Kurbel, Karl

Working Paper

Das technologische Umfeld der Informationsverarbeitung: Ein subjektiver State of the Art-Report über Hardware, Software und Paradigmen

Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 2

Provided in Cooperation with:
University of Münster, Department of Information Systems

Suggested Citation: Kurbel, Karl (1991) : Das technologische Umfeld der Informationsverarbeitung: Ein subjektiver State of the Art-Report über Hardware, Software und Paradigmen, Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 2, Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, Münster

This Version is available at:
https://hdl.handle.net/10419/59320

Arbeitsbericht Nr. 2

Das technologische Umfeld der Informationsverarbeitung

Ein subjektiver "State of the Art"-Report über Hardware, Software und Paradigmen

Prof. Dr. Karl Kurbel

unter Mitarbeit von

Dipl.-Inform. Stefan Eicker
Dipl.-Inform. Michael Nietsch
Dipl.-Inform. Thomas Nietsch
Dipl.-Kfm. Wolfram Pietsch
Dipl.-Inform. Claus Rautenstrauch

Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster,
Grevener Straße 91, 4400 Münster, Tel. (0251) 83-9750, Fax (0251) 83-9754

März 1991

Inhalt

Zusammmenfassung

Der Arbeitsbericht ist als Orientierungshilfe gedacht. Er beschreibt den Stand der Entwicklung im Hardware- und Softwarebereich der Großrechner-, Workstation- und PC-Welt. Neben der Rechnerhardware werden vor allem Betriebssysteme, Benutzerschnittstellen, Netze und Datenbanksysteme unter die Lupe genommen. Darüber hinaus werden zukunftsweisende Entwicklungen im Software Engineering und in der Künstlichen Intelligenz aufgezeigt.

1 Zielsetzung des Berichts

Angesichts des schnellen Fortschritts und Wandels im Umfeld der Wirtschaftsinformatik - laufend werden neue Dimensionen der Rechnerunterstützung erschlossen und neue Begriffe geprägt - besteht vielerorts ein Informationsdefizit. Im vorliegenden Fall wurde der Wunsch nach Orientierungshilfe im Kreis der betriebswirtschaftlichen Hochschullehrer an der Westfälischen Wilhelms-Universität Münster artikuliert.

Die Aussagen in diesem Beitrag über den gegenwärtigen Stand der Rechnerunterstützung und die Entwicklungstendenzen basieren teilweise auf Recherchen und teilweise auf subjektiven Einschätzungen, die aus eigenen Erfahrungen, Marktbeobachtung und Know-how der Mitarbeiter des Instituts für Wirtschaftsinformatik resultieren. In diesem Sinne handelt es sich nicht um eine wissenschaftliche Arbeit, sondern um einen subjektiven Versuch, etwas Licht in den Dschungel von Begriffen, Ankündigungen, Entwicklungen etc. zu bringen und damit einen Beitrag zu leisten, Informationsdefizite abzubauen.

Im folgenden werden die Bereiche

- Hardware
- Betriebssysteme
- Benutzerschnittstellen
- Netze
- Datenbanksysteme
- Softwareentwicklung
- Künstliche Intelligenz,

teilweise differenziert nach

- Großrechner- und Midrangesystemen
- Workstations
- Personal Computern,

erörtert. Eine orthogonale Einteilung der zu behandelnden Rubriken läßt sich allerdings kaum aufstellen. Manche Themenschwerpunkte sind eher an der Rechnerebene orientiert, während andere sich quer durch alle Rechnerkategorien ziehen (z.B. Benutzerschnittstellen), und wieder andere (Softwareentwicklung, Künstliche Intelligenz) liegen auf einer höheren Abstraktionsebene, so daß der Bezug zu Rechnerkategorien keine Rolle spielt. Die gewählte Einteilung ist deshalb ausschließlich durch die Praktikabilität der Darstellung begründet.

Beim Recherchieren und Zusammentragen des Informationsmaterials waren die o.g. wissenschaftlichen Mitarbeiter des Lehrstuhls für Wirtschaftsinformatik maßgeblich beteiligt. Dafür sei ihnen herzlich gedankt. Besonderer Dank gebührt Bettina Gerber, die als studentische Hilfskraft in der heißen Phase der Prüfungsvorbereitungen

eine Woche lang jede freie Minute mit dem Anfertigen der Grafiken verbrachte.

Der vorliegende Bericht repräsentiert den Stand der Erkenntnisse im Februar 1991.

2 Die "PC-Welt"

2.1 PC-Betriebssysteme

Die ersten Kleinrechner, die in die Kategorie der Personal Computer fielen (Apple II, Tandy TRS 80, Commodore Pet u.a.), wurden meist mit dem Betriebssystem CPM gefahren. Dessen Verbreitung ging Anfang der achtziger Jahre zurück, als IBM das Potential des PC-Markts entdeckte und für sein eigenes Produkt das Betriebssystem DOS übernahm. Wenn auch DOS den Anforderungen an ein "gutes" Betriebssystem in vielfacher Hinsicht nicht genügte, so bewirkte die Marktmacht von IBM einen enormen Aufschwung, wie auch aus Abbildung 1 ersichtlich ist. Obwohl auch damals schon Unix existierte und in Insiderkreisen als das wesentlich bessere Betriebssystem galt, konnte es bis heute mit dem Aufschwung von DOS nicht konkurrieren.

Abb. 1: Betriebssysteme für Personal Computer

Dies wird sich in der Zukunft möglicherweise ändern. Mit der Verfügbarkeit leistungsfähigerer Prozessoren und größerer Speicherkapazität wird wahrscheinlich Unix auch im PC-Bereich stärker vordringen[1)]. Die Überlebensfähigkeit von DOS basiert einerseits darauf, daß es bereits eine sehr große Zahl von Anwendungsprogrammen gibt, die unter DOS laufen. Zum anderen baut die Betriebssystemerweiterung MS-Windows auf DOS auf. Die einfache, kommandoorientierte Benutzerschnittstelle von

[1)] Zur Relativierung der Aussage sei aber angemerkt, daß diese Erwartung schon seit vielen Jahren artikuliert wird.

DOS wird von MS-Windows ver deckt. Der Umgang mit MS-Windows ist komfortabler, ergonomischer und wesentlich einfacher. MS-Windows hebt außerdem einige der gravierenden Restriktionen von DOS auf, so daß auch größere Programme (> 640 KB) und ein "Pseudo-Multitasking" möglich sind[2)].

Der Aufschwung des Betriebssystems OS/2, das IBM mit seiner Rechnergeneration Personal-System/2 (PS/2) ankündigte, ist bislang ausgeblieben. OS/2 in der Version 2.0 (für 80386 Prozessoren) ist im Prinzip ein sehr leistungsfähiges Betriebssystem, das MS-Windows deutlich überlegen ist. Es nutzt die Möglichkeiten des 80386-Prozessors voll aus, unterstützt echtes Multitasking und enthält ein stark verbessertes Dateisystem.

Die historischen und wohl auch zukünftigen Probleme von OS/2 sind in der schleppenden Verfügbarkeit begründet. Die Version 2.0 ist für 1991 angekündigt. Man kann erwarten, daß der Marktanteil von OS/2 angesichts der wachsenden Verbreitung von Unix und der breiten MS-Windows/DOS-Basis um so niedriger ausfallen wird, je länger es dauert, bis es als leistungsfähiges System allgemein verfügbar ist und die großen Softwarehersteller (Borland, Ashton Tate, Lotus, Microsoft etc.) ihr ganzes Produktspektrum auch unter OS/2 anbieten.

2.2 PC-Hardware

Das Anwachsen der Leistungsfähigkeit von Prozessoren, Massenspeichern und Grafikzusätzen ist in Abbildung 2 skizziert. Bei den Prozessoren wird als Maßeinheit MIPS ("million instructions per second") verwendet. Im Vergleich zu der Prozessorleistung der ersten PCs (Intel-Prozessoren 8086 und 8088), z.B. des IBM-PC XT, die deutlich unter einem MIPS lag, ist der heutige Stand durch 4 - 6 MIPS-Systeme gekennzeichnet.

Die Verarbeitungsgeschwindigkeit eines bestimmten Prozessortyps (z.B. 80386) ist nicht konstant, sondern hängt von verschiedenen Faktoren (Taktfrequenz, Effektivität der I/O-Operationen, Vorhandensein von schnellen Cache-Speichern und Coprozessoren etc.) ab; sie kann somit über und unter den angegebenen Richtwerten liegen. Beispielsweise gibt es in der 80386-Welt, in der die Prozessoren heute meist mit 33 Megaherz getaktet sind, auch Hochleistungsrechner mit 70 Megaherz und sehr hohen MIPS-Raten. Bei der Prozessorleistung scheint ein "Quantensprung" bevorzustehen. Ersten Ankündigungen zufolge soll noch 1991 der 80586-Prozessor von Intel auf den Markt kommen, dessen Verarbeitungsgeschwindigkeit angeblich in der Größenordnung von 100 MIPS liegen wird.

[2)] Das Multitasking von MS-Windows unterscheidet sich von einem echten Multitasking dadurch, daß die Kontrolle über den Wechsel von einem Programm zum anderen mit der Zuteilung von Prozessorleistung nicht beim Betriebssystem (DOS), sondern bei Windows bzw. beim Anwendungsprogramm(ierer) liegt. Dies stellt bei unsachgemäßer Handhabung eine Fehlerquelle dar.

Abb. 2: Entwicklungstendenzen im Bereich der PC-Hardware

Trends, die in der Grafik nicht dargestellt wurden, sind einerseits die zunehmende Orientierung an RISC-Architekturen ("reduced instruction set computer"), die im Workstation-Bereich bereits weite Verbreitung besitzen, sowie vereinzelte Bestrebungen in Richtung auf Mehrprozessorsysteme, die echte Parallelverarbeitung erlauben. Das Problem bei den letzteren besteht allerdings darin, daß die notwendige Software, die die Mehrprozessorarchitektur ausnützen könnte, bislang noch kaum existiert.

Im Bereich der Massenspeicher setzte eine Zeitlang der IBM-PC XT den Maßstab mit einer 10-MB-Platte. Gegen Ende der achtziger Jahre waren Festplatten mit 80-MB-Speicherkapazität Standard, während heute die "Normalausstattung" bei 150-MB-Platten liegt. Neben der konventionellen Magnetplattentechnologie - bereits heute sind Speicherkapazitäten von 300 und 600 MB nicht außergewöhnlich - werden in der Zukunft auch ***optische Platten*** mit Speicherkapazitäten im Gigabitbereich das Spektrum ergänzen. Darüber hinaus sind sehr schnelle Chip-Speicher im Massenspeicherbereich zu erwarten; dies sind etwa scheckkartengroße Platinen, die sich wegen der kleinen Abmessungen und des geringen Gewichts vor allem für Laptops eignen.

