Christian Wolff, Regensburg

Perspektiven moderner Softwaretechnologie[1]

Die rasant fortschreitende Entwicklung der modernen Softwaretechnik wirkt sich auch auf den Bereich des *computer-aided language learning* (*call*) aus. Im folgenden Beitrag, der sich an mit der EDV-Technik wenig vertraute DaF-Lehrer wendet, werden unter diesem Gesichtspunkt drei Themenkomplexe betrachtet:

1. Am Anfang steht ein knapper Überblick über den Stand der Technik auf dem für den computergestützen Sprachunterricht relevanten Computermarkt.
2. Eine Betrachtung derzeitiger Forschungsschwerpunkte der Softwaretechnologie aus informationswissenschaftlicher Perspektive unter Berücksichtigung von deren Relevanz für zukünftige *call*-Entwicklungen schließt sich an und
3. schließlich wird aufgezeigt, welche neueren Entwicklungswerkzeuge im Kontext graphischer Benutzerschnittstellen für die Erstellung von *call*-Software geeignet erscheinen.

1. Stand der Technik

Durch die sogenannte *PC-Revolution* Ende der siebziger Jahre wurden Computer in bisher nicht gekanntem Umfang auch für den persönlichen Bereich verfügbar. Durch die dramatisch gesunkenen Kosten zumindest für die benötigte Hardware konnten auch für den Unterrichtsbereich Möglichkeiten der Anwendung elektronischer Rechensysteme eröffnet werden. Dabei fußen die Entwicklungen im wesentlichen auf drei Rahmenbedingungen:

1.1 Hardwareentwicklung: Entwicklung des *Personal Computers (PC)*

Mit der Entwicklung der sogenannten Mikrocomputer (im Gegensatz zu den *mainframe computers, d.h.* Großrechnern und der mittleren Ebene der Minicomputer) wurde die EDV insoweit "demokratisiert", als sie auch im kleinen und für den einzelnen verfügbar wurde.
Daran haben die Einführung des sog. IBM-PCs als auch die verschiedenen Rechnermodelle der von vornherein stärker auf den individuellen Nutzer ausgerichteten amerikanischen Firma Apple entscheidenden Anteil.

---

[1] Der Beitrag ist eine erweiterte Fassung des Vortrags *Einführung in die Arbeitsweise mit dem Computer*, den ich auf der Jahrestagung DaF 1990 in Bonn gehalten habe.

Diese Hardwareentwicklung ist für den Bereich des *computer aided language learning* von Bedeutung, da erst durch die Verfügbarkeit dieser "kleinen" EDV-Anlagen ein sinnvoller Einsatz von Sprachlern- und lehrprogrammen im institutionellen Rahmen des Unterrichts denkbar wurde.[2] Dabei darf nicht das auch heute noch bestehende Gefälle zwischen den USA als Mutterland dieser Technologie und den westeuropäischen Verhältnissen, wie sie auch in Deutschland anzutreffen sind, übersehen werden: Die Verfügbarkeit von PCs im Unterrichtskontext ist in den USA weitaus üblicher, als es in Europa bisher der Fall ist. Aus diesen unterschiedlichen institutionellen und organisatorischen Rahmenbedingungen ergeben sich natürlich auch Folgerungen für die Einsetzbarkeit verschiedener Typen von *call*-Software: So kann z.B. auf Systeme mit Elementen des verteilten Schreibens oder der computergestützten kooperativen Arbeit (siehe dazu unten Kap. 2.3) zurückgegriffen werden, insoweit entsprechende Hardwareumgebungen zur Verfügung stehen.

Neben die prinzipielle Fortentwicklung der traditionellen Kernbestandteile der Hardware (d.h. die Rechnereinheit als solche, Bildschirme, Speichermodule, Festplatten etc.) selbst tritt die Innovation von Zusatzkomponenten, insbesondere im Multimedia- und Netzwerkbereich, die immer neue Formen der Interaktion mit dem Rechner bzw. der Kommunikation zwischen mehreren Benutzer an verschiedenen Computern ermöglicht.

1.2 Forschungsergebnisse auf dem Gebiet der Mensch-Maschine-Interaktion (MCI) bzw. der Software-Ergonomie

Neben der geschilderten Hardwareentwicklung ist die zweite wesentliche technologische Rahmenbedingung für die Entwicklung von *call*-Software die Fortentwicklung der Softwaretechnik. Die Entwicklung allgemein verfügbarer Standardsoftware im letzten Jahrzehnt ist weitgehend technologiegetrieben erfolgt, da erst mit der entsprechenden Hardwaregrundlage die revolutionäre Weiterentwicklung auf dem Gebiet der Softwaregestaltung möglich wurde, wie sie sich uns heute präsentiert: Dabei sind als wesentliche Neuerungen die Techniken der direkten Manipulation[3] sowie der graphischen Benutzerschnittstellen zu nennen (in ihrer technischen Realisierung erst durch leistungsfähige Graphikbildschirme ermöglicht).

