Marc-Oliver Pahl

Evolution der Neuen Medien in der Lehre

entstanden zum eLearning-Seminar von Frank Hanisch
am Institut für graphisch interaktive Systeme der Universität Tübingen
Sommersemester 2003

[Abstract]
Im ersten Teil der Arbeit werden einige, für die Lehre wichtige, Stationen auf dem Weg vom Zahlen verarbeitenden Stapelrechner zum Wissen verarbeitenden Desktop-Betriebssystem erläutert. Es wird auf die Pionierleistung von Bush, Nelson, van Dam, Engelbart, Meyrowitz und Kay eingegangen.
Im zweiten Teil wird ausgehend von [Meyrowitz 1989] anhand aktueller Beispiele aufgezeigt, was an heutigen Betriebssystemen zu verbessern wäre und welche Folgen das für das Lehren und Lernen mit dem Computer hätte.

Inhalt

“euphoria and hype struggle to produce material for the new medium
mature judgement disappointment and cynicism wait for the next technology
that will be the answer”

Anon

Einleitung

Der größte Teil der heute eingesetzten Software ist nicht in der Lage, zu interagieren. Produkt A kann im Normalfall nicht mit Produkt B Daten austauschen. Insbesondere für eine Lernumgebung ist dies hinderlich. Eine Kurseinheit, die sich mit Excel[1] befasst, kann das didaktisch Naheliegendste – nämlich den Lernenden direkt mit Excel lernen lassen – nicht tun, weil Excel keine Schnittstelle anbietet, mit deren Hilfe man das Handeln verfolgen könnte. Konkret heißt das, dass ich als Autor eines solchen Kurses entweder innerhalb meiner Lernumgebung Funktionalität von Excel nachbauen muss (was viel Zeit in Anspruch nimmt und niemals das Gefühl der echten Anwendung vermittelt) oder aber dem Lernenden viel Verantwortung (und Fehlerpotential) in die Hand gebe, indem ich ihn ohne die Kontrolle durch mein Lernmodul „blind“ in Excel agieren lasse. Bei Problemen steht dann in meinem Lernmodul vielleicht eine Hilfe zur Verfügung, in der der Lerner selbsttätig Unterstützung suchen muss (und hoffentlich findet). Eine gezielte Lenkung und eine Kontrolle des Ergebnisses (und Lernfortschrittes) sind nicht möglich.
Das gleiche Problem ergibt sich auch in der umgekehrten Richtung, wenn ich nicht mit meinem Lernmodul in eine Anwendung eingreifen möchte, sondern etwas in mein Lernmodul integrieren will. Das könnte beispielsweise eine 3D-Welt sein, die ich in Maya[2] modelliert habe. Für sich genommen wird die Welt funktionieren, aber da Maya nicht für interaktives Lernen gedacht ist, fehlt die komplette lernspezifische Unterstützung und somit kann ich die Welt nicht einsetzen.
Der Mangel liegt dabei weniger bei den Anwendungen als viel mehr beim Betriebssystem, auf das jede der Anwendungen aufsetzt. Die heute verbreiteten Systeme[3] sehen keine ausreichende Möglichkeit der Kommunikation zwischen Anwendungen vor. Wie wir im ersten Teil sehen werden lief die historische Entwicklung nicht unbedingt in Richtung der heutigen Daten- und Programminseln. Viele der entstandenen Ideen und Konzepte sind aber wieder in Vergessenheit geraten, was es um so interessanter macht, sie aufzuzeigen.
Im zweiten Teil werde ich dann ausgehend von einem Artikel von Norm Meyrowitz aus dem Jahre 1989 einige Verbesserungsvorschläge diskutieren, die unter anderem die eingangs geschilderten Probleme lösten.

[1] Tabellenkalkulation
[2] 3D-Modellierungwerkzeug
[3] Sommer 2003: Windows 9x/ NT/ 2000/ XP; Mac OS X

Historie

Die ersten Computer[4] waren Stapelverarbeitungsmaschinen. Auf der einen Seite legte man sein Programm in einen Lochkartenstapel und einige Zeit später konnte man auf der anderen Seite ein Ergebnis ebenfalls in Form einer Lochkarte abholen. Eine direkte Interaktion war nicht möglich und für etwas anderes, als zum Zählen und zum Rechnen waren sie nicht gedacht. [Pahl 2003]
Deshalb war es ein großer gedanklicher Schritt, den Vannevar Bush im Jahre 1945 vollzog: Er erdachte ein System zur Verarbeitung von Wissen. Das Ablegen von wissenschaftlichen Arbeiten in Aktenordnern erschien Bush verbesserungsbedürftig: Das Auffinden eines bestimmten Artikels dauert lange und die meist folgende Recherche nach ähnlichen oder weiterführenden Informationen verschlingt wieder viel Zeit, selbst wenn diese schon angegeben sind. In seinem Artikel „As we may think" beschreibt Bush den Memory Extender (Memex), eine intelligente Ablage bei der alle Artikel auf Mikrofilmen gespeichert und per Knopfdruck geladen und angezeigt werden – die manuelle Suche entfällt. Für Bezüge zu anderen Informationen ist ein zweiter Monitor vorgesehen (siehe Abbildung 1), der direkt den Artikel, auf den Bezug genommen wird, darstellt. Bei mehreren Verweisen kann man wählen, welcher Text angezeigt werden soll. Zum Annotieren können auf einem dritten Monitor (links unten) handschriftliche Notizen auf die Seite geschrieben werden. Diese werden dann abfotografiert und stehen als neues Dokument allen Anwendern zur Verfügung. Durch die Seiten im System kann man Pfade (trails) erstellen und abspeichern und somit Dossiers erstellen, die dann beispielsweise einen Neuling schnell mit einem Thema vertraut machen können. Diese Pfade sind unter den Benutzern austauschbar. [Müller-Prove 2002, w3 Keep et al. 2000, w3 de Bra 2001]

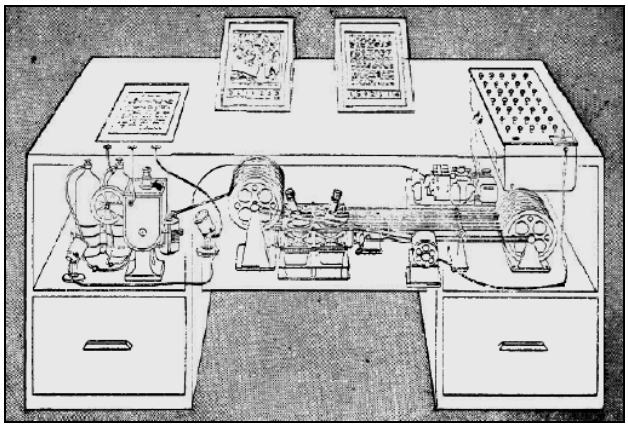

Abbildung 1: Vannevar Bush Memex: Die Zeichnung wurde 1945 nach den Ausführungen Bushs für das LIFE-Magazin angefertigt. Sie zeigt eine mögliche Realisierung des mikrofilmbasierten Memex mit zwei Monitoren für die Darstellung von Hauptdokument und einem Dokument, auf das verwiesen wird. Der Monitor unten links ist für die Annotation. Mit dem Kontrollpult rechts kann die Seite ausgewählt werden.

