

Von der MOPS zur Software-Technologie

Stationen der Software-Entwicklung in der DDR

GÜNTHER BAUER

guenther.bauer@web.de

In dem Beitrag wird die Herausbildung der Software-Technologie in der DDR skizziert. Beginnend bei der R300-Sprache MOPS werden wesentliche Stationen wie GOTO-Freiheit, interaktive Programme und Daten-Strukturierung diskutiert.

1 Die Ära der Programmiersprachen

Setzen wir den R300 als Startpunkt für den flächendeckenden Einsatz von EDV, so ist mit MOPS (Maschinen-orientierte Programmiersprache) der Beginn der breiten Programmierung markiert. Autoren von der HU wie Lemgo, Paulin und Tschirschwitz sind an der Startlinie zur Verbreitung von MOPS mit auszumachen (Verlag Wirtschaft 1970). Die Programmierung liegt auf der Stufe der Assembler-Sprachen und den damit gegebenen Möglichkeiten. Dennoch wurde damit z. B. bereits 1970 für einen eigens um einen Multiplexkanal erweiterten R300 ein Multi User System für das Training von Managern aus Betrieben und Kombinatn entwickelt (Trainingszentrum der Akademie der marxistisch-leninistischen Organisationswissenschaften der DDR in Berlin-Wuhlheide). Eine für uns beteiligte Programmierer seinerzeit völlig neuartige und auch etwas anspruchsvollere Aufgabe.

Programmiersprachen sind die notwendige Voraussetzung zur Nutzung von Rechnern. An die damit verbundenen Aufgaben erinnert Abbildung 2. Grundlagen formaler Sprachen und sich herausbildende Rechnerarchitekturen brachten Akademiker und Praktiker auf dem zu dieser Zeit sehr belebten Feld des Compilerbaus zusammen.

1968 wurde die Arbeitsgruppe „Programmiersprachen und Übersetzertechnik“ unter Leitung von I. O. Kerner als Forschungsgruppe „Programmiersprachen“ gegründet. (s. ALGOL68, Riedewald Informatik Uni Rostock). Und, Kerner war von 1965 bis 1983 Mitglied der IFIP-WG 2.1 ALGOL.

In den Studierstuben wurde gern mit PASCAL, MODULA, ADA oder gar Oberon experimentiert; in der Praxis wurde PL/1 oder auch FORTRAN eingesetzt.

Die Bemühungen, ALGOL68 für den Programmier-Alltag verfügbar zu machen, waren bekanntermaßen nicht erfolgreich. Sicherlich eine gute Vorlage für die Diskussion zur häufig aufgegriffenen These: „Nichts ist praktischer als eine gute Theorie“.

Sprachen für EDVA		
Rechner	Sprache	Referenz-Beispiele
R300 (1968)	MOPS MACRO-MOS ALGOL 60	Lemgo,K. Tschirschwitz,R. Einführung in die Programmierung des Robotron 300 Verlag Technik 1969
R21 (1971)	Assembler RPG FORTRAN (Basis) PL/1 (Subset)	Kubisch,W. Witschurke,R. ALGOL 60 für Robtron 300 Die Wirtschaft 1970
EC 1040 (1972)	... FORTRAN IV PL/1 COBOL	Bär,D. Grundstufe COBOL-Progrg Verlag Technik 1968
EC 1055 (1979)	... FORTRAN ALGOL PASCAL CDL	Paulin,G. FORTRAN: Kodierung von Formeln Verlag technik 1968
EC 1057 (1987)	... FORTRAN 77 MODULA 2	Stempell,D. Einführung PL/1 Verlag Technik 1971
		Forbrig,P: Progrg mit Pascal Verlag Technik 1972

Abbildung 1: Programmiersprachen für R300 und ESER

Das japanische Programm zur Entwicklung von Rechnern der fünften Generation und entsprechender Software führte dann noch zu einem (kurzzeitigen) Höhenflug der Sprachen LISP und PROLOG in der DDR.

Heutzutage sind Übersetzer (für Standardsprachen) weit verfügbar. Und Binär-Code wie bei Java macht es möglich, einmal geschriebene Quellprogramme auf ganz unterschiedlichen Prozessoren wie in einem Handy oder auf den Rechnern eines A380 auszuführen.