Der Übergang von der alphanumerischen Darstellung zur Grafik auf dem Bildschirm ist eng mit der sogenannten "Herkuleskarte" verbunden. Mit dem Colour Graphics Adapter (CGA-Karte) kam die Farbe hinzu. Seitdem sind Verbesserungen vor allem bezüglich der Auflösung (Zahl der Bildpunkte) und der Zahl der möglichen Farben eingetreten. Während die CGA-Grafik sich auf 320 x 200 Bildpunkte beschränkte, sind bei

der Super-VGA-Grafik heute 1024 x 768 Bildpunkte mit 256 Farben adressierbar.

Die Weiterentwicklung der reinen Bildschirmadapter geht zu eigenständigen Grafikprozessoren hin, welche auch Funktionen wie das Fenstermanagement und die Berechnung der Bildschirmkoordinaten übernehmen. Am Rande sei darauf hingewiesen, daß diese im Bereich des sogenannten Industriestandards "neue" Entwicklung bei den Apple-Systemen schon seit vielen Jahren realisiert ist.

Soweit es sich heute erkennen läßt, wird der PC der Zukunft durch eine sehr hohe Prozessorleistung (80586-Prozessor) gekennzeichnet und möglicherweise ein Mehrprozessorsystem sein. Im Bereich des Arbeitsspeichers werden in nächster Zeit Kapazitäten von 32 - 64 MB erwartet. Über spezielle Prozessoren wird Hochleistungsgrafik, die bisher den Workstations vorbehalten ist, auch auf den PCs verfügbar werden. Optische Plattenspeicher mit hoher Kapazität werden in nächster Zukunft nur zum Abrufen von Informationen ("read only") verwendet werden können. Mittelfristig ist jedoch auch im PC-Bereich mit auswechselbaren, wieder beschreibbaren optischen Platten zu rechnen. Als Vorreiter gilt der ***NeXT-Rechner***, der von Apple-Gründer Steven Jobs nach seinem Ausscheiden bei Apple entwickelt wurde. Dieser ist bereits heute mit einer optischen 600 MB-Platte ausgestattet.

Die Benutzerschnittstelle des PCs wird sich durch ergonomische objektorientierte Fensteroberflächen weiter verbessern. Die technischen Möglichkeiten für multimediale Darstellung (Bilder, Bewegtbilder, Ton) sind bereits heute verfügbar, so daß sich auch der Einsatzbereich von PCs stark erweitern wird.

2.3 PC-Software

Das breite Spektrum der PC-Software kann an dieser Stelle nicht erörtert werden. Die Leistungsfähigkeit der heutigen Endbenutzerwerkzeuge ist in einem Maße gestiegen, daß nicht nur der "normale" Umgang mit einem System, sondern auch die Durchführung von Änderungen (z.B. die Umgestaltung einer Bildschirmmaske) wesentlich vereinfacht wird. Abbildung 3 veranschaulicht diesen Sachverhalt. Während früher auch bei vergleichsweise kleinen Änderungen in einem System Programmierfähigkeiten erforderlich waren, kann der normale Endbenutzer solche Änderungen heute ohne zusätzlich Kenntnisse selbst vornehmen.

Grundlegende Änderungen sind dagegen nach wie vor schwierig. Aufgrund der Komplexität der Softwarewelt sind die Anforderungen sogar noch erheblich gestiegen. Bedurfte es früher im wesentlichen guter Kenntnisse einer Programmiersprache, so muß der Programmierer heute unter Umständen auch den Umgang mit der vielschichtigen Softwareumgebung beherrschen: Grafikwerkzeuge, Fenstersysteme, Datenbanksysteme, Netzwerkprogramme u.a., die mit- (und

manchmal auch gegen-) einander agieren, treten neben die eigentliche Programmiersprache.

Abb. 3: Anforderungen bei Änderung von PC-Software

3 Die "Unix-Welt"

Wenngleich die Verbreitung des Betriebssystems Unix im PC-Bereich noch nicht allzu hoch ist, so hat sich Unix doch bei den Workstations als "das" Betriebssystem etabliert. Andere Betriebssysteme spielen nur eine untergeordnete Rolle. Der einzige Workstation-Hersteller mit größerem Marktanteil, der auch ein eigenes Betriebssystem (VMS) forciert, ist die Digital Equipment Corporation (DEC).

Unix ist ein relativ altes Betriebssystem, das bereits in den sechziger Jahren an den Bell Labotories von AT&T entwickelt wurde, mit dem Ziel, die eigene Softwareentwicklung im Team zu unterstützen. Unix war also nicht als Betriebssystem für Endbenutzer konzipiert und ist dies auch heute nicht.

Vermutlich gibt es kein anderes Betriebssystem, von dem so viele Versionen und Derivate existieren wie von Unix. "Das" Betriebssystem Unix gibt es nicht! In Abbildung 4 sind die beiden Hauptentwicklungslinien zu erkennen. Im linken Teil der Abbildung ist der von AT&T weitergeführte Entwicklungspfad zu erkennen, der über System III, System V.0 hin zu System V.4 führte. Eine zweite Entwicklungslinie wurde ab 1979 an der University of California in Berkeley verfolgt. Diese Linie zeichnete sich durch einen anders aufgebauten Betriebssystemkern und vor allem durch zahlreiche Erweiterungen, die sogenannten *BSD-Erweiterungen*, aus. Heute sind diese Erweiterungen (z.B. der Editor "vi", eine Datenbankschnittstelle u.a.) auch bei anderen Unix-Systemen verfügbar.

Von der BSD-Linie gingen die Unix-Derivate der Marktführer im Workstation-Bereich (Hardware) aus: ***Sun-OS*** (Sun Microsystems) und ***Ultrix*** (Digital Equipment). Das Trio wird durch Hewlett Packard mit ***HP-UX*** vervollständigt, das aber stärker von der AT&T-Linie beeinflußt ist. Ultrix von DEC weist eine gewisse Verwandtschaft mit Sun-OS auf. Ultrix ist sehr umfassend; es gilt als das umfangreichste Unix-System auf dem Markt. Sun-OS wird als besonders innovativ angesehen, während HP-UX als ausgereiftes Produkt gilt, das auch unter Bedingungen der industriellen Praxis robust läuft. Im deutschsprachigen Raum besitzt noch das Siemens-Derivat Sinix eine gewisse Verbreitung.

Abb. 4: Entwicklungsgeschichte von Unix [Quelle: CW-Grafik[3)]]

Das "PC-Unix" war in der Vergangenheit unter dem Namen ***Xenix*** bekannt. Microsoft war mit Xenix der Marktführer unter den Anbietern von Unix-Derivaten für PCs. Vor einiger Zeit trat Microsoft die Xenix-Produktlinie an die Firma Santa Cruz

[3)] Vgl. Vaske (1990), S. 31.

Operations ab, die nun mit der Version *SCO-Unix* den PC-Markt beherrscht - nicht zuletzt deshalb, weil SCO-Unix eine gute Anbindung von DOS-Tasks (!) erlaubt.

Die Firma IBM, deren Unix-Politik jahrelang durch Taktieren gekennzeichnet und etwas undurchschaubar war, hat sich 1990 mit der neuen Workstation-Serie IBM RS/6000 offenbar doch entschlossen, auch den Unix-Markt systematisch anzugehen. Das Derivat *AIX* läßt sich zwar in die AT&T-Linie einordnen, ist aber doch durch eine Reihe wichtiger Erweiterungen und Änderungen gekennzeichnet.

Für den Anwender, der nicht selbst systemnahe Basissoftwareentwicklung betreibt, ist bei der Auswahl eines Unix-Systems vor allem das Angebot an Anwendungssoftware von Bedeutung. Unter Ausklammerung des PC-Bereichs (SCO-Unix u.a.) und unter Hinzunahme von IBM erhält man die folgende Reihenfolge des Softwareangebots:

1. Sun-OS
2. HP-UX (und Domain-OS[4])
3. Ultrix
4. AIX

Auf das zum heutigen Zeitpunkt breiteste Softwarespektrum kann Sun verweisen. Zwischen den Plätzen 2 und 3 klafft eine deutliche Lücke. Den letzten Platz nimmt noch IBM ein, die als letzte in den Markt eingestiegen ist. Veränderungen in der Zukunft sind jedoch wahrscheinlich.

Daß in der heterogenen Unix-Welt, aus der Abbildung 4 nur einen kleinen Ausschnitt zeigt, Vereinheitlichung wünschenswert wäre, liegt auf der Hand. Das Ringen um Marktanteile (und um Lizenzgebühren für AT&T) verhinderte jedoch bis heute einen einheitlichen Unix-Standard. Statt dessen entstanden zwei unterschiedliche Gruppierungen, die sich die Normierung von Unix auf die Fahnen schrieben:

- Unix International Inc. (UII)
- Open Software Foundation (OSF)

Die treibenden Kräfte bei Unix International waren AT&T und der Marktführer Sun, während OSF als Gegenpol gegründet und ursprünglich von HP, DEC, Apollo u.a. getragen wurde. Neben den genannten Firmen sind zahlreiche weitere Unternehmen der Hardware- und Softwarebranche in beiden Vereinigungen vertreten. Bemerkenswert ist die Tatsache, daß heute sehr viele Firmen sowohl Mitglied bei Unix International als auch bei OSF sind.

Der von Unix International angekündigte "Standard" ist die Version *System V.4*, während die Open Software Foundation eine einheitliche Version mit dem Namen *OSF/1* kreierte. In beiden Fällen handelt es sich aber noch weitgehend um Ankündigungen. Daß die zukünftigen Unix-Versionen von DEC, HP, IBM oder Siemens einen genormten Systemumfang aufweisen werden, ist

---

[4]) Domain-OS ist das Unix-Derivat der früheren Firma Apollo, die von Hewlett Packard 1989 übernommen wurde.

indessen kaum zu erwarten. Die Ankündigungen gehen eher dahin, daß die nächste Version des jeweiligen eigenen Betriebssystems "OSF-kompatibel" sein werde.

An der Unix-Standardisierung wirkt neben den oben genannten Vereinigungen auch die hersteller- und produktunabhängige

- X/Open-Gruppe

mit. Diese bezeichnet sich als eine Vereinigung, welche Spezifikationen für "offene Systeme" auf verschiedenen Gebieten der Informationsverarbeitung erarbeitet. Ein wichtiges Ziel ist dabei die *Portabilität* von Programmen. Für den Unix-Bereich, wo X/Open bisher vorrangig in Erscheinung trat, wurden in dem "X/Open Portability Guide" Betriebssystem-schnittstellen definiert, die von den Unix-Implementierungen eingehalten werden sollen. Neben zahlreichen Rechnerherstellern sind auch UII und OSF Mitglieder der X/Open-Gruppe.

4 Großrechner-Betriebssysteme

Die betriebliche Datenverarbeitung im Großrechnerbereich wird weltweit nach wie vor von IBM dominiert. Neben den eigenen Systemen des Marktführers spielen auch PCM-Systeme ("plug compatible manufacturer") eine gewisse Rolle. Bekannte PCM-Hersteller sind etwa die Firmen Amdahl und Hitachi Data Systems (HDS).