Das System war damals technisch nicht realisierbar und wurde nie gebaut. Die Idee, wissensverarbeitende Systeme zu erschaffen, wurde aber vielfach wieder aufgegriffen.
Unter anderem in den sechziger Jahren von dem Soziologen Ted Nelson. Nelson gilt als der Erfinder des Hypertexts. In seinem System Xanadu sind die Einheiten feingranularer: Nicht nur ganze Texte sind verlinkt, sondern einzelne Begriffe. Wissen entsteht in unserem Gehirn durch synaptische Verknüpfungen und Bildung von Sinneinheiten. Analog dazu sollen auch die Einheiten in Dokumenten explizit mit Links verknüpfbar und zu neuen Dokumenten kombinierbar sein. Xanadu ist dabei nicht nur zum nachträglichen Verlinken, sondern

[4] 1890 Hollerithmaschine zum Auszählen der amerikanischen Volkszählung mithilfe von Lochkarten

auch als Werkzeug zum Erstellen von nicht linearen Texten gedacht. Durch die Integration des Verlinkens in den Erstellungsprozess sollten neue Informationsquellen entstehen, die gerade nicht durch die unnatürliche Linearität eingeschränkt sind. Alle Links sind bidirektional angelegt, was auch nahe liegt: Steht ein Begriff zu einem anderen in Beziehung, besteht diese Relation auch umgekehrt. Tote Links gibt es in Xanadu nicht, denn alle Versionen eines Dokuments werden gespeichert und somit verschwindet die verlinkte Information niemals. Die einzelnen Einheiten in einem Text werden vom Dokument getrennt abgelegt. Das heißt, eine Sinneinheit kann in verschiedenen Dokumenten „eingelinkt" werden und sobald ihr Inhalt an einer Stelle verändert wird, ist er überall aktualisiert.

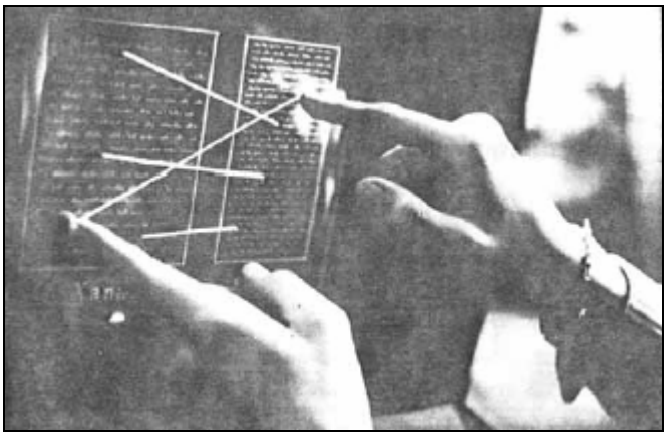

Abbildung 2: Ted Nelson Xanadu: Nelson ist der Erfinder des Hypertext. Das Bild zeigt die Verknüpfung von Begriffen mithilfe bidirektionaler Links. Es war Nelson wichtig, dem Benutzer die Links auch visuell aufzuzeigen.

Bushs Memex-Idee, Information leichter und schneller verfügbar zu machen, wurde von Nelson weitergedacht. Er wollte nicht nur die innerhalb einer Abteilung oder Organisation befindlichen Dokument zentral bereitstellen, sondern alle irgendwo auf der Welt entstehenden wissenschaftlichen Abhandlungen in ein Xanadu-Netzwerk einspeisen. Damit war er einer der Schöpfer der Idee des Internet. Durch das vollständige Aufheben der Lokalität von Information macht es keinen Unterschied mehr, wo Wissen entsteht. Dem globalen Fortschritt sind keine geographischen Grenzen mehr gesetzt. Seinen Vorläufer des Internet nannte er die Docuverse. Für Xanadu wurden einige Komponenten als Prototyp entwickelt, eine fertige Version besteht bis heute nicht. [Müller-Prove 2002, w3 Keep et al. 2000, w3 Xanadu, w3 de Bra 2001, Meyrowitz 1989]

1967 traf Nelson mit seinem Collegefreund Andries van Dam zusammen, der mittlerweile Professor an der Brown University in Rhode Island war. Beide beschlossen, ein Hypertext-Autorensystem, das Hypertext Editing System (HES), zu schaffen. In HES werden die Links vom Dokument getrennt abgespeichert. Dadurch ist es einerseits möglich, alle im System vorhandenen Links zu einem Thema zu suchen, andererseits lassen sich so die mit einem Dokument verknüpften Einheiten anzeigen und beim Editieren oder Löschen eines Textteils die bestehenden Assoziationen berücksichtigen. Links in HES sind typisiert, das heißt, dass eine Erklärung beispielsweise einen anderen Typ als ein Verweis auf einen ähnlichen Artikel hat. Somit kann dem Leser visuell angezeigt werden, was sich hinter einem Link verbirgt und er kann, ohne dem Link zu folgen, entscheiden, ob er die Information sehen will. [van Dam 1987, Müller-Prove 2002, w3 de Bra 2001, w3 Hong, w3 Keep et al. 2000]

Nachdem die Sowjetunion 1957 als erste ein Flugobjekt, den Sputnik, in den Weltraum gebracht hatten, wurde vom amerikanischen Verteidigungsministerium die Advanced Research Projects Agency (ARPA) gegründet. Ziel der Organisation war es, den wissenschaftlichen Vorsprung der Ostmacht aufzuholen und diese zu überholen. Einer der ersten Wissenschaftler, die die ARPA finanzierte, war Douglas Engelbart. Ihm war Bushs Artikel

ebenfalls bekannt und auch er wollte ein Werkzeug zum effizienteren wissenschaftlichen Arbeiten erschaffen: „Augment Human Intellect“ , die Erweiterung des menschlichen Intellekts, ist der Titel einer seiner Arbeiten. Engelbart forschte am Stanford Research Institute in Kalifornien, wo er 1968 das Ergebnis der Arbeit seines Teams präsentierte: das oNLine-System (NLS). Bei dieser Präsentation wurde zum ersten Mal ein Computersystem live vorgeführt. Für NLS wurde die Maus erfunden. Als weitere Eingabegeräte hatte der Benutzer eine Tastatur und ein fünftastiges Keyset, das je nach Kombination der Tasten andere Funktionen auslöste (siehe Abbildung 3). NLS war das erste System, das es vorsah, jeden Arbeitsplatz mit einem Monitor auszustatten und somit direkte Interaktion zu ermöglichen. Da ausreichend große Displays nicht bezahlbar gewesen wären, löste das Team die Aufgabe, indem nur der Timesharing-Großrechner[5], an den die Klienten angeschlossen waren, einen Monitor bekam. Dieser wurde von einer Fernsehkamera abgefilmt und das Signal dann an die einzelnen Terminals übertragen. Dort kam es auf einem Fernseher wieder zur Darstellung.
Auf Softwareseite wurde dem versammelten Fachpublikum eine vollständig funktionierende Implementierung von Links präsentiert sowie eine systemweite Volltext-Suchfunktion und damit automatischer Zugriff auf alle im System vorhandenen Begriffe. In NLS-Dokumente können Grafiken eingebunden werden, ein Vorläufer der E-Mails ist eingebaut und auch Drag&Drop hat das Team erfunden. Sogar eine Video-Konferenz mit Shared Desktop[6] wurde durchgeführt.
Die Präsentation gab den Anwesenden einen Ausblick, welch immense Möglichkeiten in dem neuen Medium stecken. [Engelbart 1968, w3 Engelbart, Müller-Prove 2002, w3 Santa Barbara 2003, w3 Gasch 1996, Meyrowitz 1989, van Dam 1987, Barnes 2001, w3 de Bra 2001, w3 Keep et al. 2000]