---

2 In der Softwareübersicht von Desjardins et al. (1992) finden sich ausschließlich Programme für Personalcomputer, vorwiegend für IBM-PCs bzw. die (längst veralteten) Apple-II-Rechnerserien.

3 Grundlegend hierzu Shneiderman 1982.

Durch sie kann der Benutzer in einer "natürlichen", d.h. aus dem Alltagsleben vertrauten Interaktionsform an der Maschine arbeiten.[4] Die meist in eine Metapher (z.B. die Büroumgebung als Gestaltungsmetapher) eingebundene direkte Manipulation elektronischer Objekte auf der Bildschirmoberfläche findet ihr Analogon in den semiotischen Systemen der Mensch-Mensch-Kommunikation (Deixis, Kommunikation durch Handeln etc.) und kann daher als *natürlich* bezeichnet werden. Verbunden mit dieser Entwicklungstendenz im Softwarebereich ist auch das Aufkeimen der Software-Ergonomie als die menschengerechte Erstellung und Verwendung von Software thematisierende Wissenschaft.[5] Damit ist die Gestaltung der Mensch-Maschine-Schnittstelle (unabhängig von den funktionalen Anforderungen) stärker in den Mittelpunkt der Software-Entwicklung gerückt; für die kommerzielle Softwareerstellung kann von einem Kostenanteil von etwa 50% für die Schnittstellengestaltung ausgegangen werden. Erfreulich ist in diesem Zusammenhang auch das so entstandene Problembewußtsein für Fragen der Mensch-Maschine-Schnittstelle: Die an sich selbstverständliche Bedeutung der Art der Vermittlung des Mensch-Maschine-Dialogs konnte sich einen eigenen Platz neben der stärker informatikorientierten Funktionsproblematik erobern.

Auf einer globalen Ebene sind allerdings weiterhin falsche Vorstellungen in Bezug auf die Optimierungsmöglichkeiten der Softwareentwicklung weit verbreitet: Das durch die mit der Einführung der graphischen Benutzerschnittstellen und intelligenter Unterstützungsverfahren für Softwaresysteme verbundene Trugbild von der vollständigen Selbsterklärungsfähigkeit[6] von Softwaresystemen ist noch nicht überwunden; eine interaktionsfehlerfreie Gestaltung des Dialogs mit dem Rechner ist aus prinzipiellen Gründen nicht möglich, jede Optimierung wird mit steigendem Komplexitätsgrad der Software immer schwieriger. Die Softwareentwicklung muß in dieser Hinsicht von vorneherein die möglichen Schwachpunkte einkalkuieren und durch geeignete Hilfsmittel abzufedern versuchen, etwa durch Hilfesysteme, benutzerseitige Gestaltbarkeit (Adaptierbarkeit) der Software, oder systemseitige Anpassung (Adaptivität) der Systeme an spezifische Benutzerbedürfnisse.

---

4 Zum Begriff der *Natürlichkeit* als Kriterium der Benutzerschnittstellenanalyse cf. Krause 1992:162ff.

5 Einen umfassenden Überblick zur Softwareergonomie gibt Balzert et al. 1988, vgl. aber auch den einführenden Aufsatz von Krause (in diesem Band, S. 253ff).

6 Die Problematik der Selbsterklärungsfähigkeit wird ausführlich diskutiert bei Herrmann 1986.

Auch für den *call*-Bereich darf die Tragweite dieser Problematik nicht verkannt werden: Die Fehlerhaftigkeit des interaktionsbezogenen Aspektes der Arbeit am Rechner bringt ein Gefahrenpotential auch für die eigentlichen Ziele des Sprachlern- und lehrprogrammeinsatzes mit sich: Die zusätzliche Ebene möglichen fehlerhaften Verhaltens kann sich als Interferenz auf das gesteckte Lernziel negativ auswirken, insofern dem jeweiligen Benutzer die Trennung zwischen interaktionsbedingtem Fehler und sachlicher Problemstellung nicht klar wird; werden die Ebenen vertauscht oder verwechselt, ist auch das pädagogische Ziel der erfolgreichen Sprachvermittlung gefährdet.

1.3 Der Entwicklungsstand der *call*-Technik insbesondere

Nach einer gewissen Anfangseuphorie Mitte der 80er Jahre scheint in den letzten Jahren wieder eine Beruhigung auf dem *call*-Softwaremarkt eingetreten zu sein.[7] Dabei ist die Entwicklung noch im Fluß - *call* hat sich als Unterrichtsebene noch keineswegs auf breiter Front durchgesetzt.

In funktionaler Hinsicht lassen sich zwei Programmtypen in diesem Bereich unterscheiden:

Zum einen eigenständige fertige Anwendungsprogramme (sog. *turnkey programs*), die jeweils eine bestimmte sprachliche Übung beinhalten.[8] Diese bilden die Mehrheit des verfügbaren Programmbestands.