Die Betriebssystemszene in der IBM-Welt ist insbesondere durch Heterogenität gekennzeichnet. Das älteste Betriebssystem in Abbildung 5 ist DOS (nicht zu verwechseln mit MS-DOS oder PC-DOS), das immer noch, zusammen mit dem TP-Monitor VSE, als DOS/VSE betrieben wird. Auch MVS geht bis in die siebziger Jahre zurück. MVS ist heute das Betriebssystem der Industrie. Sowohl DOS als auch MVS sind vom Betriebssy-stemkern her Batch-Betriebssysteme. Der Dialogbetrieb wird durch aufgesetzte TP-Monitore abgewickelt. Die bekanntesten TP-Monitore für MVS sind CICS, TSO und - für Anwendungen auf dem Datenbanksystem IMS - IMS/DC.

Abb. 5: Betriebssysteme in der Großrechnerwelt

Eine innovative Entwicklung, die ebenfalls in die siebziger Jahre zurückgeht, war das Betriebssystem VM (Virtual Machine), welches das Konzept der "virtuellen Maschinen" in der Betriebssystemwelt verbreitete. VM war lange Zeit mehr im Forschungs- und Entwicklungsbereich und an Hochschulen als im "harten Produktionsbetrieb" anzutreffen.

Das Prinzip der virtuellen Maschine, das auch der erweiterten Version VM/XA ("extended architecture") zugrunde liegt, bedeutet, daß das Betriebssystem für unterschiedliche Benutzer gewissermaßen eigene Rechner mit eigenen Ressourcen (z.B. Arbeitsspeicher, Peripheriegeräte etc.) simuliert, die grundsätzlich mit unterschiedlichen Betriebssystemen gefahren werden können. So kann unter VM/XA gleichzeitig für einen Benutzer VSE und für einen anderen MVS zusammen mit einem TP-Monitor betrieben werden. Eine gängige Kombination war in der Vergangenheit die, unter VM direkt den TP-Monitor CMS für den Dialogbetrieb zu benutzen.

Mit der 1990 angekündigten neuen Rechnergeneration von IBM wurde die sogenannte ESA-Architektur ("extended system architecture") vorgestellt. Dabei handelt es sich um betriebssystemübergreifende Erweiterungen von TP-Monitoren und (Basis-) Betriebssystemen. Sie sind insbesondere in der Lage, durch ein leistungsfähiges Transaktionsmanagement auch bei großen Adreßräumen externe Speicherstrukturen sehr effizient auf interne Strukturen abzubilden.

Im rechten Teil der Abbildung sind der Vollständigkeit halber einige Betriebssysteme aus der Nicht-IBM-Welt aufgeführt. Größere Verbreitung besitzen vor allem BS 2000 von Siemens und VMS von Digital Equipment. Auch Unix, das im Großrechnerbereich bislang eher ein Exot war, scheint sich mit gemäßigter Geschwindigkeit etwas mehr zu etablieren.

5 Benutzerschnittstellen und Betriebssysteme

5.1 Die Entwicklung der Benutzerschnittstellen

Mit dem Vordringen verschiedenartigster Rechnersysteme an den individuellen Arbeitsplatz hat die Gestaltung der Schnittstelle zum Benutzer hin ein erhebliches Gewicht erlangt. Die Unterstützung der Interaktion zwischen System und Benutzer ist zu einem herausragenden Gegenstand der Forschung und Entwicklung geworden.

In der großrechnerorientierten Datenverarbeitung der sechziger und siebziger Jahre war die Benutzeroberfläche der Systemsoftware nicht für Endanwender, sondern für DV-Fachpersonal gedacht. Die Kommunikation mit dem Rechner erfolgte durch Kommandos und Systemmeldungen in alphanumerischer Form. An die Leistungsfähigkeit der Bildschirmterminals waren somit keine sehr hohen Anforderungen zu stellen. Auch heute sind die typischen Großrechnerterminals noch alphanumerisch und zeilenorientiert.

Grafische Benutzeroberflächen bzw. allgemein grafische Darstellungen könnten zwar grundsätzlich auch auf Großrechnern erzeugt und verwaltet werden; echte Grafikterminals sind allerdings erst in jüngster Zeit für Großrechner attraktiv geworden. Seit die ersten Implementierungen von Basisgrafiksoftware (X-Windows) auch auf Großrechnern verfügbar sind, können sogenannte ***X-Terminals*** angeschlossen werden. Diese waren bisher eher in der Workstation-Welt zu finden. Ihr Preis liegt allerdings noch erheblich über dem von alphanumerischen Terminals. Während letztere je nach Leistungsfähigkeit etwa 500 bis 2.000 DM kosten, erstreckt sich die Preisspanne für X-Terminals noch von etwa 2.500 bis 15.000 DM.

Im mittleren Teil der Abbildung 6 sind grafische Benutzeroberflächen dargestellt, wie sie heute für PCs und Workstations charakteristisch sind. Pull down- oder Pop up-Menüs und ein Pointing Device (Maus) erleichtern den Umgang mit dem System und machen alphanumerische Eingaben über die Tastatur in vielen Fällen überflüssig. Die grafischen Benutzeroberflächen wurden bereits 1978 im Forschungszentrum Xerox PARC in Palo Alto (USA) "erfunden" und als "Xerox Star User Interface" im Forschungsbereich sehr bekannt. Der Chefentwickler Alan Kay nahm seine Ideen beim Wechsel zur Firma Apple mit, wo sie dann Eingang in marktfähige und bezahlbare Produkte fanden.

Die Trennung zwischen PC und Workstation verschwimmt immer mehr. In der Abbildung wurde sie letztlich nach dem zugrundeliegenden Betriebssystem getroffen. Während das Betriebssystem DOS nur den Einbenutzerbetrieb unterstützt, erlaubt das typische Workstation-Betriebssystem Unix sowohl den Mehrbenutzerbetrieb als auch das Multitasking (mehrere Programme können gleichzeitig ablaufen). Dies erfolgt typischerweise in verschiedenen Bildschirmfenstern, wie auch die Abbildung verdeutlicht. Dort sind gleichzeitig drei Programme in Ausführung dargestellt: ein Business-Grafik-Programm, ein Textverarbeitungsprogramm und ein Programm,

welches die Uhrzeit anzeigt. Neben der Fensterorientierung zeichnen sich die Workstation-Benutzerschnittstellen durch sehr gute Grafik aufgrund einer meist hohen Bildschirmauflösung (z.B. 1280 x 1024 Bildpunkte) aus.

Über die Darstellungsformen Text und statische Grafik hinaus sind bereits in allernächster Zeit multimediale Darstellungsformen zu erwarten. Die Einbeziehung von Bewegtbildern (z.B. Filmsequenzen) und Klängen (z.B. verbale Erläuterungen zu einem Dokument) oder Voice-Mailbox-Systeme eröffnen neue Dimensionen der Mensch-Maschine-Kommunikation. Zum Beispiel wird dadurch - unter Nutzung von Ergebnissen der Künstlichen Intelligenz - der verbale, natürlich-sprachliche Umgang mit System und Anwendungssoftware ermöglicht.

Erste Rechner, welche die hardwaremäßigen Voraussetzungen für derartige Benutzerschnittstellen bieten, haben das Forschungsstadium bereits verlassen, so z.B. der NeXT-Rechner von Apple-Gründer Steven Jobs.

Abb. 6: Benutzerschnittstellen

5.2 Benutzerschnittstellen unter PC- und Workstation-Betriebssystemen

Während in der Großrechnerwelt die "Benutzerschnittstelle" keinen großen Aufwand verursachte und vom Betriebssystem selbst mit abgedeckt wurde, sind bei den heutigen Arbeitsplatzrechnern eigenständige und umfangreiche Softwareprodukte speziell für die Erzeugung und Verwaltung der Benutzeroberfläche entstanden. Diese sind auf das eigentliche Betriebssystem aufgesetzt und bestehen im allgemeinen aus zwei oder drei Hauptkomponenten.

Die eine Komponente umfaßt die grafischen Basisfunktionen wie Verwaltung des Bildschirmkoordinatensystems, Verwaltung der Bildschirmfenster und elementare Zeichenfunktionen. Darauf aufgesetzt ist eine Komponente, welche die grafischen Elemente der Bildschirmoberfläche (z.B. Pull down-Menüs) erzeugt und verwaltet.

Im DOS-Bereich ist *MS-Windows* das Standardprodukt für die grafische Oberfläche. Es benutzt als Basissystem das "Graphics Device Interface", übernimmt aber teilweise auch elementare Funktionen (wie die Fensterverwaltung), die im Unix-Bereich etwa von X-Windows bereits bereitgestellt werden. Über das Window-Management (mit mehreren Fenstern) hinaus realisiert MS-Windows - in der Version 3.0 - auch Betriebssystem-Funktionen, wie etwa ein Pseudo-Multitasking und die Adressierung des von DOS nicht erreichten Arbeitsspeicherbereichs jenseits der 640 KB-Grenze.

Im Unix-Bereich ist das bekannteste Basissystem das X-Window-System. Die bekannten Benutzeroberflächen *OSF/Motif* und *Open Look* sind Konkurrenzprodukte der beiden Unix-Vereinigungen Open Software Foundation und Unix International. Da die beiden Unix-"Standards" (System V.4 und OSF/1) noch nicht in den Markt vorgedrungen sind, läßt sich zwar noch keine Voraussage treffen, welches der beiden Systeme das Rennen machen wird. Die jeweiligen Benutzeroberflächen OSF/Motif und Open Look existieren jedoch bereits. Hier läßt sich gegenwärtig beobachten, daß in Europa OSF/Motif weiter verbreitet ist, während in den USA Open Look den Markt anführt. Da die beiden Produkte noch nicht lange verfügbar sind - OSF/Motif kann man seit etwas mehr als einem Jahr kaufen -, läßt sich ein längerfristiger Trend aber noch nicht ausmachen.

OSF/Motif und Open Look legen die grafischen Elemente (z.B. Scroll Bars, Boxes, Pull down-Menüs, Pop up-Menüs) fest, mit deren Hilfe dann grafische Applikationen geschaffen werden können. Zur Nachbildung einer komfortablen integrierten Arbeitsoberfläche muß auch das Betriebssystem selbst "abgekapselt" werden. Auf OSF/Motif oder Open Look setzen deshalb weitere Softwareprodukte auf, die man als "Desktop-Manager" bezeichnet. Sie stellen die

Betriebssystemfunktionen, dem "Style Guide" von Motif bzw. Open Look entsprechend, in grafischer Form (Ikonen etc.) dar. Die gegenwärtig bekanntesten Desktop-Manager sind *Looking Glass* und *X.Desktop*. Wie die Abbildung andeutet, sind sie nicht an eine bestimmte Benutzeroberfläche gekoppelt. Das "Look and Feel" von X.Desktop oder Looking Glass ist demnach unterschiedlich, je nachdem, ob Motif oder Open Look zugrundeliegt. Weitere Desktop-Manager befinden sich in Entwicklung, u.a. auch von OSF und UII selbst.