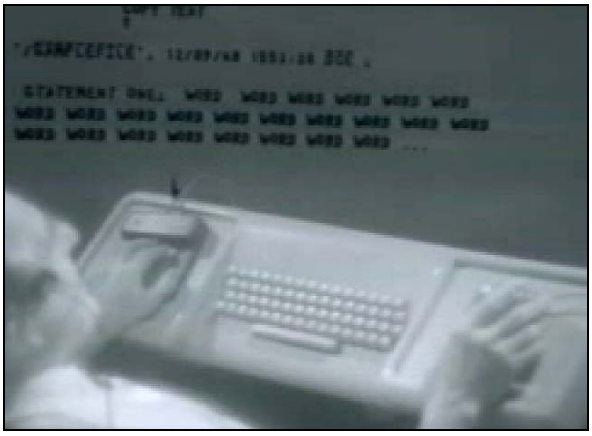

Abbildung 3: Doug Engelbart NLS: Der Screenshot zeigt, das Bild, das die Besucher der Präsentation auf einem Monitor zu sehen bekamen. Im Vordergrund sitzt Doug Engelbart, der mit dem Keyset (links), der Tastatur und der Maus das NLS bedient. Über das Bild ist die Bildschirmausgabe von Engelbarts Monitor geblendet.

Unter den Besuchern war Andries van Dam. Inspiriert von der Fülle an guten Ideen beschloss er, die Entwicklung von HES einzustellen und stattdessen mit seinen Studenten ein neues System zu entwickeln: das File Retrieval and Editing System (FRESS). FRESS sollte die Vorteile von Xanadu, HES und NLS vereinen. Zu den Funktionen von HES kamen vor allem die Volltext-Suchfunktion und die Möglichkeit große Dokumente ohne Verzögerung zu bearbeiten. Die vielleicht wichtigste Neuerung war aber die Undo-Funktion. Sie erlaubt es, einzelne Schritte rückgängig zu machen und somit gefahrlos auszuprobieren, was bei

[5] Timesharing-Großrechner: Es gibt einen Server, an den mehrere Clients/ Terminals angeschlossen sind. Die Clients dienen nur der Darstellung und der Interaktion mit dem Benutzer, die komplette Berechnungsleistung wird vom Server erbracht.
[6] Shared Desktop: Mehrere Teilnehmer (einer Videokonferenz) können denselben Bildschirminhalt sehen und bearbeiten. Bei Engelbart wurde dies hardwareseitig realisiert: Nur der Server hatte einen Bildschirm und deshalb war das Bild an allen Clients schon einheitlich. Die Konferenzteilnehmer wurden analog mit einem Videomischer dazu eingeblendet.

bestimmten Aktionen passiert. Der Benutzer fühlt sich mit diesem Auffangnetz im System sicherer und kann dessen Funktionen durch Experimentieren schneller erlernen.
FRESS wurde unter anderem in verschiedenen geisteswissenschaftlichen Kursen an der Brown-University eingesetzt. Van Dam berichtet von einem Kurs, der darin bestand, dass alle Teilnehmer sich eine bestimmte Zeit pro Woche auf dem Rechner einloggen mussten und dort Texte bearbeiteten. Die dabei entstehenden Kommentare und Lösungen wurden dann in der nächsten Woche allen Anderen zur Verfügung gestellt. Anschließend überarbeitete jeder seine Lösung noch einmal. Das Experiment zeigte, dass sonst eher stille Schüler plötzlich großes Potential entfalteten. Manche sonst rege Mitarbeitende dagegen schrieben relativ wenig. [van Dam 1987, Müller-Prove 2002, w3 Keep et al. 2000, w3 de Bra 2001]

Auf dem Gebiet der Hypertext Authoring- und Editingsysteme entstanden in der Folge noch einige Lösungen, wie NoteCards, Hyperties, Guide, HyperCard oder Storyspace, um nur einige zu nennen. Gravierende neue Konzepte in Bezug auf Lernumgebungen brachten diese jedoch nicht. [Müller-Prove 2002, w3 Keep et al. 2000, w3 de Bra 2001]

Erst 1985 wurde ebenfalls an der Brown University das Intermedia-System entwickelt. Federführend dabei war ein ehemaliger Student van Dams: Norm Meyrowitz[7]. Dieser war Leiter des neu gegründeten Institute for Research in Information and Scholarship (IRIS). Als direkter Aufsatz auf ein Unix-Betriebssystem (A/ UX) für den Macintosh bietet Intermedia sechs Anwendungen, die vollständig in das System integriert sind (siehe Abbildung 4). Dadurch ist es möglich, beliebige Objekte zu einem Dokument zu kombinieren und diese dann vom Standard-Interface aus zu editieren. Man muss zum Bearbeiten eines Bildes nicht extra in Draw wechseln, sondern bekommt die benötigte Funktionalität in der Standard-Dokumentansicht angezeigt, was das Arbeiten beschleunigt und einfacher macht. Ein Grundbestandteil von Intermedia ist das Linken. Dazu enthält jede Anwendung das Intermedia-Menu, in dem man Anfangs- und Endpunkte eines Links setzen kann. Beispielsweise kann ein Textteil mit einem Video verknüpft werden oder ein Bild mit einer Anwendung. Es sind fast beliebige Assoziationen möglich. Die Links werden wie im HES dokumentenextern gespeichert, wodurch sich unter anderem einfach eine Linkstruktur zu einem Dokument generieren lässt. Trotz der guten Ansätze fand Intermedia kaum Verbreitung, weil das darunterliegende A/ UX für Apple nur auf sehr wenigen Rechnern installiert wurde. [w3 Landow, Müller-Prove 2002, w3 Keep et al. 2000, w3 de Bra 2001]

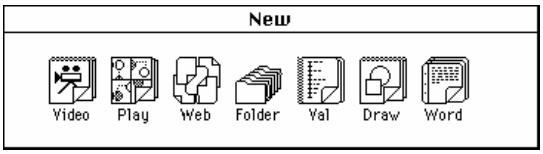

Abbildung 4: Norm Meyrowitz Intermedia: Die Abbildung zeigt das Menu mit dem neue Dokumente erstellt werden.