Auf der anderen Seite stehen die sogenannten *Autorensysteme*, die es dem Lehrenden ermöglichen, in einem vorgegebenen Entwicklungsrahmen jeweils eigene Übungen als Programme zu erstellen. Das Prinzip des Autorensystems findet sich auch in den heute üblichen Applikationsgeneratoren für Anwendungsprogramme wieder (dazu siehe unten Kap. 3).

Dem Vorteil, Übungen selbst produzieren zu können, stehen in der Regel natürlich auch die Nachteile der eingeschränkten funktionalen Anwendbarkeit gegenüber: Meist ist das Autorensystem nur auf eine bestimmte Programmart oder Übungsform zugeschnitten, in deren Rahmen Übungen variiert werden können. Demgegenüber ist es natürlich von Vorteil, wenn eine wesentliche Übungsform von dem Autorensystem abgedeckt wird, in gewohnter Umgebung beliebige Varianten ausarbeiten zu können.

---

7 Vgl. die Statistik bei Desjardins 1992:147.

8 Z.B. die bei Albers & Weiland in diesem Band (S. 279ff) vorgestellten Programme.

Betrachtet man die Gesamtheit der heute zur Verfügung stehenden *call*-Software, lassen sich als allgemeine Charakteristika die folgenden Eigenheiten feststellen:

Aufgrund der institutionellen Gegebenheiten der *call*-Entwicklungen ist in bezug auf die Übereinstimmung mit neueren Entwicklungen der Softwaretechnik ein gewisser Nachhängeeffekt festzustellen, d.h. die meisten Entwicklungen basieren noch auf den inzwischen veralteten Prinzipien der zeichenorientierten Interfacestrukturierung; graphische Oberflächen und direkte Manipulation sind bisher noch kaum integriert. Mit dem Fortschreiten der Entwicklung kommerzieller *call*-Systeme dürfte sich diese Disparität in der nächsten Zeit aber ausgleichen, da einerseits aus den USA Programme für den europäischen Markt angepaßt werden (insbesondere die vielfältige Apple-Macintosh-Software), andererseits graphische Benutzerschnittstellen bereits in so hohem Maße als Standardvoraussetzung bei der Softwareentwicklung gesehen werden können, daß von zukünftigen Neuentwicklungen von vorneherein die Konformität mit den Prinzipien dieser Softwaretechnik erwartet werden kann.

Nur in ersten Ansätzen wurde bisher versucht, aktuelle Entwicklungstrends sowohl der Hardware- als auch der Softwareforschung - hier sind die sog. intelligenten und wissensbasierten Systeme zu nennen - für die Entwicklung von *call*-Software fruchtbar zu machen.[9]

2. Entwicklungstendenzen in der Softwareforschung

In Informatik, künstlicher Intelligenz-Forschung und Informationswissenschaft gibt es eine ganze Reihe von Gebieten, die auch für die zukünftige Entwicklung von *call*-Systemen eine Rolle spielen können. Hierzu wird im folgenden je eine kurze wertende und z.T. sicher etwas vereinfachende Betrachtung in Hinblick auf die kurz- und mittelfristige Verfügbarkeit im Rahmen von *call*-Programmen gegeben.

Im Vordergrund steht eine informationswissenschaftliche Betrachtungsperspektive, die in Bezug auf die Softwareentwicklung zum einen den methodischen Primat der Pragmatik postuliert,[10] d.h. die jeweils konkreten Randbedingungen des Softwareeinsatzes als maßgebliche Beurteilungsfak-

[9] Vgl. aber die Beiträge von Rüschoff (Integration digitalisierter Tondokumente) und Piesche-Blumtritt (regelbasierte Systemgestaltung) in diesem Band (S. 325ff bzw. 353ff).

[10] Dazu grundlegend Kuhlen 1990.

toren für Entwicklungstendenzen in der Softwaretechnik begreift: Am Anfang steht nicht die prinzipielle Frage nach der Machbarkeit etwa planerkennender Systeme (z.B. als Kontrollalgorithmen für Sprachübungsprogramme), sondern die aktuelle Bedarfsanalyse des gegebenen Anwendungskontextes (also etwa eines Übungsprogramms zur Morphologie).
Hinzu tritt bei der Beurteilung die informationswissenschaftliche Kernforderung nach dem durch ein Übungsprogramm zu verwirklichenden *informationellen Mehrwert*[11] in bezug sowohl auf den prinzipiellen Einsatz von Software als auch auf den Einsatz einer zusätzlichen Technologie.
Anders ausgedrückt, muß zunächst abgeschätzt werden, ob *call*-Programme an sich eine Verbesserung oder Bereicherung der DaF-Unterrichtssituation leisten können. Bejaht man diese Frage, so ist weiter zu prüfen, ob bei der Gestaltung von Übungsprogrammen auch der Einsatz komplexerer "intelligenter" Softwaretechniken sinnvoll sein kann, dazu Näheres im folgenden Kapitel.

