

Gibt es wissenschaftliche Grundlagen für die Gestaltung von Software?

Christian Dahme

1. Vorbemerkungen

Will man die wissenschaftlichen Grundlagen der Informatik bzw. Softwareentwicklung ergründen, so sind mindestens drei Richtungen (Sichtweisen...) zu unterscheiden:

- A) aus der Sicht des Anwenders, was aus seiner Sicht mit dem Computer unterstützt bzw. durch den Computer (wünschenswerter Weise) realisiert werden sollte
- B) aus der Sicht des Informatikers (Soft- bzw. Hardwareentwicklers), was aus seiner Sicht mit einem Computer möglich ist bzw. wie Soft- bzw. Hardware zu konstruieren ist, um eine vorgegebene Spezifikation zu erfüllen.
- C) aus der Sicht des „Interface-Designers“ (im klassischen Ingenieurbereich könnte man dieses auch der industriellen Formgestaltung zuordnen), d.h. hier ein solches Werkzeug zu gestalten, das eine gute virtuelle Nachbildung der realen Situation des Anwenders darstellt.
(Es geht also um die Einbettung der Software in den Anwendungszusammenhang.)
Hier spielen Begriffe wie „usability“ (Gebrauchstauglichkeit), brauchbare Software, Benutzerfreundlichkeit, Fehlerfreundlichkeit u.dgl. eine Rolle.

Aus Sicht der Softwareentwicklung könnte man A) den frühen Phasen zuordnen, B) der Softwarekonstruktion und C) der Einbettung der Software in den Anwendungszusammenhang.

Im Folgenden werde ich mich zuerst A) und zum Schluss auch noch C) zuwenden. Die Richtung B) entspricht aus meiner Sicht der „Kerninformatik“, auf die ich primär nicht eingehen werden.

Es gibt auch Auffassungen, dass diese „Richtungen“ sich nicht trennen lassen, wovon ich aber nicht ausgehe.

Wissenschaftshistorisch lassen sich in Bezug auf A) drei Wurzeln identifizieren:

- a) Allgemein:
der Computer zur Unterstützung bis hin zur Automatisierung des Rechnens
Speziell:
der Computer als Mittel zur Unterstützung / Automatisierung von (wissenschaftlichen und kommerziellen) Berechnungen auf der Basis mathematisierter Modelle
Letzteres werde ich als gegenstandsorientierten Modellansatz bezeichnen.
- b) Automatisierung von (Teilen von) Tätigkeiten durch Übertragung an einen Computer
Diese Wurzel bezeichne ich als tätigkeitstheoretisch orientierten Ansatz.
- c) psychische Phänomene werden als „Informationsverarbeitung“ rekonstruiert
Dieses führt zum kognitivistischen Ansatz.

2. Der gegenstandsorientierte Modellansatz

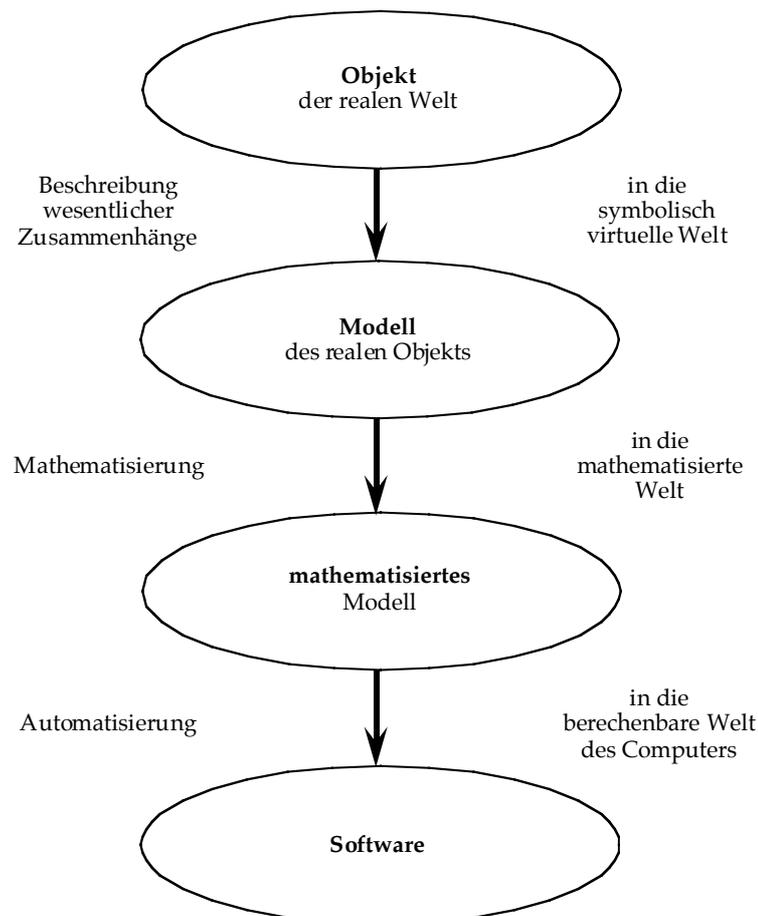
Wissenschaftshistorisch ist a) m.E. die älteste Wurzel. Es geht hier insbesondere darum, das Rechnen zu erleichtern.