Geistiger Vater der graphischen Benutzeroberfläche (GUI), wie sie auch bei Intermedia und bei nahezu allen heutigen Anwendungen eingesetzt wird, ist Alan Kay.
Kay studierte bis 1966 in Colorado Mathematik und Molekularbiologie. Anschließend wurde auch er (wie Engelbart) von der ARPA eingestellt und zwar in Utah, wo ihm 1969 der Doktortitel für den theoretischen Entwurf eines graphischen objektorientierten Computers (Reactive Engine; Flex-machine) verliehen wurde. Wie in der Biologie aus kleinen eigenständigen Zellen ein viel mächtigerer Organismus entsteht, postulierte der Biologe Kay, so müsste es auch möglich sein, dass aus vielen autonomen Modulen ein Computersystem entsteht. Genau wie die Zellen im Körper sollten auch die Module in der Lage sein, sich je nach Problem neu zu formieren und sich so optimal anzupassen. Ein gutes Konzept dazu war das neu entstandene Paradigma der Objektorientiertheit, das 1967 mit der von Kristen Nygaard und Ole-Johan Dahl entwickelten Programmiersprache Simula eingeführt worden war. Kays Doktorarbeit beinhaltet eine erste Implementierung solch eines modula-

[7] Meyrowitz ist auch der Vater von Shockwave und langjähriger Leiter von Macromedia

ren Systems. Neben Simula war Kay während des Studiums mit Ivan Sutherlands Sketchpad in Berührung gekommen. Sketchpad wurde 1962 vorgestellt und war das erste direktmanipulative System. Es erlaubte, mit einem Lichtgriffel geometrische Formen auf einen Monitor zu zeichnen.

Abbildung 5: Ivan Sutherland Sketchpad: Die Abbildung zeigt Ivan Sutherland. In der rechten Hand hält er den Lichtgriffel, mit dem er direkt den Inhalt des Bildschirms modifizieren kann. Mit der linken Hand bedient er eine Schalttafel, an der er z.B. die Form, die er zeichnen möchte (Quadrat, ...), auswählen kann.

Im Jahre 1968 wurde Kay im Rahmen seiner Tätigkeit bei der ARPA ein System zur Handschriftenerkennung vorgestellt. Außerdem besuchte er in diesem Jahr den Psychologen Papert. Papert war ein Schüler Piagets, der wiederum zusammen mit Bruner die Lerntheorie des Konstruktivismus begründet hatte (näheres hierzu in einem anderen Vortrag des Seminars: Pamela Grassl, Klassifikation Interaktiver Lehr-/ Lernsysteme, Wilhelm-Schickard-Institut, Universität Tübingen, Juli 2003). Papert vertrat die Ansicht, dass das effektivste Lernen das Lernen durch Ausprobieren sei. Er führte nach dieser Maxime erste Versuche durch, Kindern mithilfe der Programmiersprache Logo am Computers etwas beizubringen. Kay war sehr beeindruckt und formulierte „doing with images makes symbols“, im Sinne von „mit Bildern lernen schafft Verständnis“, was einen der Grundgedanken des Konstruktivismus beinhaltet: Wissen kann in drei Formen repräsentiert werden, durch Handlung, Bild und Symbol bzw. in Kays Realisierung dann durch Maus, Icon und Programmiersprache. Schließlich nahm auch Kay an der Demonstration Engelbarts teil. Beflügelt von all diesen Eindrücken kam er zu dem Schluss, dass jeder Mensch einen Computer haben sollte, der klein ist, ihn nicht stört und ihm bei allen nur erdenklichen Aufgaben unterstützend zur Verfügung steht: die Idee des Dynabook war geboren.

Abbildung 6: Alan Kay Dynabook: Die Abbildung zeigt eine Skizze, wie Kay sich sein Dynabook vorstellen konnte, als kleinen flachen Rechner, der jederzeit und überall hin mitgenommen werden kann.

Es sollte ein tragbarer Computer mit Flachbildschirm und Tastatur sein, bei dem man auch direkt auf den Bildschirm schreiben kann und der in jeder Situation von Nutzen ist. Ein Architekt sollte beispielsweise seine statischen Berechnungen auf dem Dynabook machen können und anschließend durch das dreidimensionale Modell des Gebäudes wandern. Dabei sollte das Dynabook ein Alltagsgebrauchsgegenstand wie eine Uhr werden. Es sollte keinen festen Verwendungszweck haben, sondern gemäß seines Postulats flexibel gestaltet und somit für jeden Zweck anpassbar sein. Dazu sollte es aber nicht nötig sein, sich tiefgehend mit Programmierung zu befassen, vielmehr sollte es zum Beispiel für Kinder ohne weiteres möglich sein, sich Spiele selbst zu programmieren und diese dann mit Freunden zu tauschen und gemeinsam zu spielen.

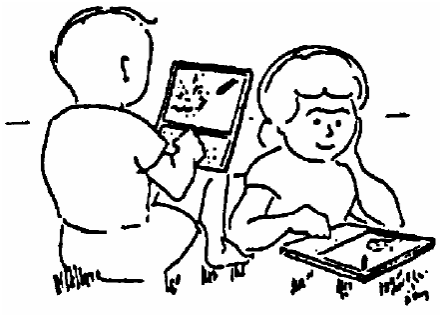

Abbildung 7: Kinder bei der Benutzung des Dynabook: Das Bild soll die kinderleichte Bedienbarkeit des Dynabook zeigen. Die Kinder sind am spielen und zwar mit ihren selbst geschriebenen Programmen.

Mit seinen Ideen kam Kay 1972 zum Xerox Palo Alto Research Center (PARC). Die Firma Xerox verdiente gut mit ihrem Patent auf das Trockenkopierverfahren (Xerographie) und wollte den technischen Fortschritt nicht verpassen. Deshalb wurde PARC gegründet (Kay: “the best way to predict the future is to invent it“). Bei Xerox wurden Kays Ideen weiterentwickelt. Auf Hardwareseite schuf man so genannte Interims-Dynabooks, die Teilaspekte des Dynabooks umsetzten. Eines dieser Interims-Dynabooks war der erste Desktoprechner: der Alto. Auf Softwareseite wurde die von Kay schon vorher begonnene Smalltalk-Umgebung weiterentwickelt, das Betriebssystem für das Dynabook. Smalltalk ist modular aus vielen Objekten aufgebaut. Der Benutzer kann sehr einfach neue Objekte erstellen oder schon vorhandene modifizieren, weil alles im System in der Sprache Smalltalk selbst programmiert ist. Das ist von großem Vorteil, wenn man Dinge benötigt, die im Auslieferungszustand vielleicht nicht vorgesehen waren. Smalltalk arbeitet, motiviert durch Kays Erfahrungen bei Papert, intensiv mit Symbolen. Es war die erste Oberfläche mit Fenstern, Symbolen, Menus und Maus (WIMP: windows, icons, menus, pointing device). Damit war es nicht mehr notwendig, sich Textkommandos zu merken. Viel mehr wird das System durch Klicken auf Symbole gesteuert, was aus psychologischer Sicht viel natürlicher und einprägsamer ist. Die Erfindung von Fenstern erlaubte es erstmals, auf einem Bildschirm mehrere Inhalte gleichzeitig zu präsentieren und damit auch parallel an Dingen zu arbeiten. Es war mit Smalltalk erstmals möglich, an einem einzigen Rechner beispielsweise zu recherchieren und in einem weiteren Fenster seinen Artikel weiter zu schreiben. Das System wurde auch für Kinder entwickelt und natürlich an diesen getestet. Dazu wurden regelmäßig Schulklassen in das Labor eingeladen. Die Kinder waren schnell in der Lage, eigene Programme zu schreiben. Mit ein wenig Unterstützung durch Kays Team entstanden Malprogramme, Spiele und ähnliches. Die Kinder, die schneller mit Smalltalk zurecht kamen begannen den Anderen zu helfen und die Schüler fingen an, in Teams größere Programme zu schreiben. Durch das Modulkonzept konnte jeder die Arbeit des Anderen direkt verwenden und darauf aufbauen. Das Projekt war ein voller Erfolg.
Vor seinem Besuch bei Papert hatte Kay die Ansicht vertreten, wer die Welt verstehen will, muss programmieren lernen. Nachdem er die Kinder experimentieren gesehen hatte, war er davon überzeugt, dass nicht der Mensch zum Computer kommen muss, sondern umgekehrt. Dementsprechend waren die Ziele für Smalltalk und das Dynabook formuliert: beide soll-