Die bereits erwähnte *NeXT*-Entwicklung von Steven Jobs ist in der Abbildung ebenfalls aufgeführt. Die spezielle Hardware- und die objektorientierte Softwarearchitektur des NeXT-Rechners sind an der Multimediafunktion des Rechners ausgerichtet. NeXTStep ist das Grafikwerkzeug, auf dem der Desktop-Manager NeXT aufsetzt.

Bei OS/2 ist eine ähnliche Systemstruktur wie bei DOS zu erkennen. Aussehen und Funktionsweise des *Presentation Manager* weisen große Ähnlichkeit mit MS-Windows und auch mit OSF/Motif auf. Dies ist nicht verwunderlich, da Microsoft sowohl federführend die Entwicklung von OS/2 als auch von MS-Windows betreibt und in der OSF-Gruppe mitarbeitet. Im Gegensatz zu MS-Windows (DOS-Basis) ist unter dem Presentation Manager (auf OS/2-Basis) echtes Multitasking möglich.

Abb. 7: Architektur der Benutzerschnittstellen

Die unterschiedliche Höhe der Kästchen in Abbildung 7 deutet den unterschiedlichen Funktionsumfang der jeweiligen Softwarekomponenten an.

6 Inhouse-Netze

6.1 PC-Netze

Die Verbreitung der PCs hatte in den Unternehmen teilweise einen erheblichen Wildwuchs an unkoordinierter Datenverarbeitung zur Folge, aus der der Bedarf nach Integration erwuchs. An den Hochschulen, an denen das dezentrale Chaos in Einklang mit der Freiheit von Forschung und Lehre eher eine gewisse Daseinsberechtigung besitzt, förderte ein formales Argument die Vernetzung: im Rahmen des Computerinvestitionsprogramms (CIP), das über das Hochschulbauförderungsgesetz (HBFG) finanziert wurde, mußten aus den vielen tausend bundesweit beschafften PCs künstlich größere Einheiten gemacht werden, die die HBFG-Schwelle von 150 TDM überschritten. Dies wurde durch vernetzte CIP-Pools erreicht.
Da im PC-Bereich DOS das Standardbetriebssystem darstellt, ist die Leistungsfähigkeit der PC-Netze nicht allzu hoch. Eine typische Konfiguration, wie sie im unteren Teil der Abbildung 8 dargestellt ist, sieht so aus, daß ein Rechner im Netz die Rolle des Servers übernimmt. Die anderen Netzknoten sind eigenständige Rechner, die insbesondere auch Programme auf dem eigenen Prozessor ausführen und unter dem lokalen Betriebssystem DOS laufen. Die wichtigsten Server-Funktionen sind die des

- *File-Servers*, auf dessen Dateien andere Rechner im Netz zugreifen können (einschließlich des Ladens ausführbarer Programme von dort),
- *Print-Servers*, der Druckaufträge von anderen Rechnerknoten bearbeitet,
- *Datenbank-Servers*, bei der ein Datenbankkern auf dem Server residiert und der Zugriff mehrerer Benutzer möglich ist.

Abb. 8: Rechnernetze

Meist sind in PC-Netzen nur die beiden ersten Server-Funktionen realisiert, da die letztere Multitasking-Fähigkeiten erfordert.

Die Netzwerkverwaltung wird von spezieller Netzwerksoftware ausgeführt; diese umfaßt Funktionen wie die normaler Betriebssysteme und darüber hinaus besondere Netzdienste (z.B. File-Transfer und Zugriff auf Programme, die auf dem Server residieren). Bekannte Netzwerksysteme sind Advanced Netware (Novell), Netware 386 (Novell), Banyan Vines, PC-LAN (IBM) und MS-Net (Microsoft). Marktführer ist mit Abstand Novell.

Unter OS/2 wird die Trennung des Betriebssystems und der Netzwerksoftware, die bei DOS eigenständig und aufgesetzt ist, aufgehoben. Der LAN-Manager von OS/2 ist integrierter Bestandteil des Betriebssystems.

Von der Netzwerksoftware zu trennen ist die technische Basis für die Informationsübermittlung. Dies sind einerseits spezielle Netzwerkkarten mit Netzwerk-Controllern und zum anderen das Übertragungskabel. Im PC-Bereich besitzen *Arcnet*-Karten eine gewisse Verbreitung. Die leistungsfähigere (und teurere) *Ethernet*-Hardware, die vor allem in Workstation-Netzen anzutreffen ist, kann jedoch auch für PC-Netze benutzt werden.

6.2 Workstation-Netze

Da in der Workstation-Welt leistungsfähigere Betriebssysteme verwendet werden (Unix, VMS im DEC-Bereich), sind

grundsätzlich auch Leistungsfähigkeit und Funktionalität der Netze erheblich größer. Da auf jeder Workstation Multitasking möglich ist, kann grundsätzlich jeder Rechnerknoten auch Serverfunktionen übernehmen (z.B. Datenbankserver, Druckserver, Remote Procedure Calls etc.). Wenngleich man aus organisatorischen Gründen ein Netz nicht so konfigurieren würde, daß jeder Rechner auf jeden anderen zugreifen kann, sind doch häufig mehrere Knoten mit Serverfunktionen sinnvoll ("Multiserver-Konfiguration"[5]).

In Abbildung 8 ist an das Workstationnetz auch noch ein Cluster angehängt, das einem PC-Netz ähnelt. Die Rechnerknoten des Clusters, die bewußt als "diskless" gezeichnet sind, können über effiziente Zugriffsmechanismen die auf dem Serverknoten residierenden Daten erreichen. Eine solche Konfiguration ist beispielsweise im CAD-Bereich ("computer aided design") anzutreffen. Dort wird für die Grafikverarbeitung an jedem Arbeitsplatz hohe Prozessorleistung benötigt, während andererseits der Umfang der Datenzugriffe relativ niedrig ist. Die einzelnen Arbeitsplätze brauchen deshalb keine eigenen Plattenspeicher. Sie können über NFS ("native file system") sehr effizient auf das Dateisystem des Serverknotens zugreifen. Der Zugriff ist "transparent"; dies bedeutet, daß der Benutzer an einem anderen Netzknoten den Eindruck hat, er arbeite auf einer lokalen Platte.

Die hardwaretechnische Netzwerkbasis ist bei den Workstations im allgemeinen Ethernet. In der ISO/OSI-Protokollhierarchie deckt Ethernet etwa die unteren drei Ebenen ab. Für die Ebene 4 bis 7 sind ISO/OSI-konforme Protokolle noch nicht verfügbar oder nicht verbreitet. Statt dessen hat sich das aus einer älteren "Netzwerk-Welt", der der Arpanet-Protokolle, stammende TCP/IP als der gegenwärtige Standard für Workstationnetze auf Ethernetbasis durchgesetzt. Netzdienste auf einer höheren Ebene als TCP/IP werden durch das Betriebssystem (z.B. Unix) selbst abgedeckt (z.B. File-Transfer).

Am Rande sei angemerkt, daß das Protokoll TCP/IP keineswegs auf Workstationnetze beschränkt ist. Am Institut für Wirtschaftsinformatik wird beispielsweise ein PC-Netz mit Novell-Software betrieben, die auf dem TCP/IP-Protokoll aufsetzt und Ethernet als Vernetzungsbasis benutzt.

Für den Übergang zwischen verschiedenen PC- und/oder Workstation-Netzen werden häufig Gateways eingesetzt. Dies sind Protokollumsetzer, die beispielsweise die Informationen eines Workstation-Netzes von TCP/IP nach SPX/IPX, dem von der Novell-Software am häufigsten benutzten Protokoll, umsetzen. Gateways können auf verschiedene Weise realisiert sein, als eigene Rechner, als spezielle Einschubkarten, als kleine Geräte oder auch als Software.

---

[5]) Vgl. Kurbel, Moazzami (1990), S. 20 ff.

6.3 Großrechner-Netze

Vernetzung in der Großrechner- und Midrangewelt muß unter verschiedenen Gesichtspunkten betrachtet werden.

Eine rudimentäre Form der Vernetzung ist die Anbindung von Terminals an den Großrechner. In der IBM-Welt sind die sogenannten 3270-Terminals zwar schon etwas antiquiert, aber sehr weit verbreitet. Das entsprechende Protokoll für die Informationsübermittlung ist 3270-Term. Neuere Protokolle, wie etwa VT 220 von DEC, erlauben eine zeichenorientierte Übertragung mit Berücksichtigung von Farbe, inverser Darstellung und anderen Eigenschaften.

Die im oberen Teil der Abbildung 8 skizzierte Verbindung zweier Großrechner basiert auf der Hyperchannel-Architektur. Dabei handelt es sich um eine sogenannte "Peer-to-Peer"-Verbindung auf Glasfaserbasis, die sehr effizient ist. Da genau zwei Rechner ohne weitere Stationen dazwischen miteinander verbunden sind, vereinfacht sich die Übertragung ganz erheblich. Viele Sicherungsprotokolle können beispielsweise wegfallen. Die Übertragungsraten liegen im Gigabitbereich. Der Hauptzweck einer Hyperchannelverbindung ist die Lastverteilung zwischen zwei Rechnern. Für den Benutzer erfolgt dies "transparent", d.h., er bemerkt nicht, welcher Prozessor gerade sein Problem bearbeitet.

SNA ("systems network architecture") ist das Konzept, mit dem in der heterogenen IBM-Welt große und mittlere Rechner miteinander vernetzt werden können. Die Anbindung von PCs an einen Großrechner fällt jedoch nicht darunter. Die einzige Möglichkeit, von einem PC auf einen IBM-Großrechner zuzugreifen, besteht häufig darin, eine 3270-Emulation zu benutzen. Dies bedeutet jedoch, daß sich der PC dann weitgehend wie ein "dummes" Bildschirmterminal verhält und seine sonstigen vorteilhaften Eigenschaften nicht verfügbar sind.

6.4 Backbone-Netze

In der Vergangenheit wurden verschiedenartige Netzwerke und/oder Rechner, sofern der Bedarf bestand, meist durch punktuelle Verbindungen miteinander integriert, etwa auf dem Wege der oben erwähnten Gateways, oder in der IBM-Großrechnerwelt über SNA. Dies war durchaus praktikabel, solange netzwerkübergreifende Leistungen eher die Ausnahme darstellten.

Mit der wachsenden Vernetzung auf den verschiedensten Ebenen gewann die Netzwerkinfrastruktur für Unternehmen und andere Institutionen zunehmende Bedeutung. Heute werden in Unternehmen und Verwaltungen, aber auch an den Universitäten sogenannte Backbone-Netze installiert. Diese sind Hochgeschwindigkeitsnetze, die primär die Aufgabe haben, die Verbindung verschiedener Netze herzustellen, und gleichzeitig auf mehreren Kanälen Datenverkehr betreiben können.