Sie ist u.a. mit dem Wunsch verbunden, aufwendige Berechnungen durch "Rechen-

werkzeuge" später Rechenautomaten zu unterstützen und Rechenfehler zu vermeiden. Mit dem Aufblühen der Naturwissenschaften im 17. Jahrhundert entstand verstärkt das Bedürfnis, die uns umgebende beobachtbare Welt (Wirklichkeit) zu erklären und ggf. zu prognostizieren. Gleichzeitig wuchs damit die Bedeutung mathematisierter Modelle. So konstruierte 1623 Schickardt eine Rechenmaschine, um Kepler seine Berechnungen zu erleichtern. Diese Richtung führte zur Unterstützung bis hin zur Automatisierung der Berechnung mathematisierter Modelle der modernen Physik und Naturwissenschaften und diente damit auch als Grundlage für die Entwicklung von Atombomben, Raketen, Raumfahrt, Computertomographie u. dgl. im Sinne von angewandten Naturwissenschaften.

Andererseits gab es schon sehr lange das Bedürfnis kommerzielle Berechnungen zu unterstützen. So entwickelte B. Pascal 1642 seine „Pascaline“, um die Verwaltungsarbeit seines Vaters zu erleichtern. Charles Thomas - Chef zweier Versicherungsgesellschaften in Paris – kam auf die Idee, eine Rechenmaschinenproduktion aufzubauen. Grundlage hierfür war das von ihm 1820 fertig gestellte "Arithmometer", vom dem ungefähr 1500 in seiner Werkstatt produziert wurden. Diese Richtung führte ebenfalls zur Unterstützung bis hin zur Automatisierung der Berechnung (normativer) ökonomischer Modelle (einschließlich Operations Research und Ökonometrie) wie z.B. Weltmodellen des Club of Rom bzw. Modelle im Rahmen der Projekte des IIASA (International Institute for Applied Systems Analysis).

Aus naturwissenschaftlicher Sicht lässt sich dieser Ansatz durch folgende Transformation beschreiben:



Software kann dann als Automatisierung der "Berechnung" von mathematisierten Modellen verstanden werden.

Grundlage für die Software ist hier ein Modell des Gegenstandes¹ – daher gegenstandsorientierter Modellansatz -, und damit die Disziplin² des Gegenstandsbereiches, zu dem der Gegenstand gehört.³

3. Der tätigkeitstheoretisch orientierte Ansatz⁴

In diesem Ansatz geht es um die Frage, welcher Anteil einer Tätigkeit sich potentiell automatisieren lässt und damit die Chance hat, von einem Automaten durchgeführt zu werden.

Diese Wurzel geht insbesondere auf die Automatisierung des Webens (1728 Lochplatte von Falcon, 1808 Jacquardscher Webstuhl - Automatisieren des Anhebens der Kettfäden zur Musterbildung) zurück, mit der gleichzeitig das für die Informatik wichtige Prinzip der "Programmsteuerung" erfunden wurde.

Ausgehend von diesem Prinzip entwickelte Herman Hollerith⁵ ein elektromechanisches Lochkartensystem für die Auswertung der Volkszählung in den USA von 1890 - im Sinne von Datenerfassung, -bereitstellung und -auswertung - insbesondere großer Datenmengen⁶ - das später u.a. zur Entstehung von Datenbanken (Informationssystemen) führte.