ten kinderleicht zu erlernen und intuitiv bedienbar sein. Zum Einsatz in großem Stil kam Smalltalk trotz der Erfolge nicht, wohl weil die Hardware zu teuer und zu viele Änderungen am Schulsystem von Nöten gewesen wären. [Barnes 2001, Kay 1977, Müller-Prove 2002, w3 Santa Barbara 2003, w3 Gasch 1996, w3 MIT 1995]
Die Smalltalk-Entwicklung besteht in einer webbasierten Version als Squeak [www.squeak.org] bis heute fort. Der komponentenbasierte Ansatz findet beispielsweise in Form von Microsoft.NET gerade wieder Einzug in die Betriebssysteme.

Da Xerox aus seinem Labor wirtschaftlichen Nutzen ziehen wollte, wurden bis 1981 die entstandenen Lösungen zu einem kommerziellen Betriebssystem, dem Xerox-Star, zusammengebaut. Star ist der direkte Vorläufer der heutigen WIMP-Systeme und zudem das erste kommerzielle System, das direkt manipulierbar war. Man kann Text mithilfe eines Cursors (im PARC erfunden) markieren, kopieren, ausschneiden (cut&paste) und verschieben, und vor allem Befehle durch Klicken auf Icons und Menuelemente auslösen (davor waren Betriebssysteme kommandozeilenorientiert). Menus mit ihrem Aufbau in Nomen als Oberkategorie und Verb als Befehl, also „Datei“ -> „öffnen“, waren ebenfalls neu und gleich konsistent implementiert, denn alle Star-Anwendungen haben mehr oder weniger dasselbe Menu („orthogonal set“). In Star wurde außerdem der dokumentenzentrierte Ansatz von Smalltalk verfolgt: Zentraler Punkt des Systems ist nicht die Anwendung, sondern viel mehr der Anwender und dessen Problem, das es zu lösen gilt („Sie wollen einen Brief schreiben“). Konsequenter Weise gibt es daher das Prinzip der „progressive disclosure“, des Ausblendens von gerade nicht benötigten Befehlen. Beim Schreiben eines Textes wird beispielsweise die Palette mit verschiedenen Pinselarten ausgeblendet und vielleicht eine für Textformate eingeblendet. Star war auch das erste System, das die Desktop-Metapher beinhaltete. Alle Operationen sind vom virtuellen Schreibtisch aus erreichbar. Zum Beispiel kann man die Dateistruktur durchsuchen, ohne genaue Kenntnis des Dateibaums haben zu müssen. Es gibt Ordner und Dateien (Akten; files), durch die man navigieren kann. Auch das uns heute vertraute What You See Is What You Get (WYSIWYG), der Ansatz, auf dem Bildschirm alles genau so darzustellen, wie es später gedruckt wird, wurde in Star erstmals vom Betriebssystem bereitgestellt. [Meyrowitz 1989, Müller-Prove 2002]
Da Xerox es geschafft hat, zwar fast alle der bahnbrechenden Entwicklungen zu tätigen (z.B. Laserdrucker, Ethernet, Client/ Server-Modell, ...), aber keine davon auf dem Markt platzieren konnte, wurde auch Star kein großer Erfolg. Das System wurde aber von den Apple-Begründern (Jobs, Wozniak, ...) besichtigt, die alle Features in ihr erstes Apple-System Lisa einbauten, dessen Nachfahren dann Mac-OS und Windows wurden. Wie auffällt, hat sich bis zur letzten Generation von Oberflächen (Windows ME, Mac OS 9) nicht allzu viel geändert. Die Systeme sind auf einzelne Benutzer an einem Rechner zugeschnitten und nicht auf Gruppenarbeit in einem Netzwerk, wie sie heute viel häufiger anzutreffen ist. Die Konzepte zur Bedienung des Systems sind starr und können nicht flexibel an die benötigten Situationen angepasst werden. Es gibt unendlich viele proprietäre Dateiformate für dieselben Arten von Daten (z.B. Vektorgrafik), die nur mit den zugehörigen Programmen bearbeitet werden können. Kurz: Es gibt viele Inseln.

Wunschdenken

Genau diese Punkte nahm die IRIS-Forschergruppe unter Norm Meyrowitz schon 1989 zum Ausgangspunkt, um in sieben Kategorien Verbesserungsvorschläge für ein besseres System zu machen [Meyrowitz 1989]. Die Kategorien sind: Integration, Ästhetik, Perspektive, Zugang, Systemdienste, Gruppenarbeit und Anpassbarkeit. Ich werde die Punkte im Folgenden ausgehend vom Artikel mit aktuellen Beispielen erläutern. Wie wir sehen werden, ist das gemeinsame Ziel aller Punkte Vereinheitlichung und damit die Schaffung von Standards.