2.1 Sprachverstehende Systeme

Für *call*-Software bietet es sich an, auf die Entwicklungen der Computerlinguistik insoweit zurückzugreifen, als damit die sprachliche Analyse oder Generierung in einem Übungsprogramm nicht mehr von fest vorgegebenen Ablaufstrukturen abhängig gemacht werden müßte, sondern dynamisch den *Parsingalgorithmen*[12] des Programms überlassen werden könnte.
Die kurz- und mittelfristigen Entwicklungschancen für in diesem Sinne intelligente *call*-Software müssen jedoch als außerordentlich schlecht angesehen werden. Drei maßgebliche Gründe seien hierfür genannt:

a) der Entwicklungsstand der Sprachtechnologie läßt zwar bereits den Einsatz von sprachverstehenden Systemen in eng umgrenzten Anwendungsgebieten zu, es kann aber nicht davon gesprochen werden, daß Sprachverarbeitungsalgorithmen nach Art des Baukastenprinzips fertig zur Integration in ein *call*-Programm vorlägen. In jedem Fall wären erhebliche Anpassungsmaßnahmen an den sprachlichen Anwendungsfall notwendig, die wiederum in der Regel dem erreichbaren Ziel gegenüber als zu aufwendig erscheinen müssen.

---

11 Hennings 1991:8 geht bei der Umwandlung von Wissen in Information davon aus, daß "dies dann als Schaffung von **informationellem Mehrwert** oder der ganze Prozeß auch als **informationelle Mehrwertgenerierung** bezeichnet werden kann."

12 Als *Parser* bezeichnet man einen Algorithmus, der es erlaubt, natürliche Sprache zu analysieren.

b) Exisitierende sprachverstehende Systeme sind auf die Verarbeitung eines korrekten syntaktischen, semantischen und lexikalischen Subsets hin ausgelegt; diese Einschränkung liegt zwar auch bei sprachlichen Einzelübungen vor, die Algorithmen haben aber eben gerade das Ziel, korrekte Sprachkonstrukte zu erkennen, nicht aber fehlerhafte Äußerungen. Diese Fähigkeit wäre aber für ein im *call*-Bereich eingesetztes sprachverstehendes System unerläßlich. Es erscheint kaum realistisch, Parsingtechnologien zu entwickeln, die neben der Fähigkeit, korrekte Äußerungen zu verarbeiten auch über das sprachliche *Metawissen* zur Fehlerbeurteilung verfügen.

c) Für die Mensch-Maschine-Interaktion in natürlicher Sprache kann es aufgrund des *computer talk*-Phänomens[13] zu Interferenzen mit dem gewünschten Lernziel kommen: Die *computer talk*-Forschung hat aufgezeigt, daß in natürlichsprachlichen Mensch-Rechner-Dialogen ein für diese Situation typisches Sprachregister (*computer talk)* vom Benutzer angewandt wird, das charakteristische sprachliche Merkmale v.a. in Bezug auf die verwendete Syntax aufweist (vergleichbar dem sog. *foreigner talk* oder auch dem *baby talk*), die in der "normalen" Kommunikation als fehlerhaft bewertet werden müßten.

In der Zusammenschau ergibt sich für den Einsatz sprachverstehender Systeme in *call*-Programmen ein eher düsteres Bild: Sie sind im allgemeinen zu wenig leistungsfähig, zu aufwendig in der Entwicklung, zu schlecht an die Notwendigkeit der Fehleranalyse im *call*-Bereich abgestimmt und bringen durch die situative Einbettung in die MCI auch Gefahrenpotentiale für den Lernerfolg (oben c)).

2.2 Expertensysteme und neuronale Netze: Anwendung von Techniken der künstlichen Intelligenz

Expertensysteme (XPS) stellen den wohl bisher kommerziell erfogreichsten Erfolg der künstlichen Intelligenzforschung (KI)[14] dar. Ursprünglich in den 60er Jahren eher als Nebenprodukt des "großen Ziels" entstanden, ein allgemein gültiges, rechnerbasiertes intelligentes Problemlösungssystem zu entwickeln (den sog. *general problem solver, GPS*), sind Expertensysteme heute für vielfältige Anwendungen (klassische Beispiele: Analyse

[13] Cf. Krause 1992:157: "[...] kann die Existenz eines computer talk in der natürlichsprachlichen MCI als nachgewiesen gelten."

[14] Nach wie vor ist der Begriff der künstlichen Intelligenz umstritten, insbesondere was das Verständnis von Intelligenz anbelangt, vgl. dazu die Diskussion des Intelligenzbegriffs in der lesenswerten KI-Einführung von Simon 1984.

chemischer Strukturen, automatische Konfiguration von Computersystemen, Analyse geomorpholohischer Daten) einsetzbar.

