

Bericht 237

**Vision and Reality of
Hypertext and
Graphical User Interfaces**

FBI-HH-B-237/02

Matthias Müller-Prove
mprove@acm.org

In die Reihe der Berichte des Fachbereichs
Informatik aufgenommen durch
Prof. Dr. Horst Oberquelle
Prof. Dr. Christopher Habel

Februar 2002

Abstract

The World Wide Web took off ten years ago. Its tremendous success makes it easy to forget the more than forty years of hypertext development that preceded the Web. Similarly, modern graphical user interfaces have drawn attention away from the many compelling ideas behind earlier user interface designs. In the present thesis, numerous early hypertext and graphical user interface systems are presented and contrasted with today's Web and desktop interfaces. The designers of early hypertext and graphical user interface systems shared a common objective: the development of a personal dynamic medium for creative thought. Not very much is left from this original vision. Retrospect reveals promising insights that might help to reconcile the desktop environment with the Web in order to design a consistent and powerful way to interact with the computer.

Zusammenfassung

Das World Wide Web hat vor nunmehr über zehn Jahren seinen unvergleichlichen Siegeszug begonnen. Dabei wird oft übersehen, daß die Idee des Hypertexts eine bereits über vierzigjährige Geschichte hinter sich hat. Die Arbeit zeigt diese Entwicklung anhand der verschiedenen Hypertextsysteme auf und kontrastiert sie mit dem Web. Die Betrachtung der Grafischen Benutzungsoberflächen zeigt ganz ähnlich, daß auch hier viele gute Ideen auf dem Wege zu den heute dominierenden Fenstersystemen verloren gegangen sind. Von der gemeinsamen Idee von Hypertext und Graphischer Benutzungsschnittstelle, nämlich die Schaffung eines persönlichen Mediums zum kreativen Umgang mit dem Computer, ist heute nur noch wenig übrig. Durch Rückschau gewinnt diese Arbeit mögliche Ansätze, die dazu beitragen können, die Interface-Welten des Webs mit denen der Desktop-Oberfläche zu verbinden.

FÜR LELLE[†]

Betreuer:

Prof. Dr. Horst Oberquelle
Arbeitsbereich Angewandte und Sozialorientierte Informatik (ASI)
Universität Hamburg, Fachbereich Informatik

Prof. Dr. Christopher Habel
Arbeitsbereich Wissens- und Sprachverarbeitung (WSV)
Universität Hamburg, Fachbereich Informatik

Contents

	Preface	vii
1	Introduction	1
2	Hypertext	5
2.1	History	5
2.1.1	Memex	12
2.1.2	Xanadu	14
2.1.3	NLS/Augment	16
2.1.4	HES and FRESS	18
2.1.5	FLEX and Smalltalk	19
2.1.6	NoteCards	21
2.1.7	Symbolics Document Examiner & Concordia	23
2.1.8	Hyperties	26
2.1.9	Guide	26
2.1.10	HyperCard	27
2.1.11	Storyspace	28
2.1.12	Intermedia	29
2.1.13	Microcosm	30
2.1.14	World Wide Web	32
2.1.15	Hyper-G/HyperWave	34
2.2	Theory of Hypertext	38
2.2.1	Hypertext Feature Matrix	38
2.2.2	The Dexter Hypertext Reference Model	40
2.2.3	Open Hypermedia Systems	43
2.3	Provisions for the Future of the World Wide Web	44
2.3.1	Identification of Nodes	44
2.3.2	Groups of Nodes	45
2.3.3	General Hyperlinks	45
2.3.4	Browser	45
2.3.5	Integrated Browser/Editor Environment	46
2.3.6	Separation between Content and Appearance	47
2.3.7	Integration of Hypertext facilities into the Operating System	48
3	Graphical User Interfaces	49
3.1	History	49
3.1.1	Man-Computer Symbiosis	53
3.1.2	Sketchpad	54
3.1.3	NLS/Augment	56
3.1.4	Flex Machine and Dynabook	59
3.1.5	Xerox Alto, the Interim Dynabook and Smalltalk	62
3.1.6	Xerox Star	65