Integration

Der erste Punkt ist die Integration. Integriert werden sollen sowohl Daten, als auch Anwendungen. Alle Arten von Daten sollen in einem Dokument einbettbar sein. Die einzelnen Informationseinheiten sollen wie in Xanadu oder im Sinne von Kay Objekte sein, die im Zusammenspiel das Dokument ergeben. Wie in Xanadu sind Informationen bei Aktualisierung an einer Stelle dann überall, wo sie vorkommen, geändert. Hauptvorteil dabei ist die Konsistenz von Daten, die mehrfach verwendet werden. (Ted Nelson bezeichnete diese Art von Einbettung in Xanadu als „Transclusion“.) Die Einheiten im Dokument sollen Daten über Ports austauschen können, eine Animation soll sich also beispielsweise durch Ändern der darunter in einer Tabelle stehenden Werte manipulieren lassen. Schließlich sind auch Links vorgesehen. Diese sollen wie in HES typisiert (Annotationstyp, Verweistyp, ...) und bidirektional sein. Meyrowitz unterscheidet zwischen statischen (z.B. Link auf einen anderen Text) und dynamischen Links. Die dynamischen Links gliedern sich noch einmal in Warm Links und Hot Links auf. Bei Warm Links werden beim Verfolgen des Linkpfades Daten übermittelt. Der Submit-Knopf eines Eingabeformulars ist ein Vertreter von Warm Links. Hot Links müssen nicht erst angeklickt werden, sondern tauschen permanent Daten mit ihrem Zielobjekt aus. Mein Beispiel mit der Animation und der Konfigurationstabelle könnte über einen Hot Link realisiert sein. Stellt man die Geschwindigkeit höher, fährt das Auto in der Animation sofort schneller. Die Ziele für die Links sollen wie in Smalltalk flexibel sein: Man soll genauso auf ein Wort verweisen können wie auf ganze Tabellenteile oder auf Anwendungen. Verlinken soll wie in Intermedia fester Bestandteil des Betriebssystems (und damit jeder Anwendung) sein. Die Links sollen wie in HES dokumentenextern gespeichert werden, womit sich beispielsweise die Problematik behandeln lässt, was mit Referenzen passiert, wenn ein Dokument gelöscht wird, weil man die Zeiger auf das Dokument leicht finden kann. Außerdem können extern gespeicherte Links einfacher mit Rechten versehen und bestimmte Links nur bestimmten Nutzern erlaubt und auch nur ihnen grafisch sichtbar gemacht werden. Eine Möglichkeit der technischen Realisierung wird auch vorgeschlagen: ein Dateisystem auf Basis einer objektrelationalen Datenbank, in der dann alles als Objekt repräsentiert ist. Diese Datenbank besitzt für jedes Systemobjekt einen Typ, also beispielsweise einen für Vektorgrafik und einen für Tabellen. Dieser Typ wird von allen Anwendungen benutzt. Durch die einheitliche Repräsentation von Daten gleichen Typs ist es problemlos möglich, beispielsweise die Vektorgrafik mit verschiedenen Anwendungen (von verschiedenen Herstellern) zu bearbeiten. Bei mehreren Programmen, die ähnliche Funktionen bieten, kann ich dasjenige wählen, das mir am besten passt ohne die Kompatibilität zu anderen Anwendern, die andere Programme einsetzen, zu verlieren. Ebenfalls umzusetzen ist der dokumentenzentrierte Ansatz von Intermedia oder Star, der eine Oberfläche bietet, in der sich konsistent die Anwendungen bei Bedarf einlinken. Angenommen, ich schreibe einen Text und will ein Video einbauen. Dann muss ich nicht in die Videobearbeitung wechseln sondern bekomme in meine bekannte Umgebung neue Funktionalität für Videobearbeitung eingelinkt.

Ästhetik

Unter Ästhetik verstehen die Autoren das Look&Feel des Systems, das Aussehen und die Bedienbarkeit. Die Benutzeroberfläche soll den aktuellen technischen Standards entsprechen. So wäre es heute ohne Weiteres möglich, viel höher auflösende Monitore zu bauen und damit ein deutlich schärferes, besser lesbares, Bild zu liefern. Das System soll in den Hintergrund treten. So wird bemängelt, dass sich die Benutzeroberfläche viel zu sehr aufdrängt, anstatt das zu sein, was sie sollte: ein nützlicher Diener im Hintergrund, der uns nicht von der eigentlichen Arbeit ablenkt oder gar durch kryptische Bedienbarkeit dabei behindert. Dazu soll die Bedienung überall konsistent erfolgen. Ansatzweise realisiert ist das heute damit, dass „Datei“ immer der erste Menupunkt ist, gefolgt von „Bearbeiten“. Zu Konsistenz gehört auch, dass sich die Unterpunkte immer im selben Menu wieder finden: „Ausschneiden“ sollte nicht plötzlich in „Format“ anstelle von „Bearbeiten“ stehen. Das System soll also auch verlässlich sein. Wenn einem Icon eine Bedeutung zugewiesen wurde, muss diese auch konsistent beibehalten werden: Das Symbol für Öffnen darf nicht in der nächsten Anwendung oder gar im selben Programm plötzlich Speichern bedeuten. Der Benutzer kann überall das erlernte Verhalten einer Funktion erwarten, auf das er vertraut. Dazu gehört auch, dass ich mit dem Papierkorb durch Drag&Drop nicht nur Dateien sondern genauso einzelne Worte löschen kann. Schließlich muss dem Anwender auch visuell klargemacht werden, was im System gerade geschieht. Der Papierkorb als Symbol für einen Ort, an dem Objekt landen, die dann beim Leeren gelöscht sind, ist hier wieder gutes Beispiel oder die Visualisierung von Kopieren, bei der ein Dokument von Ordner zu Ordner fliegt.

Perspektive

Unter Perspektive fällt erst einmal der Blickwinkel auf die Daten. Inhalt und Darstellung sollen unabhängig voneinander gespeichert werden. So soll es ohne weiteres möglich sein, eine Tortengrafik in ein Säulendiagramm zu „verwandeln“ und ähnliches. Dazu müssen die Objektformate in der oben angesprochenen objektrelationalen Datenbank, die alle Komponenten im System verwaltet, repräsentationsunabhängig sein. Für die Präsentation wird dann einer von unzähligen, schon vordefinierten Stylesheets über das Dokument gelegt. Die Abstraktion soll dabei so weit gehen, dass durch einfaches Ändern der Präsentationsschicht aus einem technischen Artikel ein allgemein verständlicher Zeitungsartikel wird. Damit erstreckt sie sich sogar auf die Semantik. Die Vorlagen sollen professionell sein und es jedem auch noch so gestalterisch Unbegabten ermöglichen, ansprechende und vor allem dem gewünschten Ziel dienliche Dokumente zu erstellen. Eine Vorlage könnte beispielsweise für einen „Vortrag in einer Halle vor 1000 Leuten“ existieren. Aber nicht nur für Dokumente, sondern auch für die Bedienoberfläche selbst soll es Stylesheets geben, die den Desktop an verschiedene Arbeitssituationen anpassen. In einer Bücherei wird man andere Anforderungen an das System haben, als in einem Trickfilmtudio, aber auch eine einzige Person wird bei der Recherche zum Beispiel andere Elemente auf dem Desktop benötigen, als beim Schreiben eines Artikels. Die Vorlagen sollen dabei so gestaltet sein, dass sie ohne tiefgehende Kenntnis der Struktur geändert werden können. Das Ändern des Schriftgrads beispielsweise soll ohne Studieren des Stylesheets oder des Handbuchs möglich sein. Ebenfalls sollen einfach neue Vorlagen erstellt werden können. Wie auf Dokumente und Desktop sollen auch verschiedenste Ansichten für andere Objekte im System bereitstehen. Beispielsweise um die Verlinkung eines Dokuments zu visualisieren, Dateien eines bestimmten Autors zu identifizieren, Bilder zu durchstöbern...