Dabei besteht allerdings bisher immer noch ein krasses Mißverhältnis zwischen Forschungsinvestition und -aufwand und wirtschftlichem Nutzen, d.h. man sollte sich vor einer Überbewertung des Nutzens der XPS-Technologie hüten.
Durch den partiellen Erfolg der künstlichen Intelligenz-Forschung ist allerdings im allgmeinen Sprachgebrauch der Begiff des Expertensystems stark verwässert. Auch im *call*-Bereich sollte nicht jedes System, das sprachliches Wissen inkorporiert, schon aufgrund dieser Tatsache als "intelligentes System" oder "Expertensystem" bezeichnet werden. Die komplexe Struktur eines XPS mit Wissensbasis, Regelinventar und Steuerungsmodul wird wohl kaum irgend ein *call*-System vollständig umgesetzt haben.

Ansonsten gilt auch die schon bei der Beurteilung natürlichsprachlicher Systeme ausgesprochene Warnung zur Vorsicht: Die komplexen Anforderungen an die Entwicklung von XPS übersteigen schnell den potentiellen Nutzen eines praktisch anwendbaren Sprachlernprogramms. Für die Anwendungen der XPS-Technologien "im kleinen" gibt es allerdings erste brauchbare Werkzeuge (s.u. Kap. 3.3).

Im letzten Jahrzehnt ist auch der urpsünglich bereits in den 40er Jahren entwickelte Gedanke der neuronalen Netzte als Systemstrukturierung in Analogie zur Funktionsweise des menschlichen Nervenapparats zum Anwendungsfall der künstlichen Intelligenzforschung geworden.[15] Der Grundgedanke ist dabei, Systeme zu entwickeln, deren Organisation an die Verschaltung von Nervenmsynapsen angelehnt ist. Solche Systeme sind fähig, automatisch hinzuzulernen und Dateninput zur Umstrukturierung des Systems selbst zu verwenden. Dieses spannende Forschungsgebiet konnte in kurzer Zeit bereits gewisse Erfolge verbuchen, etwa bei der Entwicklung von Mustererkennungsalgorithmen. Von einer praktischen Nutzung für den *call*-Bereich dürften die neuronalen Netze allerdings auch mittelfristig noch weit entfernt sein.

[15] Zur damit verbundenen wissenschaftlichen Richtung des Konnektionismus innerhalb der KI-Forschung vgl. Smolensky 1988.

2.3 Nutzung vernetzter Strukturen

Nachdem die Individualisierung des Computergebrauchs durch die *PC-Revolution* die Massenverfügbarkeit der Rechnertechnik die Entwicklung der 80er Jahre dominiert hatte, ist seit einigen Jahren eine gegenläufige Tendenz zu beobachten: In immer stärkerem Maße werden PCs (meist durch sog. *Local Area Networks, LANs*) miteinander vernetzt, die Nachteile der individuellen Nutzung (Wartungsproblematik, Schwierigkeiten beim Datenaustausch und der Arbeitskooperation etc.) wurden deutlich.

Die Vernetzung bringt auch für den *call*-Bereich neue Entwicklungschancen: In den meisten Fällen handelt es sich beim Sprachunterricht um eine kommunikative Situation, bei der dem Lehrenden eine größere Anzahl von Lernern gegenübersteht. Die typische Anwendung von *call*-Standardsoftware sieht aber bisher den PC-Einzelarbeitsplatz als Dialogsituation vor.
Durch die Vernetzung und die Entwicklung sog. verteilter Systeme ließe sich nun auch der typischen Unterrichtssituation angemessene im Netz zu nutzende Software mit entsprechenden Lerner- und Leherermodulen entwickeln - eine Analogie zu den Möglichkeiten des Sprachlabors drängt sich an dieser Stelle auf.
Auch hier ist allerdings zu erwarten, daß das Paradigma der *computer supported cooperative work (CSCW)*[16], das als typische Anwendungsgebiete das verteilte Arbeiten am Computer in Betrieben oder die Modellierung von Kommunikationsvorgängen in Besprechungen vorsieht, erst in absehbarer Zeit in den *call*-Bereich übernommen wird.[17]

2.4 Hypertext und Multimedia

Ein weiteres Beispiel für die technologiegetriebene[18] Entwicklung der Softwaretechnik gibt der Bereich Multimedia ab: Mit der hohen Innovationsgeschwindigkeit bei gleichzeitigem erheblichen Preisverfall[19] ist der soge-

---

16 Einen kurzen Überblick hierzu gibt Greenberg 1991.

17 Die Zusammenstellung von DaF-bezogener call-Software bei Desjardins et. al 1992 enthält bezeichnenderweise weder Hinweise auf netzwerkfähige Software noch werden Grundbegriffe wie Netzwerk, *LAN* oder *CSCW* im Glossar erwähnt.