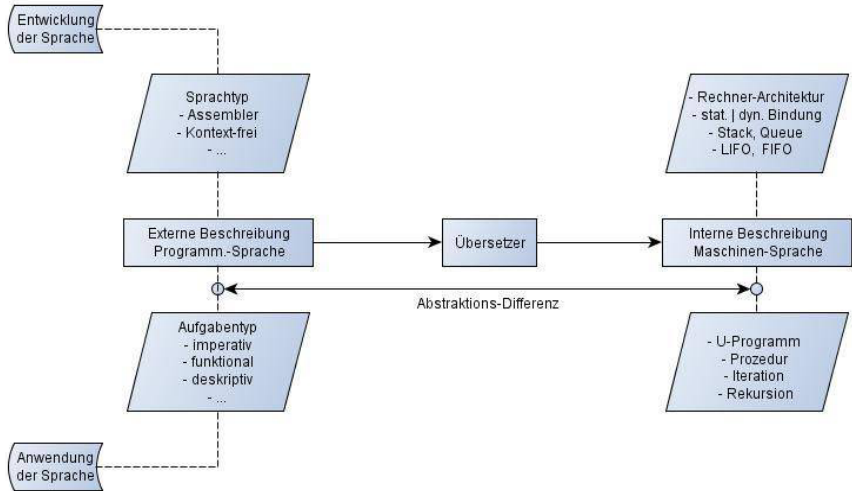


Abbildung 2: Hauptaufgabe Programmiersprachen

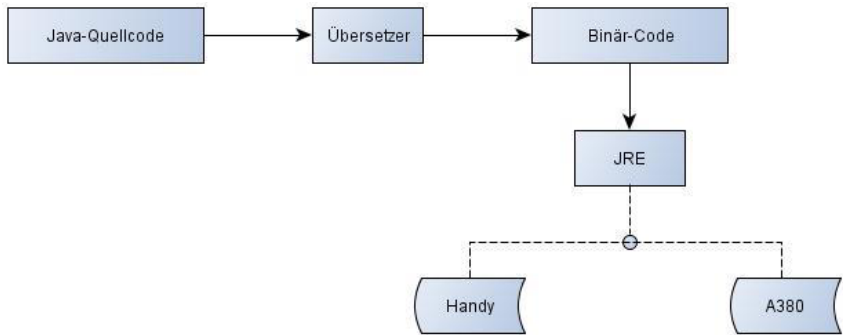


Abbildung 3: Eine Sprache – unterschiedliche Architekturen

2 Die Automatisierungswelle

In den späten 60er und Anfang der 70er Jahre machte sich in der DDR eine gewisse Euphorie bezüglich der Möglichkeiten der Automatisierung breit.

So wurde z. B. für den Flughafen Schönefeld die Vision verfolgt, eine durchgängig automatisierte Gepäckabfertigung bereitzustellen, wie sie in Denver (Denver International Airport, Baubeginn 1988, Fertigstellung 1993) – übrigens unter großen Anlaufschwierigkeiten – erst in den Jahren 1993 bis 1995 in Betrieb genommen wurde. (Dieser Vergleich lässt die Komplexität einer solchen Aufgabe erahnen.)

Die Mathematiker (der Akademie der DDR) rechneten 'Große Systeme'. „Ich wurde in eine Kommission Große Systeme berufen. ... Ich kann mich an kein einziges vernünftiges Resultat erinnern. Dabei lag ein Negativergebnis auf der Hand. ... Nämlich, dass eine zentrale Steuerung eines so großen Systems wie es eine Volkswirtschaft war, nicht funktionierte.“ (Helga Königsdorf „Landschaft in wechselndem Licht“, AtV 2005, S. 145)

Die Software-Technologie stellte zunächst R300-Typenprojekte (Zentralinstitut für Automatisierung Dresden, später Institut für Datenverarbeitung Dresden) und dann Problemorientierte Systemunterlagen (POS) bereit.