Da Backbone-Netze auf Glasfasertechnik basieren, können sehr hohe Übertragungsgeschwindigkeiten erreicht werden. Diese liegen in der Größenordnung von 100 Megabit pro Sekunde. Das Übertragungsprotokoll für solche Hochgeschwindigkeitsnetze ist FDDI ("fiber distributed data interface").

Exkurs: Im Zusammenhang mit dem Zusammenwachsen heterogener Netze muß auch der Begriff SAA ("systems application architecture") erwähnt werden. SAA ist das Integrationskonzept von IBM zur Verbindung heterogener Rechnerwelten und Netze in einem einheitlichen System und unter einer einheitlichen Benutzeroberfläche. Unter SAA sollen Netze und Rechner, wie die in Abbildung 8 dargestellten, betrieben werden können, solange es sich ausschließlich um IBM-Systeme handelt. Daß IBM hier eine gigantische Integrationsaufgabe bewältigen will, liegt auf der Hand. Es wird sicher noch viele Jahre dauern, bis SAA durchgängig realisiert ist.

7 Softwareentwicklung

7.1 Sprachen und Werkzeuge

In den sechziger und siebziger Jahren war es üblich, die Programmiersprachen - analog zu der Einteilung der Rechnerhardware - bestimmten Generationen zuzuordnen. Wenngleich nur bis zur dritten Generation eindeutige Abgrenzungskriterien vorhanden sind, ist das Denken in Generationen, nicht zuletzt unter Marketing-Gesichtspunkten, auch heute noch verbreitet. Daß dies nicht sonderlich sinnvoll ist, wird die Erläuterung der Abbildung 9 zeigen.

Im linken Teil der Abbildung sind die sogenannten Generationen aufgeführt. Die erste Sprachgeneration umfaßt die reinen Maschinensprachen (Binärcode, evtl. in externer Oktal- oder Hexadezimaldarstellung). In die zweite Generation fallen die maschinenorientierten Sprachen (Assemblersprachen). Dem Programmierer stehen hier verständlichere, z.T. frei wählbare Wortsymbole für die Benennung von Befehlsarten und -operanden sowie eine erste Form von Unterprogrammen (Makros) zur Verfügung.

Mit den Sprachen der dritten Generation wurde die Orientierung an dem anlagenspezifischen internen Befehlsformat über Bord geworfen. Die Ausdrucksweise der sogenannten "höheren" oder "problemorientieren" Programmiersprachen ist teilweise der englischen Umgangssprache, teilweise der Mathematik entlehnt. Wenngleich damit noch nicht die Unabhängigkeit von Maschinen garantiert ist, lassen sich die Programme doch mit geringerem Aufwand von einer Hardware auf eine andere portieren. Die ersten Sprachen der dritten Generation entstanden Ende der fünfziger Jahre (Fortran um 1956/57, Cobol 1959/60).

Die Explosion der Softwarekosten in den sechziger Jahren mit der Erkenntnis, daß die Programmierung in einer Krise steckte, führte zum Entstehen der "Strukturierten Programmierung". Maßstäbe für zweckmäßige Programmstrukturierung setzte Wirth mit der Entwicklung von Pascal (1968/70). Dies zeigt sich nicht zuletzt darin, daß Pascal auch jeweils die Basis

für wichtige neue Programmiersprachen (Ada, Modula) darstellte.

Jenseits der dritten Generation verschwimmen die Grenzen. Die unterschiedlichsten Produkte werden als Sprachen der vierten oder fünften Generation bezeichnet. Darüber hinaus fehlt ein wichtiges Kriterium, das für die ersten drei Generationen galt: Die Sprachen sind bzw. waren Universalsprachen, und von einer Generation zur nächsten wuchs die Mächtigkeit der Sprachelemente. Mit einer Sprache der dritten Generation lassen sich grundsätzlich die gleichen Anwendungsprobleme wie mit einer Assemblersprache lösen, aber mit wesentlich höherer Produktivität. Beim Weiterzählen der Generationen gilt dies nur noch teilweise (bei der vierten Generation) oder gar nicht mehr (bei der fünften Generation).

Bei den Sprachen der vierten Generation, die man gern als 4GL ("fourth generation languages") bezeichnet, lassen sich grundsätzlich zwei Linien unterscheiden[6]:

a) Zum einen gibt es 4GL, die eher auf den Paradigmen der dritten Generation aufbauen, die Entwicklung von Anwendungsprogrammen jedoch mit wesentlich leistungsfähigeren Sprachelementen und höherer Produktivität unterstützen. Solche Sprachen sind etwa Natural und Ideal. Im weiteren Sinne kann man auch direkte Erweiterungen von Sprachen der dritten Generation zu den 4GL rechnen, z.B. Pro*C (C, erweitert um SQL-Elemente) oder die Codasyl-Erweiterungen von Cobol (Cobol mit Datenbankelementen) dar.

b) Zum anderen gibt es 4GL, die auf Basis von Datenbankmanagementsystemen oder Abfragesprachen entwickelt wurden. Ihre Stärken liegen bei Datenauswertungen und -manipulationen auf der Grundlage einer Datenbank. Zu dieser Kategorie gehören etwa Focus, SQL*Forms und ebenfalls Natural.

[6]) Vgl. Kurbel, Eicker (1988), S. 21 ff.

Abb. 9: Generationen und Entwicklungsrichtungen bei Sprachen und Werkzeugen

Der Begriff der fünften Generation ist noch diffuser. Als gemeinsames Merkmal hat sich mittlerweile das Anwendungsfeld "Künstliche Intelligenz" herauskristallisiert. Einerseits werden die typischen KI-Programmiersprachen - die wichtigsten Vertreter sind Lisp und Prolog - der fünften Generation zugerechnet, zum anderen Entwicklungswerkzeuge für Expertensysteme (Expertensystem-Shells) wie Emycin, ESE, Xi Plus und Nexpert Object.

Bei der fünften Generation zeigt sich deutlich, daß der universelle Charakter der ersten drei Generationen nicht mehr gegeben ist. Die Mächtigkeit der Sprachelemente ist sehr hoch; diese sind aber sinnvoll nur für ganz spezielle Problemkategorien einsetzbar.

Wie unzweckmäßig es ist, in Generationen weiterzuzählen, kommt im rechten Teil der Abbildung zum Ausdruck. Aus der Weiterentwicklung und Differenzierung der im linken Teil dargestellten Merkmale und Sprachen entstanden Softwareprodukte, denen andere Leitgedanken zugrundeliegen. So wäre *Ada* etwa, betrachtet man nur die nackte Programmiersprache, der dritten Generation zuzurechnen. Ada enthält aber Sprachkonstrukte und Werkzeuge in der Sprachumgebung, die auf den gesamten Prozeß der Softwareentwicklung gerichtet sind und das Software Engineering über alle Phasen der Software Life Cycle unterstützen sollen.

Smalltalk verwirklicht das Paradigma der objektorientierten Programmierung, die im linken Teil noch gar nicht auftauchte. Es ist allerdings einzuräumen, daß es durchaus Marketingstrategen gibt, die objektorientierte Werkzeuge den 4GL zurechnen. Smalltalk paßt auch aus anderen Gründen nicht zu den "Sprachen" im engeren Sinne, da es nicht nur als Programmiersprache, sondern gleichzeitig auch als Softwareentwicklungsumgebung und Betriebssystem konzipiert wurde.

ZIM ist ein Beispiel für ein neueres Werkzeug der vierten Generation, welches auf einem aktiven Data Dictionary aufsetzt und im Sinne eines modernen CASE-Tools Unterstützung

bei der Datenmodellierung bietet (vgl. auch "datenbankzentrierte Entwicklungsumgebungen" in Abschnitt 7.2).

ZENO vereinigt in sich Merkmale eines datenbankorientierten Entwicklungswerkzeugs mit denen einer Expertensystem-Shell, die sowohl deduktive als auch induktive Ansätze zur Problemlösung unterstützt. Die Integration mit anderen Softwaresystemen ist besonders hervorhebenswert: ZENO besitzt einerseits Schnittstellen zu zahlreichen Werkzeugen; andererseits können die erstellten Wissensbasen in C- oder Pascal-Code compiliert und nahtlos in operative Anwendungssysteme eingebunden werden.

Ein erster Meilenstein auf dem Weg zur domänenorientierten Wissensmodellierung ist das bislang als Prototyp vorliegende Werkzeug *SALT*. Damit wird ein gravierender Nachteil der "Universal-Shells" wie Xi Plus oder Nexpert Object beseitigt: Wissen in unterschiedlichen Problemdomänen (z.B. Konfigurierung, Diagnose, Planen) kann sehr unterschiedlich strukturiert sein, in unterschiedlicher Form vorliegen oder in unterschiedlichen Wissensquellen enthalten sein. SALT ist in diesem Sinne ein Wissensakquisitionswerkzeug für Konfigurierungsprobleme. Es erzeugt u.a. ausführbare Regeln in OPS5.

Auf die Ansätze der fallbasierten Wissensverarbeitung und des Konnektionismus, die sich der Zuordnung zu Generationen völlig verschließen, wird in Kapitel 8 eingegangen.

Wenn man die Ebene der nackten Programmiersprache verläßt und die Unterstützung weiterer Phasen des Software Life Cycle mit einbezieht, gewinnt die Integration der zahlreichen Einzelkomponenten und -ergebnisse im Hinblick auf ihre Wieder- oder Weiterverwendbarkeit erhebliches Gewicht. Bei einem "konventionellen" Softwareentwicklungsprojekt auf Datenbankbasis sind als Integrationsinstrumente etwa eine Projektbibliothek und ein Data Dictionary unverzichtbar. Eine detaillierte Erläuterung der im rechten Teil der Abbildung angesprochenen Integrationsinstrumente kann im Rahmen dieses Überblicks nicht gegeben werden. Die Unterstützung des Softwareentwicklungsprozesses insgesamt wird jedoch in Kapitel 7.3 nochmals aufgegriffen.

7.2 Datenbankmodelle und Datenbanksysteme

Die Entwicklung der Datenmodelle und der darauf aufbauenden Datenbanksysteme ist in Abbildung 10 dargestellt. Eine ähnliche Übersicht über die Entwicklung findet man bei Parsaye u.a.[7]. Die klassischen Datenmodelle sind das hierarchische Modell, das Netzwerkmodell und das relationale Modell. Zwar gilt das relationale Modell heute weitgehend als State of the Art. Datenbanksysteme auf Basis der beiden anderen Modelle sind in der Praxis jedoch noch sehr weit verbreitet und werden sicherlich noch lange Zeit existieren.

[7]) Vgl. Parsaye u.a. (1989), S. 144.

Dies gilt insbesondere für IMS (IBM), aber auch für UDS (Siemens) und IDMS (Cullinet).