Zugang

Mit Zugang ist hauptsächlich der Zugriff auf das gespeicherte Wissen gemeint. Vor dem Benutzer soll nichts im System verborgen bleiben. Jedes Objekt soll ihm auf viele Arten dargeboten werden. Wie schon bei Integration angesprochen kann im idealen System nahezu alles verlinkt werden. Mit diesem Mechanismus soll sich der Anwender seine eigenen Bookmarks auf die Dinge im System machen können, die er benötigt. Das System soll

automatisch Buch führen über die verwendeten Objekte, so dass der Benutzer, der weiß, dass er eine bestimmte Sache vor drei Tagen vor der Mittagspause gemacht hat, keine Mühe hat, diese Sache wieder zu finden. Alle Objekte im System sollen über Volltextsuche auffindbar sein. Dazu soll das System schon beim Erstellen der Objekte Indexeinträge anlegen, die auch Text im Bild und andere heute noch nicht durchsuchbare Dinge enthalten. Neben der höheren Trefferzahl durch sonst nicht auffindbares Wissen wird die Suche durch den Index deutlich beschleunigt, was ihre Benutzung erst ermöglicht. Wenn es nicht mehrere Minuten dauern würde, einen gesamten Festplattenteil zu durchsuchen würden sicherlich mehr Menschen die Suche bei der täglichen Arbeit als Standardhilfsmittel verwenden. Man soll nach Kategorien wie Schlüsselworten oder anderen, auch selbst vergebenen Attributen suchen können. Neben dieser gesteuerten Suche sollen im System auch Agenten mitlaufen, die sich den Gewohnheiten des Benutzers anpassen und Informationen liefern, die ihrer Meinung von Interesse sein könnten. Suche ich beispielsweise nach einem bestimmten Reiseziel im Internet, könnte mir der Agent Ausflugsziele und ähnliches in der Umgebung zusätzlich präsentieren. Ein interessanter Vorschlag ist auch, die Suche mit dem Thesaurus zu verbinden: Suche ich nach Zimmer, so wird die Suche automatisch auf Raum, Bude, Stube und ähnliches ausgeweitet und liefert mir so natürlich viel mehr passende Ergebnisse. Diese Ergebnisse und auch die Suchanfrage selbst sollen speicherbar und damit auch für andere Benutzer verfügbar sein. Schließlich soll es für den Benutzer keinerlei Unterschied machen, auf welche Art von Daten er gerade zugreift oder wo diese liegen (z.B. lokal oder im Netz). Alles soll sich einheitlich präsentieren.

Systemdienste

Systemdienste sind grundlegende Funktionalitäten, die bereitgestellt werden müssen. Dazu zählen Hilfsmittel wie Wörterbuch, Thesaurus, Rechtschreibhilfe, Kalender und ähnliches. Wichtig ist, dass diese nicht von der Anwendung sondern vom System bereitgestellt werden und damit in jeder Anwendung zur Verfügung stehen. Somit muss das Rad nicht jedes Mal neu erfunden werden. Die Schnittstelle für solche Dienste muss offen und standardisiert sein, so dass jeder Anbieter einen Systemdienst bereitstellen kann. Während Benutzer A die Rechtschreibkorrektur von Firma X benutzt, kann Benutzer B die von Firma Y einsetzen, weil er diese schöner findet. Genauso kann die Softwarefirma Z, die eine neue grundlegende Funktionalität benötigt, diese als Systemdienst erstellen und somit allen anderen Anwendungen – verbreitet beispielsweise über das Internet – auch zur Verfügung stellen. Schließlich sollen auch grundlegende Dinge, wie Selektoren, Iteratoren, Manipulatoren, ein Linkservice, Suchfunktionalität und ähnliches natürlich für alle Anwendungen gleichermaßen bereit stehen. Wichtiges hier zu nennendes Feature ist auch das Scripting: Im System soll es nicht viele verschiedene Scriptsprachen zum Anpassen von Dingen oder Lösen von Aufgaben geben sondern eine einzige einfach zu erlernende und mächtige Sprache, derer sich alle Anwendungen bedienen.

Gruppenarbeit

Ein gutes Betriebssystem soll alle Funktionalität zum gemeinsam Arbeiten schon mitbringen. Beispielsweise soll es nicht viele verschiedene Chat-Tools geben, von denen jedes eine andere Sprache spricht, sondern einen Standard, so dass wirklich jeder mit jedem kommunizieren kann. Genauso sollen Videokonferenz, Whiteboard, gemeinsames Bearbeiten von Dateien mit Versionsmanagement, Rechtemanagement und alle anderen Bereiche die zur „Computer Supported Cooperative Work“ (CSCW) gehören, eingebaut und vor allem standardisiert sein. Zum verteilten Arbeiten gehört auch, dass Benutzerprofile zentral gespeichert werden und jeder Anwender, egal auf welchem Rechner er sich einloggt, genau seine ihm vertraute Oberfläche wieder findet und dass es keinerlei Unterschied in Bedienung und Aussehen macht, ob er nun gemeinsam mit anderen oder alleine an einer Aufgabe arbeitet.

Anpassbarkeit

Wie wir jetzt schon mehrfach gesehen haben, soll alles im System anpassbar sein. Wichtigstes Mittel dazu für den Endanwender sind Stylesheets und Scriptsprache. Zur Interoperabilität von Anwendungen sollen alle APIs[8] öffentlich sein. Wie bei den Agenten zur Suche sollen im System auch Agenten laufen, die das System dem Benutzer anpassen. Eine (wenn auch vielleicht nicht so gelungene) Umsetzung davon sind die sich ausblendenden Menuelemente in Microsoft Office, die sich meinem Nutzungsverhalten anpassen oder der Assistent, der mir nach mehrmaligem Schließen das Angebot macht, nicht mehr automatisch zu erscheinen.

Fazit für die Lehre mit Neuen Medien

Wenn all die genannten Dinge heute umgesetzt wären, sähe unser Beispiel vom Anfang anders aus. Genauer, es würde gar nicht existieren, weil der Inhalt des Kurses nicht mehr Excel sein könnte, sondern höchstens noch das Erstellen von Tabellen – nicht Anwendungen stünden im Vordergrund, sondern Funktionalität. Mit der Integration von Anwendungen und dem dokumentenzentrierten Ansatz gäbe es nur noch eine Schnittstelle zum Benutzer. Das Umlernen von Anwendung zu Anwendung würde wegfallen, da die Bedienung der Funktionen standardisiert wäre und man sich folglich nicht mehr auf die Suche nach Features machen müsste. Der Großteil des Funktionsumfangs würde wahrscheinlich in die Systemdienste, also unter die Anwendungsschicht, verlagert und damit vereinheitlicht. Viele Schulungsmaßnahmen würden unnötig, weil die dabei zugrunde liegenden anwendungsspezifischen Probleme wegfielen. Das Erstellen von ansprechenden Dokumenten oder Spezialvisualisierungen müsste man nicht mehr lernen, da es im Idealfall schon Standardvorlagen (Stylesheets) gäbe. Man könnte sich also bei der Lehre voll und ganz auf die wissenschaftlichen Inhalte konzentrieren und müsste nicht wie heute sehr viel Zeit in das Erlernen und „Bändigen“ von Software stecken, die eigentlich die bestmögliche Unterstützung bei der Arbeit zum Ziel haben sollte. Ein System, das die Vorschläge umsetzt, würde solch eine Unterstützung wirklich bieten. Zum Lehren und Lernen könnten wir auf alles im System vorhandene zugreifen (auch auf die 3D-Welt aus der Einleitung). Aber auch Möglichkeiten, die heute aufgrund fehlender Standardisierung ein Nischendasein beim Lernen führen, wie der Chat und gemeinsames Lernen in einem Netzwerk, wie dem Internet, würden sicher verstärkt eingesetzt.