18 Die Vorstellung von der technologiegetriebenen Forschungsentwicklung darf nicht zu eng verstanden werden: In vielen Fällen wird durch technische Innovation lediglich die praktische Umsetzung einer bereits lange existierenden Idee möglich, so bei den neuronalen Netzen und beim Hypertext, für die die grundlegenden theoretischen Arbeiten bereits aus den 40er Jahren stammen.

19 Typische Hardwareergänzungen wie CD-ROM Laufwerke oder Soundkarten sind heute schon relativ preiswert im Fachhandel erhältlich; ihre Installation und Handhabung ist in der Regel problemlos.

nannte *multimedia PC* (MPC) schneller als erwartet Wirklichkeit geworden. Damit werden aber für die Aufbereitung von Sprachlehrmaterialien in *call*-Software auf breiter Basis neue Möglichkeiten geschaffen:

- Bilder wie Bildsequenzen (*Animation*) können in die Programme integriert werden und den Sprachlehrgehalt stützen (etwa in Form von Bildwörterbüchern).
- Digitalisierte Sprache macht *call*-Programme auch zu einem geeigneten Medium für das Training der gesprochenen Sprache.[20]
- Entsprechende Aufnahmemöglichkeiten können - wie im Sprachlabor - auch für die Korrektur und Bewertung durch den Lehrer genutzt werden.

Die vorläufig noch einschränkenden Nachteile des erheblich erhöhten Speicherplatzbedarfs bei Bild- und Sprachverarbeitung dürften aufgrund verbesserter Datenkompressionstechniken sowie weiteren Preisverfalls bei den Speichermedien in der absehbaren Zukunft weiter entschärft werden.

Eng verbunden mit der Entwicklung hin zur Nutzung von Multimedia - Techniken ist die Umsetzung des Hypertextgedankens in Rechnersystemen, der eine nicht lienare Informationsstrukturierung als wesentliches Gestaltungsphänomen nutzt:[21]

Texte, Bilder, Graphiken oder auch Tondokumente können in Hypertextstrukturen hierarchisch oder auch netzwerkartig durch Hypertext-*Links* so verbunden werden, daß dem Benutzer verschiedene Sichten und Betrachtungspfade in bezug auf den zugrundegelegten Text (im weiteren Sinn) möglich werden: Im Normalfall kann er bei der Lektüre des Basistextes an verschiedenen Stellen Verknüpfungen folgen, die ihn aus der linearen Textrezeption herauslösen und auf verwandte Themengebiete verweisen, einen Begriff näher erläutern oder multimediale Zusatzinformation präsentieren. Die Nutzbarkeit dieses Strukturprinzips bei der Erstellung von *call*-Software ist evident:
Gerade die Entwicklung der modernen Sprachlehrbücher gibt ein anschauliches Beispiel für die Notwendigkeit, von der früher stark ausgeprägten Orientierung am lienaren Text abzuweichen und Arbeitsmaterialien durch Bilder Verweise und Hintergrundinformationen zu ergänzen. Damit ist auch eine Flexibilisierung des Lernprozeßes verbunden, da ausgehend von

[20] Cf. etwa den Beitrag von Rüschoff in diesem Band (S. 325ff).

[21] Kuhlen 1991:6 wählt die folgende Definition: "Hypertext ist von der Grundkonzeption her eine nicht-lineare Form der Darstellung bzw. der Aneigung von Wissen."

Basismaterialien vielfältige Verzweigungsmöglichkeiten enstehen, die bei Bedarf eine Vertiefung des Lernstoffes erlauben.
Optimistisch darf man in bezug auf die Umsetzung des Hypertextprinzips in *call*-Software auch deshalb sein, weil es hierfür bereits ebenso leistungsfähige wie einfach zu handhabende Entwicklungstools gibt (cf. Kap. 3.2).

3. Moderne Werkzeuge der Softwareentwicklung

Im Anschluß an die Diskussion der Entwicklungstendenzen in der Softwaretechnik sollen nun exemplarisch geeignete Entwicklungswerkzeuge vorgestellt werden, mit denen sich *call*-Programme realisieren lassen.