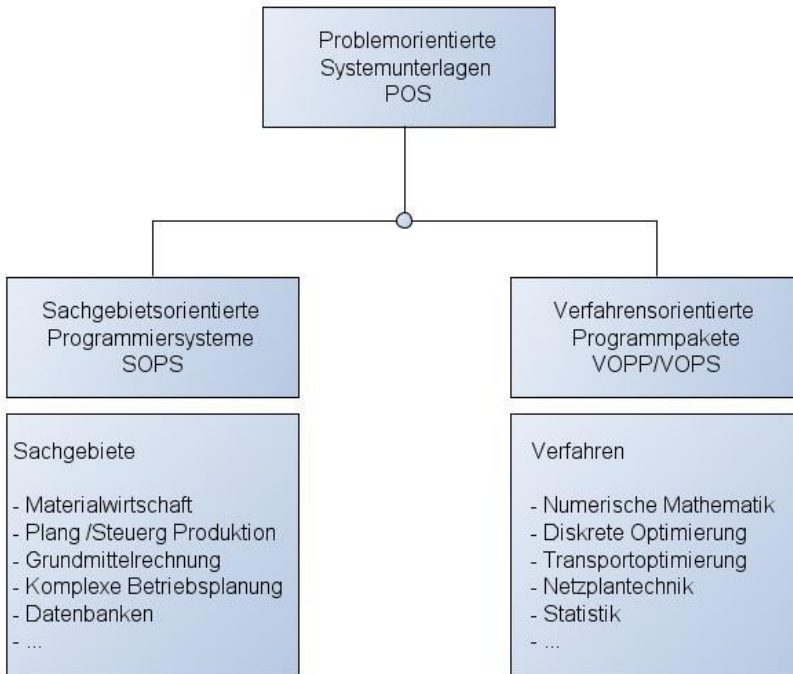


Abbildung 4: Übersicht POS von Robotron (s.d. rt/dv Beiheft 1, 1972)

Derartige Software wurde zunächst vor allem in MOPS geschrieben (Ende 1968 305 Mitarbeiter des idv). Die Linie Typenprojekte wurde aufgrund von Realisierungsproblemen nicht bis zum Ende verfolgt. Als wesentliche Schlussfolgerung wurde festgehalten: „Allgemeingültige und variable Software muss die Problemvariabilität gewährleisten sowie ...“ (Merkel und Junge 2006, S. 3) SOPS wurden 1968 bis 1974 (DOS bzw. DOS/ES und OS/ES); VOPP/VOPS von 1969 bis 1989 entwickelt. (Am Rande sei vermerkt: Das Rechenzentrum

für den Airport BBI wurde im Juli 2010 mit rund 500 Servern, deren Leistungsvermögen dem von etwa 8000 PC entspricht, fertig gestellt.)

3 Die Software-Krise

Bereits in den 60er Jahren wurden die Programmentwickler weltweit unruhig. Die Programme wurden lang und länger, sie wurden zu komplex. Die Programmierer waren den Anforderungen nicht mehr gewachsen – es gab eine Software-Krise. Die wesentliche Ursache ging zurück auf eine Beobachtung von Dijkstra: Die Zahl der Fehler in einem Programm ist direkt proportional der Anzahl der GOTOs.

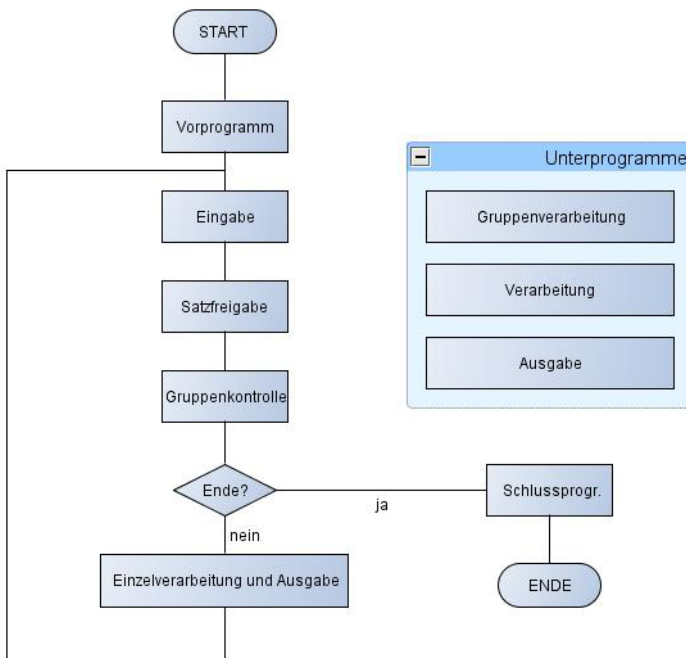


Abbildung 5: Schema Normierte Programmierung

Die erste Schlussfolgerung lautete deshalb: GOTO-Freiheit. Insgesamt war eine verbesserte Systematik bei der Konstruktion von Programmen erforderlich. Denn, so wurde gezeigt: Gut strukturierte Programme haben weniger Fehler und sind leichter zu warten (Gesetz von Dijkstra, Mills & Wirth, vgl. Endres und Rombach 2003). O. Herrlich (TU Dresden) setzte sich ein für die Normierte Programmierung. Ihr Prinzip: Verallgemeinerung bestimmter Pro-

grammabschnitte und deren Zuordnung zu generalisierten Aufgaben. Als wesentliches Merkmal derartiger Programme wird hier vermerkt: Die Steuerung des Ablaufs liegt beim Programm (d. h. einmal angestoßen, kann der Benutzer nicht mehr eingreifen).