[8] API = application programming interface = Schnittstelle, auf der Programme miteinander kommunizieren

Schlusswort

Was ist das Ziel von Software? Wenn man sich heute den Markt anschaut, kann man zu dem Schluss kommen, dass es einzig und allein Gewinnmaximierung ist. Es werden zumeist gerade so viele Verbesserungen an bestehender Software vogenommen, wie nötig sind, um besser verkaufen zu können, als der Konkurrent. Die Kompatibilität von Anwendungen wird auf ein Mindestmaß beschränkt, damit man bei einem Hersteller bleiben muss, um möglichst komfortabel arbeiten zu können.
Da wir einen Teil der geschichtlichen Entwicklung gesehen haben, wissen wir spätestens jetzt, was das eigentliche Ziel von Software war und sein sollte: uns so weit wie möglich in allen Bereichen vom alltäglichen Leben bis hin zu Wissenschaft und Forschung zu unterstützen. Wie ich im Fazit dargelegt habe, sind die momentan bestehenden Daten- und Programminseln dazu sehr hinderlich. Viel zu viel Zeit und Energie nimmt die Anwendung und nicht das Lösen des eigentlichen Problems in Anspruch.
Der Artikel von Meyrowitz, dessen Vorschläge ich präsentiert habe, stammt aus dem Jahre 1989. Er entstand zu einer Zeit, zu der die heute meist genutzten Betriebssysteme Windows und Mac OS noch in den Kinderschuhen steckten (Windows 3.0, Mac OS 6). Damals wäre es noch relativ einfach gewesen, die teilweise grundlegenden Features wie beispielsweise das Linken ins System einzubauen. In anbetracht der Millionen von Programmen, die heute existieren, gestaltet sich dieses Vorhaben nun etwas schwierig.
Microsoft geht mit Microsoft.NET gerade einen Schritt in die richtige Richtung. Zwar nicht als Betriebssystem, aber als Zwischenschicht, die noch unter der Anwendung liegt, bietet .NET die Möglichkeit, Komponenten zu schreiben, die dann von allen .NET-Anwendungen benutzt werden können. Dabei müssen die Komponenten nicht auf dem eigenen Rechner sein, sondern können auch über ein Netzwerk, wie das Internet verwendet werden. Standardfunktionalität kann in solchen Komponenten gekapselt werden und muss dann nicht für jede Anwendung neu „erfunden“ werden. Auch die Vielfalt von Programmiersprachen wird – notgedrungen von der Spitze des Baumes – angepackt: Es gibt für viele der heute verbreiteten Sprachen Compiler in das gemeinsame .NET-Format und damit können die Programme interagieren.
Es bleibt abzuwarten, wann und in wieweit alle Verbesserungsvorschläge umgesetzt werden, oder es sogar noch bessere Ansätze zum effizienteren Lehren, Arbeiten und Forschen gibt...

Quellenangabe

[Hanisch 2003] einige Stunden Gespräche mit Frank Hanisch.

[Barnes 2001] Susan B. Barnes, Alan Kay: Transforming the Computer Into a Communication Medium, Fordham University.

[Meyrowitz 1989] Meyrowitz, Norman, The Desktop of Tomorrow: From User-Centered to Information-Centered Computing, Institute for Research in Information and Scholarship (IRIS), Brown University, July 1989,
http://klynch.com/documents/tomorrow

[Engelbart 1968] Engelbart, D. C., The original 90-minute live public demonstration of the online system NLS, In: Engelbart Collection in Special Collections of Stanford University, December 1968.
http://sloan.stanford.edu/mousesite/1968Demo.html

[Johnson 1989] Johnson, J., Roberts, T. L., Verplank, W., Smith, D. C., Irby, C. H., Beard, M., and Mackey, K., *The Xerox Star: A retrospective. IEEE Computer, 22, 9, 11-29, 1989.* http://www.digibarn.com/stories/desktophistory/bushytree.html (updated Bushy tree by John Redand and Bruce Damer).

[Kay 1977] Kay, A., and Goldberg, A., *Personal Dynamic Media. IEEE Computer, 10(3),* pp. 31- 41, March 1977.

[Meyrowitz 1987] Meyrowitz, N., *The Missing Link: Why We're All Doing Hypertext Wrong.* The society of text: Hypertext, hypermedia and the social construction of information}, pp. 107-114, MIT Press, 1989.

[Müller-Prove 2002] Müller-Prove, M., Vision and Reality of Hypertext and Graphical User Interfaces, *Department of Informatics,* University of Hamburg, February 2002.

[Pahl 2003] Pahl, Marc-Oliver, Mein? Privatleben, Wilhelm-Schickard-Institut, Universität Tübingen, April 2003.

[van Dam 1987] van Dam, A., *Hypertext '87 Keynote Address*, In: Communications of the ACM 31(7), p. 887-895, 1988. http://www.cs.brown.edu/memex

[van Dam 2002] van Dam, A., Next-Generation Educational Software, Brown University and the NSF STC for Graphics and Visualization, EdMedia 2002 Keynote Presentation, June 27, 2002. http://www.cs.brown.edu/people/avd

[Wimmer 1997] Wimmer, Helmut, Zur Konvergenz von Technologie und Denken, Hypertext und Internet, Institut für Softwaretechnik, Technische Universität Wien, Januar 1997.

[w3 Engelbart] Douglas Carl Engelbart, Bootstrap Alliance Website, http://www.bootstrap.org/engelbart/, Stand 5.6.2003.

[w3 de Bra 2001] Prof. Dr. Paul de Bra, Hypermedia, Eindhoven University of Technology, February 2001, hypermedia.ppt.

[w3 Santa Barbara 2003] MAT 200A Winter2003, Dynamic Interaction Through Technology, http://www.mat.ucsb.edu/~g.legrady/academic/courses/03w200a/dynData/, Stand 3.6.2003.

[w3 Keep et al. 2000] Christopher Keep, Tim McLaughlin, Robin Parmar, The Electronic Labyrinth, Institute for Advanced Thechnology in the Humanities, University of Virginia, http://www.iath.virginia.edu/elab/hfl0267.html, Stand 5.9.2003.

[w3 Hong] Hong, Jason, Andy Van Dam, The Creator of Hypertext Systems Developing Four Revolutionary Systems at Brown University, Computer Science Division, University of California, Berkeley, http://www.cs.berkeley.edu/~jasonh/cs39i-seminar/project1/AndyVanDam/, Stand 5.9.2003.

[w3 Landow] Landow, George P., Intermedia: An Introduction, Brown University, University of Singapore, http://www.scholars.nus.edu.sg/landow/cpace/ht/HTatBrown/Intermedia.html, Stand 5.9.2003.

[w3 Gasch 1996] Gasch, Scott, Alan Kay, http://ei.cs.vt.edu/~history/GASCH.KAY.HTML, Stand 14.7.2003.

[w3 Meyrowitz 2002] Meyrowitz, Norman, The Contribute vision, http://www.macromedia.com/devnet/contribute/articles/contribute_vision.html, Stand 23.6.2003.

[w3 MIT 1995] "As We May Think" -- A Celebration of Vannevar Bush's 1945 Vision, An Examination of What Has Been Accomplished, and What Remains to Be Done, Biography Alan Kay, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, October 1995, http://www.eecs.mit.edu/AY95-96/events/bush/ak.html

[w3 van Dam 2002] Van Dam, A., Andries Van Dam, Brown University, http://www.cs.brown.edu/people/avd/, Stand 5.9.2003.

[w3 Xanadu] Project Xanadu, DEEP HYPERTEXT: The Xanadu Model, http://www.xanadu.com/xuTheModel/, Stand 5.9.2003.

Abbildungsverzeichnis