit seiner These «*Code is Law*» auf das Problem hingewiesen, dass Programmcode in unserer vom Internet geprägten Welt eine normative Wirkung entfalten kann, die durchaus mit der Wirkung staatlicher Regulierung zu vergleichen ist. Software transportiert implizite Wertvorstellungen und kann diese praktisch durchsetzen: «*Our choice is not between 'regulation' and 'no regulation'. The code regulates. It implements values, or not. It enables freedoms, or disables them. It protects privacy, or promotes monitoring. People choose how the code does these things. People write the code.*» (Lessig, 2000)

Das Risiko einer «kontextblinden» Informatik, die man auch «Informatik der Zauberlehrlinge» nennen könnte, wäre somit, dass normative Fragen systematisch als technische Fragen getarnt und dann von einer kleinen Gruppe von Technikgestaltern und ihren Auftraggebern entschieden werden. Dieses Risiko nimmt weiter zu, wenn die Artefakte der Softwareentwickler mehr Eigenständigkeit erlangen. Die intendierte Verwirklichung «Künstlicher Intelligenz» lässt Softwareprodukte autonome Entscheidungen fällen und aus Erfahrung lernen. Dabei kann es sogar praktisch notwendig werden, diese Artefakte als «*moral agents*» zu verstehen und mit «Wertvorstellungen» auszustatten, etwa bei selbstfahrenden Fahrzeugen oder bei Kriegsrobotern (Christen, 2016). Die Frage, wer diese normativen Entscheidungen

trifft, eine demokratische Gesellschaft in einem offenen Diskurs oder ein paar selbsternannte «Gesetzgeber» bei Pizza und Cola, sollte uns beschäftigen.

**Perspektive 4: Modellierung von Realweltphänomenen**

Diese Perspektive der Informatik lässt sich am besten erklären, wenn wir zunächst zur zweiten Perspektive (Gestaltung des Virtuellen) zurückkehren und den Spezialfall der Künstlichen Intelligenz (KI) betrachten. Die KI ist ein Teilgebiet der Informatik und zugleich ihr ehrgeizigstes Projekt. Die KI hat die Schaffung von Maschinen zum Ziel, die ein Verhalten zeigen, welches man bei Menschen als «intelligent» bezeichnen würde. Wir haben es hier also mit drei Gegenständen zu tun: Menschen, Maschinen und Intelligenz. Nehmen wir nun einfach an, dass unser Interesse gar nicht der Maschine, sondern der Intelligenz gilt. Intelligenz scheint ein natürliches Phänomen zu sein, das bei biologischen Organismen vorkommt und – wenn man die PSSH akzeptiert – mit maschinellen Mitteln reproduzierbar ist. Wir können nicht ausschliessen, dass es weitere Systeme gibt, die Intelligenz hervorbringen, z.B. Systeme aus interagierenden Individuen, so wie ein Ameisenstaat sich – in einem noch zu definierenden Sinn von «Intelligenz» –intelligenter verhält, als eine einzelne Ameise es könnte. Wenn es das Naturphänomen ist, das uns interessiert, wird die Informatik zu einer Naturwissenschaft (Denning, 2007). Aus dieser Perspektive trägt sie potenziell zum Erkenntnisgewinn auf vielen Gebieten bei, indem sie eine bestimmte Sichtweise beisteuert: Sie betrachtet Phänomene unter dem Aspekt der Informationsverarbeitung.