Wie die Abbildung zeigt, ist die Entwicklung mit dem Relationenmodell noch lange nicht am Ende angelangt. Zur Beseitigung von Nachteilen des Relationenmodells und teilweise auch aus anderen Entwicklungsrichtungen heraus sind alternative Ansätze und Erweiterungen entstanden:

NF²-Datenbanksysteme

NF²-Datenbanksysteme ("non first normal form") machen einen "Fortschritt" der Datenbankentwicklung rückgängig, der sich in manchen Fällen doch als Nachteil herausstellte, nämlich das Verbot von Wiederholungsgruppen und die stark atomisierte Struktur der relationalen Darstellung. Diese räumt zwar sehr viel Flexibilität ein, verursacht jedoch auch sehr hohen Aufwand bei Auswertungen. Das einzige bekanntere NF²-Datenbanksystem auf dem Markt ist bislang PISA.

"Entity-Relationship-Datenbanksysteme"

Aufgrund der geringen Semantik der relationalen Darstellung beschäftigt man sich im Datenbankbereich schon lange mit sogenannten semantischen Datenmodellen. Eine sehr bekannte, einfache Ausprägung davon ist das Entity-Relationship-Modell (ER-Modell) von Chen[8]).

[8]) Vgl. Chen (1976).

Abb. 10: Datenbankmodelle und Datenbanksysteme

Den Begriff "Entity-Relationship-Datenbanksystem" besetzte neuerdings die Software AG mit ihrem weitverbreiteten System *Adabas*. Dieses ist ein älteres und eigentlich eher atypisches Datenbanksystem. Es hat jedoch die angenehme Eigenschaft, n:m-Beziehungen zuzulassen. Diese kann man sonst zwar im ER-Modell darstellen; man muß sie aber, wenn man ein relationales Datenbanksystem benutzt, anschließend künstlich auflösen. Adabas ist dagegen in der Lage, die ER-Darstellung unmittelbar strukturell abzubilden. Darüber hinaus gibt es grafische Werkzeuge, welche die Erzeugung der ER-Diagramme unterstützen und sie automatisch in die Datendefinitionen von Adabas umsetzen.

Non-Standard-Datenbanksysteme

Vor allem im Software Engineering, wo man sich mit Softwareentwicklungsumgebungen beschäftigte, entstand der Bedarf, Entwurfs-, Implementierungs- und andere Zwischenergebnisse des Softwareentwicklungsprozesses abzulegen. Hier wurden eigene Datenmodelle entwickelt, die spezifisch auf die Anwendung "Softwareentwicklung" zugeschnitten sind und keinen Bezug zu den klassischen Datenmodellen aufweisen. Ein typisches Objekt in einer Non-Standard-Datenbank ist etwa ein SADT-Diagramm, ein Programmodul oder eine Modulbibliothek.

Objektorientierte Datenbanksysteme

Die drei genannten Ansätze gingen in die Entwicklung der objektorientierten Datenbanksysteme ein. Die Erweiterungen gegenüber herkömmlichen Datenbanksystemen sind allerdings weniger spektakulär, als der Begriff vielleicht erwarten läßt. Als objektorientierte Datenbanksysteme bezeichnet man heute Systeme, die

a) die Definition eigener Datentypen, insbesondere strukturierter Typen,

b) die Schachtelung von Datenobjekten in einer Hierarchie, ähnlich wie NF^2-Datenbanksysteme, erlauben oder

c) beide Eigenschaften a) und b) besitzen; die letzteren Systeme bezeichnet man auch als "Objektbanksysteme".

Der Einsatz objektorientierter Datenbanksysteme ist dort sinnvoll, wo sehr komplex strukturierte Objekte verwaltet werden müssen, z.B. in der Künstlichen Intelligenz (Frames, semantische Netze, Regeln etc.) oder im CAD-Bereich, wo komplexe Produktmodelle gespeichert werden müssen.

Verschiedene neuere Konzepte im Datenbankbereich sind modellübergreifend bzw. nicht im Zusammenhang mit einem bestimmten Datenmodell entstanden:

- *Verteilte Datenbanken* erlangen dadurch zunehmendes Gewicht, daß mit der Vernetzung unterschiedliche Unternehmensbereiche informationstechnisch zusammenwachsen und die Zahl der Datenbankzugriffe über das Netz ansteigt. Dies kann die Netzlast erheblich erhöhen. Ein wichtiges Prinzip ist deshalb, Daten so zu verteilen, daß sie an dem Netzknoten angesiedelt sind, an dem die meisten Zugriffe zu erwarten sind ("die Last folgt den Daten").

- Der *SQL-Standard* ("structured query language") wird zwar meist mit dem Relationenmodell in Verbindung gebracht, er ist jedoch auch für andere Datenmodelle verfügbar; z.B. besitzen netzwerk- oder objektorientierte Datenbanksysteme und ein System wie Adabas ebenfalls eine SQL-Schnittstelle.

- *Multimedia-Datenbanken* gewinnen in dem Maße an Bedeutung, wie die multimedialen Darstellungen in Benutzeroberflächen Eingang finden. Sie zeichnen sich u.a. durch den Datentyp BLOB ("binary large object") aus; dieser ist darauf ausgerichtet, andere Objekte als Text und Zahlen (z.B. Bilder, Klänge, Programmcode etc.) aufzunehmen.

- *Datenbankgestützte Entwicklungsumgebungen* sind in der CASE- und 4GL-Szene auszumachen. Anders als bei den allgemeinen CASE-Werkzeugen, auf die im nächsten Kapitel eingegangen wird und die den Softwareentwicklungsprozeß im Software Life Cycle unterstützen, handelt es sich hier um Werkzeuge, die auf den komfortablen Umgang mit einem Datenbanksystem und dem zugrundeliegenden Datenmodell ausgerichtet sind.

7.3 Softwareentwicklungswerkzeuge und CASE

Die Softwarekrise der sechziger Jahre, die sich in einem dramatischen Anstieg der Kosten niederschlug, führte zur Entstehung der Disziplin des Software Engineering. Zur Reduktion der Problemkomplexität im Softwareentwicklungsprozeß wurden u.a. Vorgehensmodelle entwickelt, sogenannte Software Life Cycle-Modelle, die den Entwicklungsprozeß in Phasen zerlegen. In Abbildung 11 ist ein solches vereinfachtes Phasenmodell dargestellt.

Abb. 11: Die Entwicklung der Computerunterstützung im Softwarelebenszyklus

Während man lange Zeit lineare Phasenmodelle verwendete und auch heute noch verwendet, hat sich in Forschung und Entwicklungspraxis mittlerweile die Erkenntnis durchgesetzt, daß Rückkopplungen und Revisionen der Zwischen- und Endergebnisse in vielen Fällen unabdingbar sind ("evolutionäre Entwicklung", "Prototyping"). In der Abbildung wird dies durch die rückwärts gerichteten Pfeile zum Ausdruck gebracht. Die Wartungsphase, die sich über die gesamte Lebensdauer eines Softwareprodukts erstreckt, hat ihr eigenes Zyklusmodell. Dieses ist in der Abbildung aus Platzgründen nicht aufgeführt.

Die vertikale Struktur der Abbildung drückt im Groben den Zeitablauf von der Vergangenheit in Richtung Zukunft aus. Zunächst beschäftigte man sich im Software Engineering mit Vorgehensweisen, Prinzipien und Methoden für die einzelnen Phasen:

- Für die Phase "Anforderungsdefinition" wurden allgemeine Vorgehensweisen und spezielle Methoden erarbeitet, welche die Problemanalyse und die Dokumentation der Anforderungen an das zu entwickelnde Softwaresystem (z.B. als Pflichtenheft) beinhalten (aufgrund der herausragenden Bedeutung dieser frühen Phase für den gesamten Softwareentwicklungsprozeß ist mittlerweile mit dem "Requirements Engineering" eine eigene Teildisziplin entstanden).

- In die Entwurfsphase dürfte der größte Teil der Forschungsaktivitäten geflossen sein. Hier wurden zahlreiche Prinzipien (Top-down-Entwicklung, Abstraktionsprinzipien, abstrakte Datentypen etc.) und Methoden, z.B. SADT (Structured Analysis and Design Technique), JSD (Jackson Structured Design), SD/CD (Structured Design/Composite Design) etc. entwickelt.

- Die Implementierungs- und Testphase wurde sehr stark an Softwarequalitätsmerkmalen (z.B. Zuverlässigkeit, Verständlichkeit, Benutzerfreundlichkeit, Robustheit,

Wartungsfreundlichkeit u.a.) orientiert. Methodische Unterstützung für die Implementierung (z.B. strukturierte Programmierung) und das Testen (z.B. Blackbox-, Whitebox-, Pfadtesten etc.) wurde hier in den Vordergrund gestellt.

Der Großteil der Softwareentwicklungsaktivitäten, vor allem in den frühen Phasen, mußte von Hand durchgeführt werden. Computerunterstützung gab es nur für die Implementierungsphase und teilweise für die Testphase: ein Editor für das Erstellen der Programme und ein Compiler für die Übersetzung stellten den Kern der Werkzeugunterstützung dar. Wenn ein Debugger für die Fehlersuche verfügbar war, handelte es sich schon fast um eine Luxusausstattung. Die genannten Werkzeuge waren meist Einzelwerkzeuge, die isoliert nebeneinander standen.

Zur Steigerung der Produktivität wurde im Software Engineering dann zunehmend versucht, den Softwareentwicklungsprozeß zu automatisieren; dies kommt auch in dem Schlagwort CASE ("computer aided software/systems engineering") zum Ausdruck. Ein Teil der Aktivitäten war darauf gerichtet, die Softwareentwurfsmethoden durch grafische Werkzeuge zu unterstützen. So gibt es beispielsweise Werkzeuge, mit denen der Entwurf eines Softwaresystems in Form von SADT-Diagrammen interaktiv durchgeführt werden kann.

Das Spektrum der Entwurfswerkzeuge ist sehr breit. Es reicht vom einfachen Grafik-Editor bis hin zu Systemen, die auch Semantik verwalten und z.B. Konsistenzprüfungen durchführen. Letztlich handelt es sich um komfortable Dokumentationshilfen. Die reinen Entwurfswerkzeuge waren bzw. sind vom Grundsatz her Stand alone-Werkzeuge; d.h. die Entwurfsergebnisse können nicht automatisch von einem Implementierungswerkzeug weiterverarbeitet werden. Manchmal existieren jedoch Konvertierungswerkzeuge, die die Übergabe an ein ganz bestimmtes Implementierungswerkzeug erlauben. Bekannte Werkzeuge sind etwa AKL/KLAR von Siemens (für SADT) und Speedbuilder (für JSD).

Die sogenannten *Programmierumgebungen* zur Unterstützung der Implementierungs- und Testphase waren zeitlich gesehen die ersten integrierten Softwarewerkzeuge. Bekannt wurde etwa UCSD-Pascal. Bei den Programmierumgebungen sind unterschiedliche Ansätze zu beobachten:

- Bei den sprachzentrierten Umgebungen ranken sich die Werkzeuge um eine bestimmte Programmiersprache herum. Beispiele sind etwa die Turbo-Umgebung für Pascal und die APSE (Ada programming support environment) für Ada. Vor allem die 4GL stehen selten als nackte Sprache dar, sondern sind in eine Umgebung eingebettet (z.B. Natural).