Heutige Programmiersprachen wie JAVA haben keine (expliziten) Sprunganweisungen mehr.

4 Wechsel zu interaktiven Programmen

Die wachsende Zahl von Funktionen, die in einem Programm realisiert wurden und die damit verbundene Variabilität bei der Gestaltung des Arbeitsablaufs machten einen Paradigmenwechsel erforderlich: Die Ablaufsteuerung wurde vom Programm an den Benutzer verlagert. Die Konstruktion interaktiver Programme folgt dem in Abbildung 6 skizzierten Prinzip.

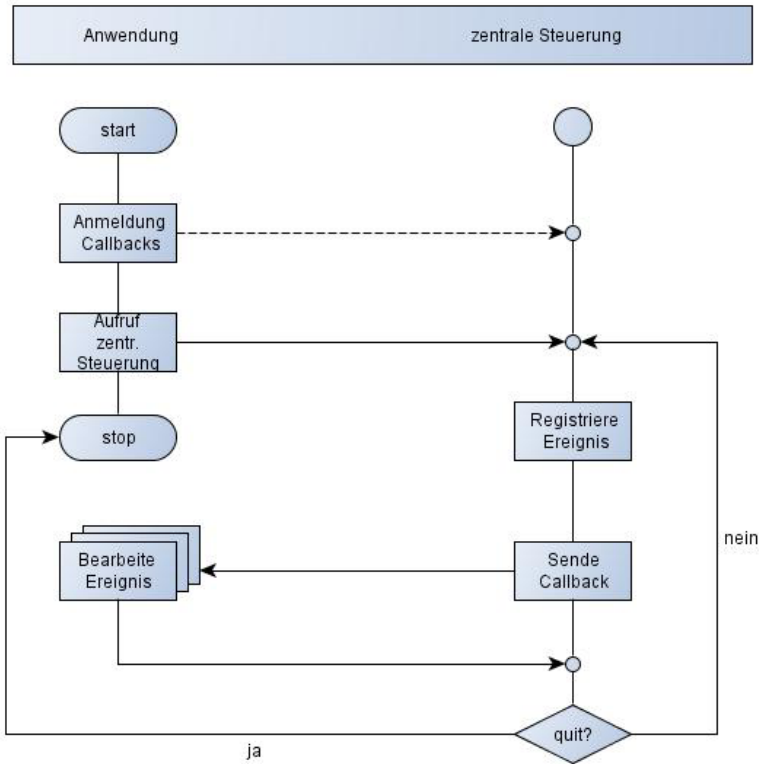


Abbildung 6: Schema interaktives Programm

Mit dem MVC-Paradigma (1979) entstand ein grundlegendes Architekturmuster für die Implementierung der Interaktion. Die nunmehr völlig neuartigen Benutzungsoberflächen brachten zusätzliche Anforderungen an die Programmierer mit sich. Zum einen in Bezug auf weitere Komponenten im Programm, zum anderen hinsichtlich der Berücksichtigung psychologischer Faktoren der Benutzer wie Belastbarkeit (des Kurzzeitgedächtnisses), sichere Führung etc. Die Software-Ergonomie musste entwickelt werden. Die Sektion Psychologie der HU Berlin (H. Wandke) z. B. lieferte zahlreiche Beiträge dazu. Und mit dem CUA (Common User Access) von IBM im Jahre 1987 entstanden die grundlegenden Bausteine für unsere heutigen Oberflächen und führte letztlich auch zum Prinzip des WYSIWYG. Natürlich darf das Design einer Oberfläche auch elegant sein. Die weitaus tiefer gehende Problematik ist jedoch die Gestaltung von Arbeit beim Einsatz von Rechnern. Die Sektion WTO (HU Berlin) und die Arbeitswissenschaftler der TU Dresden (W. Hacker) waren treibende Kräfte bei der Entwicklung der Gedanken zu Partizipation des Anwenders und zu Arbeits- sowie Organisationsgestaltung für EDV-Anwendungen. Parallel dazu entwickelte sich eine – auch philosophisch basierte – Diskussion über den Stoff, der da maschinell bearbeitet werden sollte: Information. HU Berlin und Akademie der Wissenschaften mit Protagonisten wie Fuchs-Kittowski, Wenzlaff, Klix oder Völz waren Zentren der Ideen-Entwicklung. Im Moment findet man die wesentlichen Elemente der Software-Ergonomie unter dem Begriff *Usability*.