Ausgehend von dem Anliegen, Teile einer Tätigkeit einem Automaten in Form von Software zu übertragen, ergibt sich folgende Interpretation von Software:

Software

- als Automatisierung von operationalisierbaren Anteilen (innerer) menschlicher Tätigkeit,
- als Resultat der Transformation von Anteilen realer bzw. möglicher menschlicher Tätigkeit in eine maschinelle Form,
- als Vergegenständlichung von menschlichen Fähigkeiten

Grundlage für die Softwareentwicklung ist hier die Tätigkeitstheorie.

Aus dieser Sicht könnte man die Tätigkeitstheorie als eine "Grundlagendisziplin" für die Softwareentwicklung ansehen.

In diesem Zusammenhang ergeben sich zwei Fragen:

1. Welcher Anteil einer Tätigkeit hat die Chance in Software übertragen zu werden?
2. Wie kommt man von diesem Anteil zu einem Automaten, der diesen realisiert?

Die erste Frage wurde in „Ein tätigkeitstheoretischer Ansatz zur Entwicklung von brauchbarer Software“ (Dahme, Ch.; Raeithel, A. 1997, S. 6) weitgehend beantwortet.

Folgende Transformation ist dafür notwendig:

1. Der Anteil, der die Potenz hat, sich in Software transformieren zu lassen, bezieht sich auf Anteile einer inneren, orientierenden Tätigkeit oder lässt sich in solche transformieren.
2. Diese Anteile liegen als Operationen vor oder lassen sich von Handlungen in Operationen überführen (Operationalisierbarkeit).

¹ s. hierzu: Dahme 1997, insbesondere das 1. Kapitel, aber auch Dahme 2001

² das können auch mehrere Disziplinen sein ==> Aus dieser Sicht ist Softwareentwicklung ein interdisziplinärer Prozess.

³ Dabei kann die Software selbst Rückwirkungen auf den ursprünglich abgebildeten Gegenstand haben, so dass man verkürzt auch spricht von: Software als (Mittel zur) System- bzw. Organisationsgestaltung

⁴ siehe: Dahme, Ch.; Raeithel, A. 1997, S. 5-12

⁵ Herman Hollerith Firma war Vorläufer von IBM

⁶ Ausgehend von den Erfahrungen bei der Volkszählung wurde dieses Verfahren von staatlicher bis hin zu betrieblicher Verwaltung wie unterschiedlichste Statistiken, Lagerhaltung, Kostenerfassung oder Lohnabrechnung angewandt.

3. Es liegt Wissen vor, mit dem sich diese Operationen (vollständig) reproduzieren lassen – reproduzierbares Wissen.
4. Dieses Wissen ist mitteilbar.
5. Es kann in öffentliches Wissen überführt werden.

Die Frage, wie man zu einem Automaten kommt, ist dann aus meiner Sicht ein Ingenieursproblem, d.h. es geht darum einen Automaten zu erfinden (d.h. hier eine Software zu entwickeln), die diesen Teil der Tätigkeit realisiert.

4. Der kognitivistische Ansatz

Wissenschaftshistorisch ist diese Richtung am jüngsten. Sie entstand 1956 am MIT, ist eng mit der Kybernetik⁷ verbunden und wird in der Psychologie auch als kognitive Wende bezeichnet. Charakteristisch hierfür ist das so genannte kognitivistische Paradigma: psychische Phänomene lassen sich als „Informationsverarbeitung“ rekonstruieren, d.h. sie basiert auf einer Metapher des Computermodells des Geistes. Diese Richtung wird später als Kognitivismus bezeichnet.

Ein spezielles Interesse bestand in der Erforschung des Problemlösens⁸.

Das führte u.a. zum GPS (General Problem Solver)⁹ aber auch zur KI.

Wenn Handeln wie beim GPS und im Kognitivismus als Problemlösen verstanden wird, dann ist Zielrealisierung das erfolgreiche Suchen, d. h. Finden des Weges zum Zielzustand. Genau diesen Problembegriff unterstellt auch die (theoretische) Informatik.

Software wird folglich als Automatisierung (Maschinisierung) eines Algorithmus¹⁰, der wiederum (selbst) eine Problemlösung repräsentiert, verstanden.

Diese Art der Charakterisierung von Software ist heute noch weit verbreitet und wird gerne in der theoretischen Informatik bevorzugt, da er explizit auf dem Begriff des Algorithmus basiert.