- Betriebssystemzentrierte Umgebungen enthalten großenteils Werkzeuge, die sprachübergreifend, wenn auch oft nur mäßig integriert sind, wie Editoren, Compiler, Linker, Debugger, Maskengeneratoren, Hilfen für die Quellcode- und

Konfigurationsverwaltung, Performanceanalysen u.a. Beispiele sind OPSS und VAX-Set.

- Bei den datenbankzentrierten Entwicklungsumgebungen (vgl. auch Kapitel 7.2) ranken sich die Werkzeuge um das eigentliche Datenbanksystem. Beispiele hierfür sind Adabas und Datacom/DB.

- Zu den Programmierumgebungen gehören im weiteren Sinne auch Endbenutzerwerkzeuge wie Lotus 1-2-3, Excel oder dBase. Der Begriff Programmierumgebung wird hierfür aber seltener verwendet, da er eher auf die Unterstützung beim Umgang mit prozeduralen Sprachen der dritten oder vierten Generation gerichtet ist.

Der Schritt zur *Softwareentwicklungsumgebung* (oder Softwareproduktionsumgebung) zielt zum einen auf die Integration der Phasen im Software Life Cycle; zum anderen wird versucht, auch die früheren und späteren Phasen (Anforderungsdefinition, Wartung) mit einzubeziehen. Die grundlegende Idee ist die phasenübergreifende Nutzung von Zwischenergebnissen durch Verwendung eines aufeinander abgestimmten Werkzeugbündels. Dies erfordert entweder ein zentrales Meta-Informationssystem, in dem die Zwischenergebnisse in einer allgemeinen Form abgelegt werden, so daß sie von einem Werkzeug der nächsten Phase weiterverarbeitet werden können; oder es müssen Funktionen existieren, welche die Konvertierung von der Darstellungsform des einen Werkzeugs in die eines anderen leisten.

Wenn etwa der Entwurf durch SADT-Diagramme dokumentiert ist, kann in der Implementierungsphase zum Beispiel automatisch der Programm- bzw. Modulrahmen von einem entsprechenden Werkzeug erzeugt und durch Pseudocode teilweise gefüllt werden. Aufgabe des Programmierers ist es dann, den Pseudocode weiter auszufüllen bzw. zu verfeinern, so daß das Implementierungswerkzeug automatisch den Programmcode generieren kann.

Auf der Datenseite kann die Integration etwa so aussehen, daß die Datenmodellierung in einem ER-Diagramm dokumentiert ist, das durch Hinzufügen von Konsistenzbedingungen und Tuning-Maßnahmen verfeinert und erweitert wird. Das Implementierungswerkzeug überführt dann automatisch die erweiterte Darstellung in den Code der Datendefinitionssprache (DDL) eines bestimmten Datenbanksystems (z.B. in SQL-Anweisungen zum Anlegen der Tabellen).

Bei den bisherigen Softwareentwicklungsumgebungen (Promod, Prados, Maestro u.a.) wurden die frühen und späten Phasen kaum abgedeckt. Außerdem ist bei derartigen Systemen das Werkzeugbündel fest definiert. Für jede Phase wird grundsätzlich ein ganz bestimmtes Werkzeug verwendet. Es besteht keine Möglichkeit, Werkzeuge auszuwählen oder andere einzubinden - abgesehen von den Übersetzern, die meist für mehrere Programmiersprachen zur Verfügung stehen. In diesem Sinne handelt es sich um "geschlossene Softwareentwicklungsumgebungen".

7.4 Offene Softwareentwicklungsumgebung

Den "geschlossenen" stehen "offene" Umgebungen gegenüber. Bei diesen fungiert ein zentrales Metainformationssystem als Integrationsinstrument; es definiert Schnittstellen zum Anschluß beliebiger, aufeinander abgestimmter Werkzeuge. Die Zukunft wird zweifellos den offenen Softwareentwicklungsumgebungen gehören. Der Weg dorthin ist allerdings noch weit. Bisher gibt es ein in Ansätzen realisiertes Konzept von IBM (AD/Cycle) und eine Ankündigung von DEC (Cohesion).

Das Prinzip der offenen Softwareentwicklungsumgebung wird in Abbildung 12 am Beispiel von AD/Cycle veranschaulicht. Dem IBM-Konzept liegt ein eigenes Phasenmodell mit den Phasen "Requirements", "Analysis and Design", "Produce", "Test" sowie "Production and Maintenance" zugrunde.

Im Kern der Systemarchitektur steht das ***Repository*** (die genaue Bezeichnung lautet Repository Manager), welches das Informationsmodell von AD/Cycle beherbergt. Bei diesem handelt es sich um eine logische Beschreibung aller Informationen, die im Entwicklungsprozeß eine Rolle spielen: Geschäftsprozesse, Datenklassen, Informationsflüsse, Entwurfsdokumente, Entities und Relationships, Bildschirmmasken, Programme, Kontrollfluß- und Datenflußinformationen, Testfälle u.v.a. Für seine Datenbasis nutzt der Repository Manager das Datenbanksystem DB2.

Abb. 12: Offene Softwareentwicklungsumgebung in der IBM-Welt

Die Vielzahl unterschiedlicher Informationen stellt enorme Anforderungen an das Informationsmodell. Die 1990 erstmalig ausgelieferte Pilotversion des Repository enthielt ein Informationsmodell, welches den praktischen Anforderungen nicht standhielt und von IBM als "nicht endgültig" klassifiziert wurde. Insbesondere den Softwarehäusern, welche die AD/Cycle-Werkzeuge entwickeln, waren Eigenschaften und Leistungsfähigkeit des Informationsmodells noch nicht ausreichend. Im gegenwärtigen Umfang stellt das Repository eine reine Entwicklungsdatenbank dar. Eine Unterstützung des Produktionsbetriebs - etwa in der Art, daß die Datendefinitionen des Repository direkt von einem Datenbankmanagementsystem benutzt werden können - ist nicht möglich. Für den praktischen Einsatz fehlen noch wichtige Teile.

Über den gigantischen Entwicklungsaufwand für das Repository, an dem IBM seit mehr als zehn Jahren arbeitet, kann man nur spekulieren. Im Lauf der Jahre machte auch die Zielsetzung des Projekts verschiedene Wandlungen durch. Während man zuerst ein mächtiges Data Dictionary im Auge hatte, wurde die Richtung mehrfach verändert - hin zu einem allgemeinen Meta-Informationssystem.

In Abbildung 12 kommt auch die grundsätzliche Arbeitsteilung zwischen dem zentralen Großrechner (Host) und dezentralen Workstation-Systemen zum Ausdruck; die letzteren sind im IBM-Konzept Rechner der Personal-System-Serie (PS/2). Das Integrationsinstrument Repository residiert auf dem Zentralrechner. Die Softwarewerkzeuge für die Phasen des Entwicklungsprozesses sind weitgehend der Workstation-Ebene zugeordnet. Das heißt, der Softwareentwickler arbeitet in erster Linie auf seiner Workstation. Entwicklungsergebnisse werden von den Werkzeugen im Repository abgelegt bzw. abgerufen. Zu diesem Zweck muß das zentrale Informationsmodell von allen angebundenen Werkzeugen bedient und "verstanden" werden. Den erforderlichen Darstellungstransfer zwischen verschiedenen Werkzeugen unterstützt das Repository dadurch, daß es die Ergebnisse in der einheitlichen Form des Informationsmodells ablegt.

Abbildung 12 zeigt auch deutlich den Unterschied zwischen einer offenen und einer geschlossenen Softwareentwicklungsumgebung auf: Für jede Phase sind unterschiedliche Werkzeuge zugelassen; diese müssen "nur" die Anforderung erfüllen, die Schnittstelle des Repository zu versorgen bzw. zu benutzen. Die Möglichkeit, zwischen mehreren Werkzeugen wählen zu können, ist sehr sinnvoll, da sich Eignung und Leistungsfähigkeit unterscheiden und z.B. auch vom Anwendungsproblem abhängen. Einige Werkzeuge sind in der Abbildung aufgeführt. In den meisten Fällen werden sie von Drittfirmen angeboten, ein Indiz dafür, daß auch die

Softwarebranche das zukunftsweisende AD/Cycle-Konzept aufgreift. Vielfach sind die Werkzeuge bzw. ihre Anpassung an die Repository-Schnittstelle noch in Entwicklung.

Die meisten Werkzeuge sind für die Entwurfsphase verfügbar. Dagegen wird die Implementierungsphase fast noch gar nicht unterstützt. Mit Ausnahme der KI-Werkzeuge ESE und TIRS ist zur Zeit nur CSP ("cross systems product"), ein verbreitetes 4GL-Produkt von IBM, verfügbar. Compiler für die konventionellen Sprachen der dritten Generation bzw. Generatoren, die aus den Entwurfsergebnissen automatisch Cobol-, Pl/1- oder anderen Code erzeugen, fehlen noch, ebenso Konfigurations- und Bibliotheksmanagementsysteme.

Auch die Wartungsphase ist nicht eingebunden, da es sich hierbei um ein außerordentlich schwieriges Unterfangen handelt. Damit Änderungen eines Softwaresystems, die im Zuge der Wartung (im Programmcode) durchgeführt werden, auch in den Repräsentationsformen der früheren Phasen (z.B. in SADT-Diagrammen) berücksichtigt werden können, wäre eine Art "Rückwärtsintegration" zu leisten. Davon ist Kunst noch weit entfernt.

Exkurs: Eine beschränkte Art der "rückwärts gerichteten Neustrukturierung" von Programmen wird heute unter der Bezeichnung "Re-Engineering" versucht. Dabei geht es insbesondere darum, alte Programme nach den Regeln der Strukturierten Programmierung[9)] automatisch zu reorganisieren und damit wartungsfreundlicher zu machen. Noch weiter gehen Werkzeuge, die aus dem Objekt- oder Quellcode eines Programms eine prozedurale Spezifikation, z.B. als HIPO-Diagramm, Struktogramm, Programmablaufplan o.ä., erzeugen ("Reverse Engineering").

Neben den phasenbezogenen Werkzeugen sind in AD/Cycle auch übergreifende Werkzeuge vorgesehen. Hervorzuheben ist zum Beispiel die geplante Unterstützung des Projekt- und Prozeßmanagements (Planung und Kontrolle von Ressourcen, Terminen, Ablaufstrukturen etc.).

Die weitere Ausfüllung des von AD/Cycle gesetzten Rahmens sowie die Integration der verschiedenen Werkzeuge auf den beiden Ebenen dürfte noch Jahre in Anspruch nehmen. Selbst der in Abbildung 12 dargestellte Umfang wird sich voraussichtlich erst im Lauf des Jahres 1991 oder 1992 einstellen.