Eine Analogie soll diesen Gedanken verdeutlichen. Sowohl Vögel als auch Flugzeuge fliegen. Aus guten Gründen gibt es keine Diskussion darüber, ob man Flugzeuge als «künstliche Vögel» bezeichnen soll, denn dies wäre irrelevant. Es stört uns auch nicht weiter, dass Flugzeuge auf eine völlig andere Weise fliegen als Vögel, von den klimaschädlichen Emissionen einmal abgesehen. Dennoch gibt es eine Gemeinsamkeit: Vögel und Flugzeuge müssen die Gesetze der Aerodynamik respektieren, sonst könnten sie nicht fliegen. Mit Sicherheit haben wir durch den Versuch, fliegende Maschinen zu bauen, mehr über Aerodynamik gelernt als durch die Ornithologie, also die Vogelkunde. Zugleich hat die Entwicklung des künstlichen Fliegens unseren Respekt für die Leistung der Evolution vertieft, die fliegende Organismen hervorgebracht hat. Ich bin sicher, dass man mit ein wenig Verständnis für Aerodynamik den Flug eines Vogels eher für ein Wunder hält als ohne dieses Verständnis. Es mag paradox klingen, wenn ich sage, dass gerade der Versuch der technischen Rekonstruktion des natürlich Vorgefundenen eine Haltung fördern kann, die der Ethiker Giovanni Maio als «die Haltung der Achtung, die Haltung des Staunens, die Haltung des begierdefreien Betrachtens» bezeichnet (Maio, 2016).

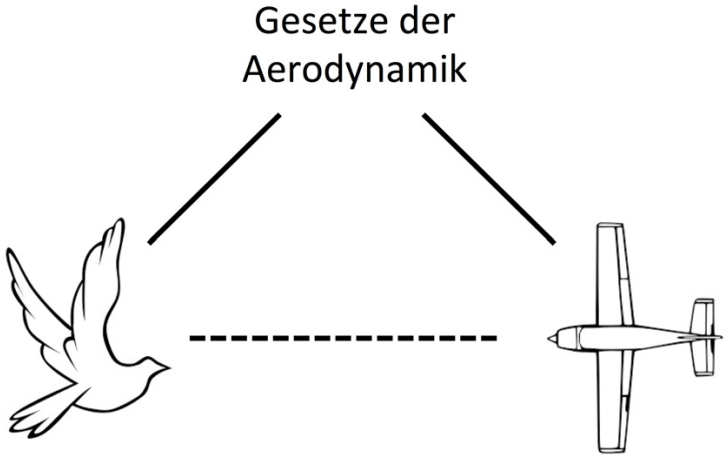


***Abbildung.*** *Der Vogel als Ergebnis der biologischen Evolution und das Flugzeug als Ergebnis der kulturellen Entwicklung verwenden verschiedene Methoden des Fliegens. Beide sind aber den Gesetzen der Aerodynamik und damit den gleichen Naturgesetzen unterworfen.*

Ersetzen wir in unserem Beispiel die Gesetze der Aerodynamik durch die Prinzipien des Rechnens oder – wenn man die PSSH akzeptiert – der Informationsverarbeitung und Intelligenz, dann ist die Gestaltung virtueller Maschinen als die Suche nach diesen Prinzipien zu verstehen. Die Geschichte der KI zeigt, dass sie unseren Respekt für Phänomene, die uns selbstverständlich sind – wie beispielsweise das Erkennen von Objekten auf Bildern, das Verstehen von Geschichten oder die Koordination eines Körpers mit vielen Gelenken – gerade deshalb gesteigert hat, weil die Versuche der KI dagegen sehr bescheiden ausfallen. Dass diese Bescheidenheit in der öffentlichen Wahrnehmung der KI nicht ankommt, ist eine Folge kommerzieller, nicht wissenschaftlicher Interessen.

Indem die Informatik virtuelle Modelle realer Phänomene konstruiert, bringt sie eine Forschungsmethode ein, die Richard M. Karp als «*Computational Lens*» bezeichnet (Karp, 2011). Diese Linse besteht allerdings nicht einfach im «*number crunching*» für die numerische Lösung von Gleichungen, was unter dem Stichwort «*Computational Sciences*» gelegentlich propagiert wird. Das wäre keine Informatik. Das Besondere an der neuen Linse ist vielmehr die Hervorhebung der Prozesse der Informationsverarbeitung. Es geht um ein Verstehen der Informationsprozesse in Realweltphänomenen durch den Versuch ihrer (Re-)Konstruktion. Neben der Intelligenz von Organismen sind hier insbesondere auch ökonomische Phänomene zu nennen, deren Erforschung mit der

«*Computational Lens*» uns helfen könnte, den Erfolg und das Versagen von Märkten besser zu verstehen.