5. Der Zusammenhang zwischen diesen Wurzeln

Der kognitivistische Ansatz kann als ein spezieller gegenständlicher Modellansatz der kognitiven Psychologie verstanden werden (s. Klix 1971).

Im Buch „Systemanalyse menschlichen Handeln“ habe ich gezeigt, wie sich der tätigkeitstheoretisch orientierte Ansatz in einen gegenständlichen Modellansatz einbetten lässt.

Das führt zu folgender Aussage:

Software basiert auf einem Modell, unabhängig davon, ob man sich dessen bewusst ist oder nicht.

Andererseits sind gegenständliche Modelle Gegenstand bzw. Mittel innerer orientierender Tätigkeit¹¹, d.h. der gegenstandsorientierte Modellansatz lässt sich tätigkeitstheoretisch einbetten.

Beide Ansätze sind demnach gleichberechtigt.

⁷ einer der damals führenden kybernetisch-mathematischen Vertreter der kognitiven Psychologie war Friedhard Klix (Information und Verhalten; Berlin 1971), der an der Humboldt-Universität wirkte.

⁸ Siehe hierzu u.a.: Dörner, Dietrich 1983 und 1987, Klix, Friedhart 1971 und 1980

⁹ Newell und Simon 1961, 1972, 1988

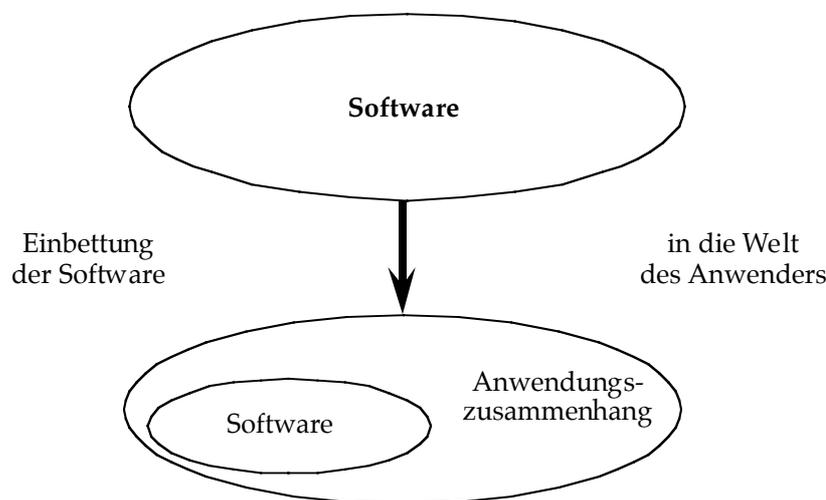
¹⁰ im Sinne von Turing (siehe 1988)

¹¹ siehe: Dahme, Ch.; Raeithel, A. 1997, S. 8f.

6. Einbettung der Software in den Anwendungszusammenhang

Hier geht es darum, quer zu den „automatisierten“ Operationen¹² bzw. der „automatisierten“ Berechnung von Modellen¹³ (dieses wird häufig als Funktion der Software bezeichnet) die Oberflächen- und Interaktionsebene, die mit der Funktionsebene harmonisieren muss, so umzubauen, „ dass sie für den Anwender als leicht verständliche und wirksame Software-Objekte ihre orientierende, virtuelle Kraft entfalten können. Im Vertrauen auf die Verlässlichkeit der angestoßenen Operationen, der angezeigten Materialzustände und Aufgabenerfüllungsgrade können die Anwender ihre jeweilig nächsten Teilziele in Ruhe bestimmen und ansteuern. Neben einer entsprechenden Gestaltung der Arbeitsfenster (z.B. Werkzeugteil, Materialsicht, Orientierungspanel) und der Hilfsfunktionen ist wieder die Einbettung der Software in die existierende Anwender-tätigkeit von äußerster Wichtigkeit.. Die Anwender begrüßen in der Regel die Übernahme ihrer spezifischen Sprache, ihrer üblichen Arbeitsschritte in Dialoge und Menüs. Dies erfordert die Nachbildung ihrer gewohnten Mittel und Materialien als Software-Objekte“ (Dahme, Ch.; Raeithel, A. 1997, S. 9).