8 Künstliche Intelligenz

Kaum ein Begriff hat in den letzten Jahren so viele Erwartungen geweckt wie der der "Künstlichen Intelligenz" (KI). Wenngleich die Forschungsarbeiten schon seit drei Jahrzehnten laufen, setzte ein enormer Aufschwung erst in der zweiten Hälfte der achtziger Jahre ein. Wenn man den Berg an

[9)] Vgl. Kurbel (1990), S. 71 ff.

mythischen und mystischen Spekulationen beiseite räumt, erkennt man ein breites Spektrum wissenschaftlicher Arbeiten, teilweise mit handfestem und geradezu "handwerklichem" Inhalt.

Die Künstliche Intelligenz wird hierzulande als ein Zweig der Informatik eingestuft. Wichtige Einflüsse kommen auch aus anderen Disziplinen, insbesondere der Cognitive Science bzw. der Psychologie, aber auch aus der Logik, der Pädagogik, der Linguistik und der Physiologie.

In Abbildung 13 wird versucht, das breite Feld der Künstlichen Intelligenz mit seinen vielen Teildisziplinen aufzufächern und den Einfluß anderer Wissenschaftszweige transparent zu machen. Die Forschungsarbeiten im KI-Bereich sind teilweise stärker methodisch orientiert, und teilweise stehen die Anwendungsgebiete im Vordergrund. In der Abbildung wird eine entsprechende Zuordnung vorgenommen. Dies sollte jedoch nicht als eindeutige Abgrenzung interpretiert werden, da häufig der methodische Aspekt und der Anwendungsaspekt sehr eng miteinander verbunden sind.

Die zahlreichen Teilgebiete haben sich mit unterschiedlicher Geschwindigkeit und Breite und teilweise auch anders als erwartet entwickelt. Im Rahmen dieses Berichts ist es nicht möglich, alle Teilgebiete zu erläutern. Statt dessen soll auf einige Gebiete eingegangen werden, die aus Sicht der Wirtschaftsinformatik von besonderem Interesse sind. In Abbildung 13 wurden diese durch fett ausgezeichnete Kästchen hervorgehoben.

- *Wissensrepräsentation:* Bei den Methoden der Wissensrepräsentation setzt sich eine Entwicklung fort, welche die ganze Geschichte der KI durchzieht. Während man anfangs versuchte, Wissen und menschliches Denken "an sich" zu modellieren, bildete sich in den siebziger Jahren die Erkenntnis heraus, daß die Problemlösungsfähigkeit eines Programms um so besser ist, je mehr spezifisches Wissen über einen bestimmten Problembereich es enthält. Mittlerweile hat sich herausgestellt, daß eine nach unterschiedlichen Problemklassen differenzierte Wissensrepräsentation noch bessere Ergebnisse verspricht. Die Darstellung von Wissen wird deshalb heute nach Problemklassen (z.B. Diagnose, Konfigurierung, Planung u.a.) untersucht.

- *KADS:* Wissensbasierte Systeme werden meist nach einem Prototyping-Ansatz und durch "Trial and error", häufig auch "quick and dirty" entwickelt. Dies ist aus softwaretechnischer Sicht unbefriedigend. Deshalb wird in den letzten Jahren verstärkt versucht, in Analogie zum konventionellen Software Engineering auf der Basis von Spezifikationen oder "Wissensmodellen" vorzugehen. Von den modellbasierten Ansätzen hat die KADS-Methodologie ("Knowledge Acquisition and Documentation System"[10]), die

[10]) Für die Abkürzung KADS findet man im Lauf der Zeit unterschiedliche Erklärungen. Mittlerweile hat sich die KADS-Gruppe darauf zurückgezogen, KADS nicht mehr als Akronym, sondern als Eigennamen zu betrachten!

im Rahmen eines sehr umfangreichen Esprit-Projekts entstand, den größten Bekanntheitsgrad erlangt. Auch in die Praxis findet KADS mittlerweile Eingang.

- *Fallbasierte Methoden:* Während in den heutigen Expertensystemen das Wissen explizit dargestellt wird, z.B. in Form von Regeln, gibt es wohl kaum einen menschlichen Experten, der sein Wissen in Form solcher Regeln im Kopf organisiert hat. Neben Intuition, Erfahrung und anderen unbewußten Formen des Wissens spielen insbesondere Analogieschlüsse aufgrund von ähnlichen früheren Fällen eine wichtige Rolle. Bei der fallbasierten Wissensverarbeitung wird versucht, Wissen in Form von Fällen zu erfassen und für spätere Auswertungen zur Verfügung zu stellen.

Abb. 13: Methoden und Anwendungsgebiete der Künstlichen Intelligenz

- Mit *Benutzermodellen* versucht man, das Wissen des Benutzers eines Softwaresystems über das System und über das Anwendungsgebiet sowie seine Anforderungen, Erwartungen, Berechtigungen etc. zu modellieren. Darauf aufbauend kann dann z.B. das Dialogverhalten des Systems dem Kenntnisstand des Benutzers angepaßt bzw. ihm der Zugang zu bestimmten Ressourcen (Informationen, Funktionen etc.) eingeräumt oder verwehrt werden. Benutzermodelle stellen u.a. eine wichtige Voraussetzung für intelligente Lehr- und Lernsysteme dar. Diese werden heute unter der Bezeichnung ICAI ("intelligent computer aided instruction") erforscht.

- *Benutzer(schnittstellen)management-Systeme* (UIMS, "user-interface management systems") haben die Aufgabe, die Dialogsteuerung an der Benutzerschnittstelle von den Funktionen eines Softwaresystems zu entkoppeln. Da sie sowohl Wissen über das Anwendungsgebiet als auch über den Benutzer benötigen, bedienen sie sich dazu eines Benutzermodells.

- *Expertensysteme:* Das am stärksten gewachsene Anwendungsgebiet der Künstlichen Intelligenz ist das der Expertensysteme. Hier kann man inzwischen eine starke Ausdifferenzierung nach Fachdisziplinen beobachten. Das Gebiet der Expertensysteme wird zwar auch heute noch von KI-Wissenschaftlern bearbeitet. Daneben ist aber eine starke Durchdringung der verschiedenen Anwendungsfächer zu erkennen: Betriebswirte, Maschinenbauer, Chemietechniker, Hardware-Konstrukteure und viele andere Fachwissenschaftler nutzen die Expertensystemtechnik zur Lösung ihrer spezifischen Probleme.

- Als *intelligente Datenbanksysteme* bezeichnet man einerseits Systeme, die in der Lage sind auch Wissenseinheiten (z.B. Regeln, Frames) zu speichern und zur Verfügung zu stellen ("knowledge-base management systems"), und andererseits Systeme, die komplexe Abfragen - z.B. aus einer Expertensystem-Shell heraus - bearbeiten können ("expert database systems").

- *Knowledge-Engineering Workbench:* Ähnlich wie im Software Engineering, wo zur Unterstützung der Software Life Cycle-

Phasen Werkzeuge entwickelt und nach und nach integriert wurden (CASE), gibt es nun Bemühungen, auch die Erstellung von KI-Systemen mit Computerunterstützung durchzuführen (CAKE - "computer aided knowledge engineering"). Die Aktivitäten stehen im engen Zusammenhang mit der KADS-Methodologie.

- *Neuronale Netze:* Anders als Expertensysteme enthalten neuronale Netze nicht explizit modelliertes und dargestelltes Fachwissen auf der symbolischen Ebene. Problemlösungskompetenz wird vielmehr dadurch erreicht, daß in Analogie zum menschlichen Gehirn viele Verarbeitungsknoten ("Zellen") zusammenwirken ("Synapsen"). Die Verarbeitungsmethoden sind weder algorithmisch noch heuristisch, und das Wissen wird "subsymbolisch" repräsentiert. Seine Problemlösungsfähigkeit erhält das Netz durch "Lernen", d.h., es wird mit einer möglichst großen Zahl von Fällen trainiert.

  Neuronale Netze, die in unmittelbarem Zusammenhang mit den Methoden des Konnektionismus zu sehen sind, entstanden ursprünglich im Bereich der Verarbeitung natürlicher Sprache. Dort konnten mit neuronalen Netzen bessere Erfolge erzielt werden als auf der Basis vieler der früheren linguistischen Ansätze. Aber auch im betriebswirtschaftlichen Bereich und in der Statistik gibt es bereits erfolgreiche Anwendungen, z.B. für die Vorhersage von Börsenkursen, für das Traveling Salesman-Problem und für die Datenanalyse.

9 Ausblick

Der Überblick in diesem Beitrag repräsentiert den subjektiven Stand der Erkenntnisse im Februar 1991. Da sich das hardware- und softwaretechnologische Umfeld der Wirtschaftsinformatik sehr schnell weiterentwickelt, mögen manche Aussagen ein paar Wochen oder Monate später schon überholt oder korrekturbedürftig sein. Im Hardwarebereich wird manchmal ein "Gesetz" zitiert, nach dem sich etwa die Packungsdichte der Chips alle zwei Jahre verdoppelt[11]. Der Leser soll auf die Vergänglichkeit der Aussagen mit Nachdruck hingewiesen werden.

Literatur

Bode, A.: Rechnerarchitekturen und Rechnerkategorien; in: Kurbel, Strunz (1990), S. 877-892.

[11] Vgl. Bode (1990), S. 892.

Chen, P.P.: The Entity-Relationship Model: Towards a Unified View of Data; ACM Transactions on Database Systems 1 (1976) 1, S. 9-36.

Kurbel, K.: Programmentwicklung, 5. Auflage; Wiesbaden 1990.

Kurbel, K., Eicker, S.: Ein Streifzug durch die Welt der Programmiersprachen; DSWR 17 (1988) 1-2, S. 18-25.

Kurbel, K., Moazzami, M.: Kopplung von "Elektronischen Leitständen" und PPS-Systemen in unterschiedlichen Umgebungen; in: Reuter, A. (Hrsg.), GI - 20. Jahrestagung II - Informatik auf dem Weg zum Anwender; Berlin u.a. 1990, S. 16-25.

Kurbel, K., Strunz, H. (Hrsg.): Handbuch Wirtschaftsinformatik; Stuttgart 1990.

Parsaye, K., Chignell, M., Khoshafian, S., Wong, H.: Intelligent Databases; New York u.a. 1989.

Vaske, H.: Die OSF/1-Politik der Großen bleibt unkonkret; in: Computerwoche 47, 23. November 1990, S. 31-32.

Arbeitsberichte des Instituts für Wirtschaftsinformatik

Nr. 1 Bolte, C., Kurbel, K., Moazzami, M., Pietsch, W.: Erfahrungen bei der Entwicklung eines Informationssystems auf RDBMS- und 4GL-Basis, Februar 1991.

Nr. 2 Kurbel, K.: Das technologische Umfeld der Informationsverarbeitung; Ein subjektiver "State of the Art"-Report über Hardware, Software und Paradigmen, März 1991.