5 Die Daten-Strukturierung

Programmierer haben in der Zeit der Programmiersprachen besonders gern in Stacks und Queues, in LIFO und FIFO etc. gedacht. Aber auch bei externer Sicht auf zu verarbeitende Daten in Form des *Datensatzes* zeigten sich Schwierigkeiten bei der Beschreibung von Objekten der Realität. Beim Betrachten eines Gebildes der Art

```
NAME  VORNAME  NUMMER  BETRAG
```

sind selbst viele Fachleute der (unzutreffenden) Meinung, man hat eine Daten-Struktur vor sich. Die vorliegende Form gibt u. a. keine Auskunft über die Bedeutung (Semantik). Ordnet man einzelnen Elementen Bedeutung zu, wird der Raum für mögliche Assoziationen eingeengt. Setzen wir z. B.

```
NAME VORNAME  ---> Person
NUMMER          ---> Kontonummer
BETRAG         ---> Summe
```

so meinen wir sofort, die EDV-technische Beschreibung eines Bankkontos und seines Inhabers vor uns zu haben. (Es könnte aber auch der Bearbeiter eines Kontos beschrieben sein.) Erhält NUMMER die Bedeutung Versicherungs-

nummer, ist die Semantik unseres Gebildes sofort völlig anders. Somit wird deutlich: Ein und dieselbe Struktur repräsentiert zwei unterschiedliche Objekte (und weitere sind leicht denkbar). Und was ist dabei die Daten-Struktur? Es gab also einigen Klärungsbedarf, und so wurde neben der 'gängigen' Beschäftigung mit Datenbanken (BASTEI, DBS/R, ...) auch tiefer über das Wesen von Daten und deren Strukturierung nachgedacht (z. B. Fuchs-Kittowski u. a. 1976). Der Ansatz zur Lösung des angegebenen Problems ist, eine präzise Beschreibung des Terminus *Datenstruktur* herauszuarbeiten.

Das Ergebnis ist in der Abbildung 7 dargestellt. Gewonnen haben wir

- Definition: Daten-Struktur ist die Menge der (invarianten) Beziehungen zwischen Daten-Elementen.
- Erklärungsmodell: WAS ist eine (Aufgaben-bezogene) Datenstruktur.
- Konstruktionsregeln: WAS muss festgelegt werden.

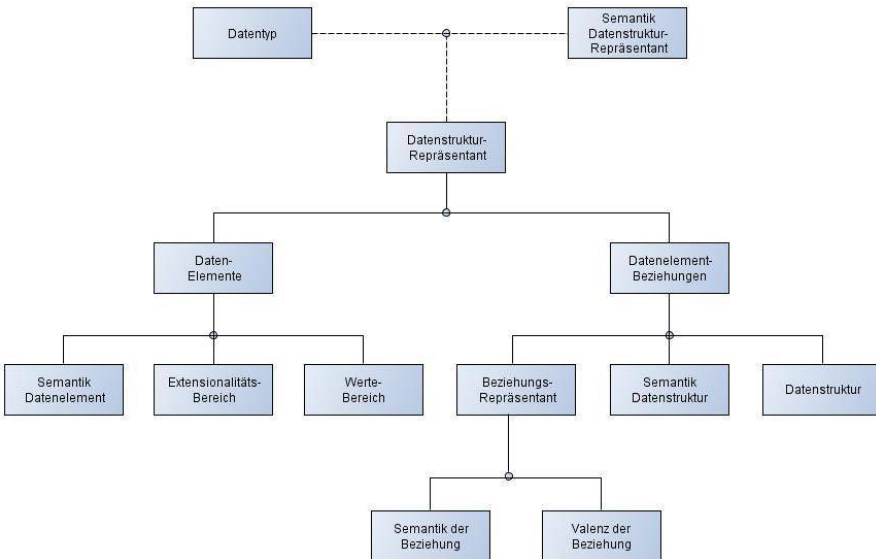


Abbildung 7: Begriffsbestimmung 'Datenstruktur' (Bauer 1986)

Verwendet man nun die mathematisch bestimmten Eigenschaften der Beziehungen (Relationen) zwischen den Daten-Elementen, so lässt sich folgende grundlegende Systematik von Daten-Strukturen herleiten (Bauer 1985, 1988).