Allen besprochenen Werkzeugen ist folgendes gemeinsam:

a) Sie sind bereits in einer *graphischen Benutzerumgebung* realisiert (hier: MS-Windows[22]). Lange Zeit war die Entwicklung von Software für grapische Benutzeroberflächen dadurch gehemmt, daß der Benutzerfreundlichkeit der Oberfläche wegen der sehr komplexen Programmierung ein gesteigerter Entwicklungsaufwand gegenüber stand. Erst seit Anfang der 90er Jahre sind leistungsfähige Applikationsgeneratoren verfügbar, die dieses Problem überwinden helfen.

b) Es handelt sich dabei um *Entwicklungsshells*, die jeweils mit eigener Programmiersprache versehen sind und den Benutzer nicht auf das Erlernen einer der traditionellen prozeduralen Programmiersprachen (etwa Basic, C oder Pascal) verweisen, gleichzeitig aber in einer fremden Programmiersprache geschriebene Module integrieren können.

c) Die Einbettung in eine graphische Benutzerschnittstelle bedeutet, daß auch die Bildschirmgestaltung für das Zielprogramm (also hier die *call*-Software) mit den Mitteln der direkten Manipulation durch *Dialogeditoren* erfolgen kann (*visuelle Programmierung*). Dies erleichtert den Aufbau des gewünschten Programms erheblich.

d) Die Applikationsgeneratoren sind in der Regel auf eine *bestimmte Zielfunktionalität* hin ausgerichtet (z.B. Schwerpunkt Datenbanken).

e) Ihrer Funktionsweise nach sind diese Applikationsgeneratoren den im *call*-Bereich traditionell stark vertretenen Autorenprogrammen verwandt, da die vorgegebene Entweicklungsumgebung gleichzeitig die Programm-

[22] Da es sich bei MS-WINDOWS um die mit Abstand am weitesten verbreitete graphische Benutzerschnittstelle handelt, erschien es ratsam, die Beispiele aus diesem Bereich zu wählen; gleichzeitig sei aber darauf hingewiesen, daß einige der genannten Werkzeuge ursprünglich für den Apple Macintosh entwickelt worden waren, für den es eine Alternative zur graphischen Benutzerschnittstelle wie MS-DOS auf dem PC nie gegeben hat.

erstellung vereinfacht und die Entwicklungsmöglichkeiten auf eine bestimmte funktionale Bandbreite eingrenzt.

3.1 SQL-WINDOWS (Schwerpunkt Datenbanken)

Hierbei handelt es sich um einen Generator für Datenbankschnittstellen. Seine Relevanz für den *call*-Bereich ergibt sich aus der Problematik, größere Mengen sprachlichen Materials in einem Programm verarbeiten zu können (etwa in einem Lexikon).
Neben der visuellen Dialoggestaltung, die auch für den Umgang mit (hier: relationalen) Datenbanken besonders geeignete Bildschirmelemente wie Tabellen vorgefertigt anbietet ist für dieses System besonders die vollständige Integration der Datenbankabfragefunktionen und ihre Anbindung an das Interface hervorzuheben.

In einer gut erlernbaren Beschreibungssprache kann der Entwickler den Programmentwurf, den er im Dialogeditor gestaltet hat, mit der entsprechenden Semantik der Datenbankabfrage verbinden.

3.2. TOOLBOOK (Schwerpunkt Multimedia und Hypertext)

Die Funktionalität von TOOLBOOK ist auf die Entwicklung von Multimedia- und Hypertextanwendungen hin ausgerichtet.

Dabei erfolgt die Programmstrukturierung in Analogie zum Buchkonzept: Das fertige Programm wird als Buch aufgefaßt, die einzelnen Bildschirme als dessen Seiten. Jedes Bildschirmelement kann mit Hintergrundtexten, Bildern oder Klängen verknüpft, die einzelnen Seiten durch Verweisungen (Links) miteinander verbunden werden. Besonders geeignet dürfte Toolbook für die Gestaltung von Unterrichtsmaterialien sein, die dann Grundlage weitere Sprachübungen sein sollen.

3.3 KAPPA-PC (Schwerpunkt wissensbasiertes Entwickeln)

KAPPA-PC ist eines der wenigen Entwicklungswerkezeuge, das es erlaubt, mit einfachen Mitteln XPS-Techniken für PC-Programme zu verwenden. Neben dem Dialogeditor steht dem Benutzer hier insbesondere ein Werkzeug zur Erstellung von Objekthierachien zur Verfügung, das eine Erstellung eines Wissensbaumes möglich macht. An einzelne Bildschirmelemente oder Instanzen und Klassen in der Objekthierarchie können Regeln gebunden werden, die nach verschiedenen Verfahren (*forward chaining* und *backward*

chaining) abgearbeitet werden können. Dabei können auch Gewichtungen für die Priorität der Regelanwendung vergeben werden.

Nach dem oben Gesagten ist fraglich, inwieweit auch bei einfachen sprachlichen Problemen der Aufwand des Aufbaus eines Regelsystems nicht bereits den erhofften Nutzen übersteigt; andererseits ist es durchaus denkbar, ausgewählte (und stark formalisierbare) Kapitel der deutschen Grammatik in einem Regelgeflecht in KAPPA-PC aufzubereiten und dem Lernenden in einer Übung zu präsentieren. Hervorzuheben ist bei diesen System die gelungene Integration von graphischer Benutzerschnittstelle und visualisierter XPS-Technologie.