Es geht folglich um die Transformation:



Hierbei stehen sich mindestens zwei Kulturen (die der Anwender und die der Softwareentwickler) gegenüber, deren Vertreten mit einander kommunizieren und gegebenenfalls kooperieren müssen/sollten. Dabei können die unterschiedlichsten Kommunikations-, Kultur- und Koordinationsprobleme¹⁴ auftreten.

Das Kriterium brauchbar - im Sinne von brauchbarer Software - ist ein typisches Kriterium für diesen interkulturellen Prozess.

Solche interpersonellen, interkulturellen, kooperativen Prozesse finden in der Regel als Selbstorganisationsprozesse statt. Selbstorganisationsprozesse lassen sich nicht direkt sondern nur über seine Randbedingungen beeinflussen. Der Umgang mit solchen Prozessen erfordert ein evolutionäres Herangehen.

Fazit:

In diesem Sinne kann Softwareentwicklung als ein Transformationsprozess, der in einen Selbstorganisationsprozess eingebettet ist, interpretiert werden.

Lit:

Coy, W. (Hrsg.), Sichtweisen der Informatik Braunschweig, Wiesbaden: Vieweg 1992

Dahme, Ch.; Raeithel, A.: Ein tätigkeitstheoretischer Ansatz zur Entwicklung von brauchbarer Software, in:

¹² im Sinne des tätigkeitstheoretisch orientierten Ansatzes

¹³ im Sinne des gegenstandsorientierten Modellansatzes

¹⁴ s. u.a. Dahme, Ch.; Raeithel, A. 1997, S. 5

- Informatik-Spektrum · 20 · Heft 1 · Februar 1997,
 (<http://waste.informatik.hu-berlin.de/Dahme/Da%2FRa.html>)
- Dahme, Ch.: Systemanalyse menschlichen Handelns - Grundlagen und Ansätze zur Modellbildung, Westdeutscher Verlag 1997, Opladen
- Dahme, Ch.: Wissenschaftstheoretische Positionen in bezug auf die Gestaltung von Software. In: Fuchs-Kittowski, K. / Parthey, H. / Umstätter, W. / Wagner-Döbler, R. (Hrsg.): Organisationsinformatik und Digitale Bibliothek in der Wissenschaft: Wissenschaftsforschung Jahrbuch 2000. Berlin: Gesellschaft für Wissenschaftsforschung 2001. S. 167-178
- F. Nake: Informatik und Maschinisierung von Kopfarbeit, in: Coy, W. (Hrsg.), Sichtweisen der Informatik Braunschweig, Wiesbaden: Vieweg 1992
- F. Nake (Hrsg.), Die erträgliche Leichtigkeit der Zeichen, Agis Verlag, Baden-Baden 1993
- Raeithel, A., Selbstorganisations, Kooperation, Zeichenprozess. Hrsg. v. Ch. Dahme. Opladen: Westdeutscher Verlag 1998.
- Software Development and Reality Construction. Hrsg. v. C. Floyd et al. Berlin: Springer 1992.
- Dörner, Dietrich: Problemlösen als Informationsverarbeitung. Stuttgart etc. : Kohlhammer, 1987. – 3. Aufl.
- Dörner, Dietrich ; Bick, Thomas: Lohhausen vom Umgang mit Unbestimmtheit und Komplexität. Bern etc. : Hans Huber, 1983. – 2. Aufl.
- Klix, Friedhart: Information und Verhalten. Berlin : Dt. Verlag der Wissenschaft, 1971. – 4. Aufl
- Klix, Friedhart: Erwachendes Denken eine Entwicklungsgeschichte der menschlichen Intelligenz. Berlin : Dt. Verlag der Wissenschaft, 1980. – 1. Aufl.
- Newell, Allen ; Simon, Herbert A.: GPS, A Program that Simulates Human Thought. In: Billing, H. (Hrsg.): Lernende Automaten. München : Oldenbourg, 1961, S. 109–124
- Newell, Allen ; Simon, Herbert A.: Human problem solving. Englewood Cliffs : PrenticeHall, 1972
- Newell, Allen ; Simon, Herbert A.: The Theory of Human Problem Solving. In: Collins, A. (Hrsg.) ; Smith, E. E. (Hrsg.): Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence. San Mateo : Kaufmann, 1988, S. 33–51
- Turing, A. M.: Computing Machinery and Intelligence. In: Collins, A. (Hrsg.) ; Smith, E. E. (Hrsg.): Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence. San Mateo : Kaufmann, 1988, S. 6–19