Datenstruktur-Typ	Kategorie der Relation
Lineare Ordnung - Sequenz - Tupel - Vektor	Lineare Ordnungsrelation
Zerlegung einer Menge - einfache Zerlegung - genestete Menge - Baum	Äquivalenzrelation
Graph - azyklisch - allgemeine Liste	Gesamtwertebereich-Relation
Assoziative Struktur	unbeschränkte Relation

Nimmt man nun eine Notationsform hinzu, wie sie z. B. später mit XML (1998) entstanden ist, so wird exakt vorgegeben, WIE eine Objektbeschreibung mittels Daten auszuarbeiten ist.

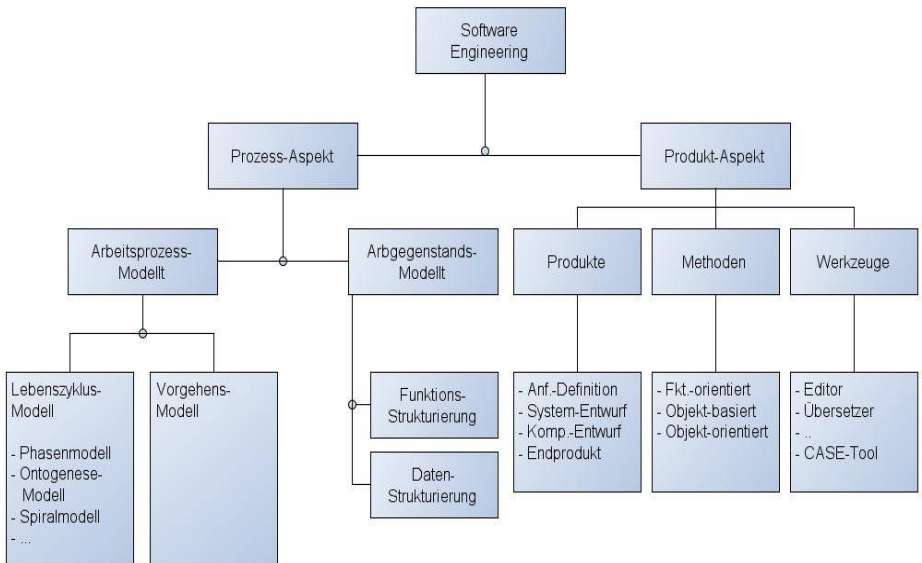


Abbildung 8: Struktur-Bild Software Engineering (Software-Technologie)

Werden Einsichten solcher Art auf die o. a. Datenstruktur-Repräsentanten *Konto* bzw. *Versicherung* angewandt so zeigt sich, dass zwei Mal die gleiche Datenstruktur, nämlich die lineare Sequenz, vorliegt.

6 Software-Technologie

Zunächst galt es, den Begriff *Software-Technologie* sowohl intentional als auch extentional auszuarbeiten. Dabei wurde u. a. sehr auf Kompatibilität zum altherwürdigen Begriff *Technologie* geachtet. Mit dem Strukturbild wurde ein entsprechendes Ergebnis vorgelegt (Bauer 1989). Eine weiterentwickelte Variante davon wird in Abbildung 8 gezeigt.

Die Zweiteilung in Prozess- und Produkt-Aspekt ist ein geeigneter Ausgangspunkt, um Produktions- und Konstruktions-Systematik auszuarbeiten. So wurde an der HU Berlin das sogenannte Ontogenese-Modell entwickelt. Ontogenese durchaus im Sinne der individuellen Entwicklung eines Projekts.