3.4 Authorware (Schwerpunkt Tutoren, programmierter Unterricht und Multimedia

Das letzte hier vorzustellende Produkt, Authorware, ist grundsätzlich auf die Erstellung tutorieller Systeme hin ausgerichtet. Damit ist es schon von der Konzeption her auf die Lernsituation hin ausgelegt. Es bietet wohl auch das beste Beispiel für die visuelle Programmierung im Rahmen derartiger Entwicklungsshells: Der Programmablaufplan kann vom Benutzer interaktiv mit der Maus durch Erstellen einer Art Datenflußdiagramm gesteuert werden. Dabei sind Verschachtelungen in beliebiger Tiefe möglich, sodaß auch bei komplexen Programmen die Übersicht nicht verloren gehen muß.

Gleichzeitig bietet es stärker als andere Systeme die Möglichkeit der Einbindung von Bildern und Animationen.

Besonders geeignet erscheint es für die Umsetzung des sog. "programmierten Unterrichts" auf dem PC, da Ablaufvarianten und Verzweigungen besonders einfach angelegt werden können und hierfür nicht einmal mehr der Rückgriff auf eine (wenn auch vereinfachte) Programmiersprache notwendig ist.

4. Fazit

Bemerkenswerte Grundkonstituente der Entwicklung von *call*-Software ist für die 80er Jahre das Zurückhängen hinter der allgemeinen Entwicklung der Softwaretechnik. Gleichzeitig hat aber die rasante Innovationsgeschwindigkeit auf dem PC-Sektor in großer Zahl neue Gestaltungsmöglichkeiten hervorgebracht. Während einige Forschungsschwerpunkte im Bereich der "künstlichen Intelligenz" hinsichtlich der konkreten Umsetzbarkeit in *call*-Software noch weit von der Praxisnähe entfernt sind, konnten andere

Konzepte (Hypertext, Datenbankschnittstellen, Multimedialität) bereits erfolgreich eingesetzt werden. Die Vielzahl neuer Entwicklungsumgebungen insbesondere für graphische Benutzerschnittstellen ist in vollem Umfang für *call*-Software nutzbar und läßt auf eine gesteigerte Produktivität (erweiterte funktionale wie gestalterische Möglichkeiten bei vereinfachter Entwicklungskomplexität) in diesem Bereich hoffen.

Literaturverzeichnis

Asymetrix Corp. (edd.) (1989). Using ToolBook. Bellevue/WA.: Asymetrix Corporation.

Authorware, Inc. (edd.) (1991). Authorware Professional for Windows. Reference Manual.

Balzert, H. et. al (edd.) (1988). Einführung in die Software-Ergonomie. Berlin/New York: deGruyter [= Mensch Computer Kommunikation Grundwissen Bd. 1].

Desjardins, Lise; Martin, Bernhard; Walthi, Katrin (1992). Dokumentation zum computergestützten Unterricht in Deutsch als Fremdsprache. Hrsg. v. Hans-Herbert S. Räkel & Thomas Steinfeld [= InfoDaF 19 (2) (1992)].

Greenberg, Saul (1991). "Computer-Supported Cooperative Work and Groupware: an Introduction to the Special Issues." In: International Journal of Man-Machine Studies 34 (1991), 133-141.

Gupta Technologies, Inc. (edd.) (1991). SQLWindows Technical Reference Manual.

Hennings, Ralf-Dirk (1991). Informations- und Wissensverarbeitung. Berlin: De Gruyter [= Mensch Computer Kommunikation Bd.6].

Herrmann, Thomas (1986). Zur Gestaltung der Mensch-Computer-Interaktion: Systemerklärung als kommunikatives Problem. Tübingen: Niemeyer.

Intellicorp, Inc. (edd.) (1992). KAPPA-PC Quick Start. Version 2.

Krause, Jürgen; Hitzenberger, Ludwig (edd.) (1992). Computer Talk. Heidelberg et., al: Olms [= Sprache und Computer Bd. 12].

Kuhlen, Rainer (1990). "Zum Stand pragmatischer Forschung in der Informationswissenschaft." In: Herget, Josef; Kuhlen, Rainer (edd.) (1990). Pragmatische Aspekte beim Entwurf und Betrieb von Informationssystemen. Proc. 1. Internationales Symposium für Informationswissenschaft (ISI '90). Konstanz: Universitätsverlag [= Konstanzer Schriften zur Informationswissenschaft Bd. 1], 13-18.

Kuhlen, Rainer (1991). Hypertext. Ein nicht-lineares Medium zwischen Buch und Wissensbank.Berlin et. al: Springer.

Langenscheidt-Redaktion (edd.) (1985). Computergestützter Fremdsprachenunterricht. Ein Handbuch. Berlin et. al: Langenscheidt.

Simon, G.L. (1984). Introduction to Artificial Intelligence.

Smolensky, Paul (1988). "On the proper treatment of connectionism." In: Behavioral and Brain Sciences 11 (1988), 1-23.