Auch in dieser letzten Perspektive der Informatik stellt sich die Frage nach ihren Grundlagen. Dies führt uns zurück auf die PSSH und die Kritik an diesem Paradigma. Grundlagenforschung einer so verstandenen Informatik ist ihr Beitrag zum Diskurs in der analytischen Philosophie des Geistes, aber auch zu Grundfragen der Sozialwissenschaften wie der nach dem Verhältnis von Individuum und Gesellschaft.

**Fazit**

Die Informatik lässt sich aus unterschiedlichen Perspektiven betrachten, die unterschiedlichen Positionen zur Frage nach Erkenntnisziel und Methoden dieser Wissenschaft entsprechen. Diese Positionen existieren heute nebeneinander. Je nach Position benötigt die Informatik andere theoretische Fundamente und stellt andere Grundfragen. Diese lauten, stark zusammengefasst:

1. Wie können wir die Existenz und die Eigenschaften von Algorithmen zu gegebenen Problemen beweisen?
2. Wie können wir virtuelle Artefakte konstruieren, deren kognitive Komplexität beherrschbar bleibt?
3. Mit welchen Methoden können wir soziotechnische Systeme analysieren und durch den Einsatz von IKT in einer erwünschten Weise umgestalten?
4. Wie können wir zum Verständnis von Naturphänomenen beitragen, indem wir sie unter dem Aspekt der Informationsverarbeitung modellieren? ■

***Literatur***

Christen, M. (2016). Das Gute in der Informatik, Bulletin VSH-AEU 42(1), «Das Gute», S. 59–65 . Forch: Vereinigung der Schweizerischen Hochschuldozierenden.

Coy, W., Nake, F., Pflüger, J. M., Rolf, A., Seetzen, J., Siefkes, D., Stransfeld, R. (Hrsg.) (1992). Sichtweisen der Informatik. Braunschweig: Vieweg.

Denning, P. J. et al. (1989). Computing as a discipline. Communications of the ACM, (32)1, 9–23.

Denning, P. J. (2007). Computing is a natural science. Communications of the ACM, (50)7, 13–18.

Dijkstra, E. W.: The next fifty years. Manuscript, 1996.

Easterbrook, S. (2014). From Computational Thinking to Systems Thinking: A conceptual toolkit for sustainability computing. Proceedings of the 2014 Conference ICT for Sustainability, 235–244.

Engbring, D. (2003). Informatik im Herstellungs- und Nutzungskontext. Dissertation. Universität Paderborn.

Frank, U., Klein, S., Krcmar, H., Teubner, A. (1999). Aktionsforschung in der WI – Einsatzpotentiale und Einsatzprobleme. In: Wirtschaftsinformatik und Wissenschaftstheorie. Grundpositionen und Theoriekerne. S. 71–90, Universität Essen.

Greiffenberg, S. (2003). Methoden als Theorien der Wirtschaftsinformatik (2003).Wirtschaftsinformatik Proceedings 2003, 100.

Gsell, M. (2013). Vorwort. In: J. Kohlas, J. Schmid & C.A. Zehnder (Hrsg.), informatik @ gymnasium: Ein Entwurf für die Schweiz, S. 5–6. Zürich: Verlag Neue Zürcher Zeitung.

Karp, R. M. (2011). Understanding Science through the Computational Lens. In: Journal of Computer Science and Technology, Vol. 26(4), S. 569–577.

Kling, R. (2007). What Is Social Informatics and Why Does It Matter? The Information Society. 23(4): 205–220.

Lessig, L. (2000). Code is Law. On Liberty in Cyberspace. Harward Magazine, 1.1.2000.

Maio. G. (2016). Technik als Veränderung unseres Verhältnisses zur Welt. In: Bulletin VSH-AEU 42(3/4), «Eingriffe», S. 4–11. Forch: Vereinigung der Schweizerischen Hochschuldozierenden.

Newell, A; Simon, H. A. (1976). Computer Science as Empirical Inquiry: Symbols and Search. Communications of the ACM, 19 (3): 113–126.

Ropohl, G. (2009). Allgemeine Technologie. Eine Systemtheorie der Technik. 3., überarbeitete Auflage. Karlsruhe: Universitätsverlag.