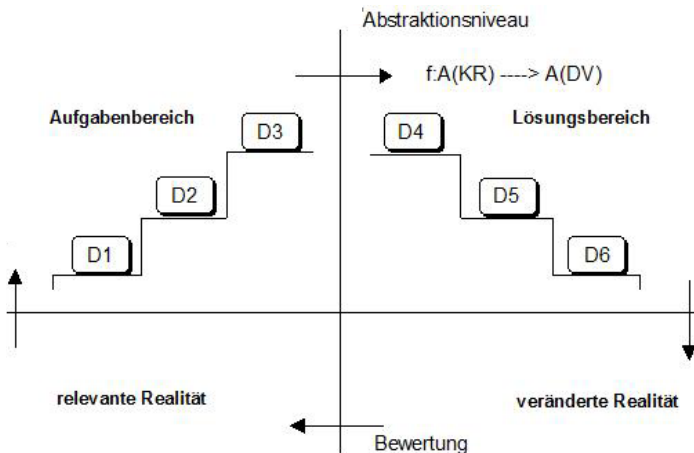


Abbildung 9: Ontogenese-Modell (Bauer 1989)

Dieses Modell hat folgende Vorzüge:

- Vier deutlich unterschiedene Bereiche, über die „nachgedacht“ werden kann.
- Wechsel von einem Bereich zu einem anderen ist zu jedem Zeitpunkt möglich, also wird z. B. keine starre Abfolge verlangt.
- Die Vorstellung der Modellierung wird mit dem Aufgabenbereich besonders unterstützt durch die stufenweise Formalisierung und Erhöhung der Abstraktion von der relevanten Realität.
- Die Unterscheidung zwischen relevanter und veränderter Realität hebt deutlich hervor, dass mit der Entwicklung von Anwendungen die Wirklichkeit verändert, also Systemgestaltung vorgenommen wird.

- Das Modell lässt sich zuschneiden (konkretisieren) auf unterschiedliche Entwicklungsszenarien.

Die Software-Technologie (bzw. das Software Engineering) ist heutzutage eine respektabel ausgebaute wiss. Disziplin. Umso erstaunlicher, dass noch immer Softwareprojekte völlig missraten. Jüngstes Beispiel: Das SIS II (Schengener Informationssystem) kommt mindestens zehn Jahre später und wird wohl mit ca. 150 Mio. Euro zehnmal teurer als geplant (vgl. MOZ). Gegenwärtig liegt der Fokus des Interesses auf Software-Architektur.

7 Literatur

- [1] BAUER, G. (2009): Architekturen für Web-Anwendungen. *Vieweg + Teubner*.
- [2] ENDRES, A. & ROMBACH, D. (2003): A Handbook of Software and Systems Engineering. *Pearson Education Ltd*.
- [3] FUCHS-KITTOWSKI, K.; KAISER, H.; TSCHIRSCHWITZ, R. & WENZLAFF, B. (1976): Informatik und Automatisierung. *Berlin*.
- [4] HUMBOLDT-UNIVERSITÄT ZU BERLIN, ORGANISATIONS- UND RECHENZENTRUM (Hrsg.) (1985): Systematik der Prinzipien der Datenstrukturierung. *Informatik-Skripten 1*.
- [5] HUMBOLDT-UNIVERSITÄT ZU BERLIN, ORGANISATIONS- UND RECHENZENTRUM (Hrsg.) (1986): Datenstruktur – eine zentrale Kategorie der Softwaretechnologie. *Informatik-Skripten 2*.
- [6] HUMBOLDT-UNIVERSITÄT ZU BERLIN (Hrsg.) (1988): Technologische Betrachtung der Datenstrukturierung. *Diss. B*.
- [7] HUMBOLDT-UNIVERSITÄT ZU BERLIN, ORGANISATIONS- UND RECHENZENTRUM (Hrsg.) (1989): Grundlagen der Software-Technologie. *Informatik-Skripten 9*.
- [8] ENDRES, A. & ROMBACH, D. (2003): A Handbook of Software and Systems Engineering. *Pearson Education Ltd*.
- [9] FUCHS-KITTOWSKI, K.; KAISER, H.; TSCHIRSCHWITZ, R. & WENZLAFF, B. (1976): Informatik und Automatisierung. *Berlin*.
- [10] (1972): Kurzbeschreibung der Systemunterlagen. *rt/dv 1. Beiheft, S. 20-65*.
- [11] MERKEL G. & JUNGE, S. ET AL. (2006): Sammlung von Beiträgen zur Geschichte der Zentralen Forschungs- und Entwicklungseinrichtung des VEB Kombinat Robotron. *Sprachen und Compiler für EDVA*.
- [12] MOZ (04.06.2010). *Märkische Oderzeitung, S. 4*.