3.1.7	Spatial Data Management System	67
3.1.8	Apple Lisa	70
3.1.9	Apple Macintosh	74
3.2	Human Factors	77
3.2.1	Fitts' Law	77
3.2.2	Three Stages of Human Development	78
3.2.3	Interactivity	80
3.3	Windows, Icons, Menus, and Pointing Device	81
3.3.1	Windows	81
3.3.2	Icons	83
3.3.3	Menus	83
3.3.4	The Mouse and other Graphical Input Devices	84
3.4	Provisions for the Future of the Desktop Model	86
3.4.1	Filing	87
3.4.2	Document-Centered Design	89
3.4.3	User Illusion	90
4	Beyond the Desktop	91
4.1	Web GUI meets Desktop GUI	91
4.2	Provisions for the Future	93
5	Synopsis	97
	APPENDIX	99
	Acronyms	100
	Software	102
	Credits to Figures	104
	References	107

Preface

You are about to read my master thesis. If you are also interested to follow the trails of my voyage of discovery in the World Wide Web you are invited to start at the URL:

<http://www.mprove.de/diplom/>

I would like to thank Horst Oberquelle and Christopher Habel at the Computer Science Department of the University of Hamburg.

The following people have supported my work with valuable comments and copies of hard to retrieve articles: Rolf Schulmeister at the Interdisciplinary Center for Higher Education, University of Hamburg; Michael Friedewald at the Fraunhofer-Institut für Systemtechnik und Innovationsforschung; Ulrich Klotz at the Hochschule für Gestaltung, Fachbereich Produktgestaltung in Offenbach am Main; and Hartmut Obendorf at the Arbeitsbereich Angewandte und Sozialorientierte Informatik, Computer Science Department of the University of Hamburg.

I am much obliged to Alan Kay and Jeff Johnson. Their e-mails helped to shape my understanding of the work at Xerox PARC.

My special thanks go to my American colleagues and friends Irv Kanode and Chris Dryer who have read the manuscript and made essential contributions to content and style.



1 Introduction

Research and development of computer systems have always been a dissension between vision and reality. After development has successfully implemented a working system, nobody remembers the roots. The vision and dreams that led to the product fade away. Years later the original outline might serve as a realistic specification for a new product. At least it might be fruitful to compare the plan with what has been achieved. As technology evolves over time it might have finally become possible to implement the product without restraining the original idea. But the current technology dominates the market, and the old ideas seem to be outdated.

This thesis contrasts the vision of hypertext with the prevailing situation of the World Wide Web. Many compelling aspects of hypertext systems of the last 35 years are not present in the current implementation of the Web, although they should be considered in order to understand where the Web should be improved. A discourse on the history of selected hypertext systems will point out such features and ideas, that are far from being dated.

The same approach will be taken for the field of graphical user interfaces, where graphical user interfaces like Apple Macintosh and Microsoft Windows dominate the scene. It is wise not to mix up the term graphical user interface (GUI) with WIMP interfaces, where WIMP stands for windows, icons, menus, and pointing device. WIMP interfaces are just one possible set of alternatives for graphical user interfaces. They often come along with the desktop metaphor, that has been developed in the 1970s at Xerox PARC. Other characteristic graphical user interfaces are indeed possible, even though they might be hard to imagine directly.

Each interface, that is not based on windows is a good candidate for a non-WIMP interface. MS-DOS of course is not a WIMP interface, but it is not a graphical user interface either. MetaCreations' Bryce, a 3D landscape editor, is an example for a GUI different from the classical WIMP world. And the entire field of information visualization explores new graphical concepts to convey deep structure of data.

What might be the reason to discuss hypertext and GUIs together? First, the history of research and development is already overlapping. And second, it is time to reconcile these two branches of information media that vie against each other on our PC screens.

All pioneers in the field of hypertext and GUI strive for an intensive and interactive dialogue between human and machine. Over 40 years ago, Joseph Licklider called it *Man-Computer Symbiosis* [Licklider 60]. A few years later Doug Engelbart titled his life-long research topic: *A Conceptual Framework for the Augmentation of Man's Intellect*

[Engelbart 63]. The mouse is invented by Engelbart and Bill English in 1963. Interactive text editing, hyperlinking, computer-supported cooperated work (CSCW), video conferencing, among other technologies, are first developed for the system NLS on a time-shared mainframe computer at Stanford Research Institute (SRI).

In the 1970s Ted Nelson reminisces the famous article of Vannevar Bush *As We May Think* [Bush 45] as he entitles an essay *As We Will Think* [Nelson 72]. *Computer Lib / Dream Machines* [Nelson 74] and *Literary Machines* [Nelson 93] – the latter first published in 1981 – depict Nelson’s vision of worldwide hypertext, a universe of literature and personal writings where «everything is deeply intertwined» [Nelson 74, p. DM 45].

The concept of personal computing is formulated by Alan Kay. In his doctoral thesis *The Reactive Engine* [Kay 69] he postulates three principles for computer systems to be used successfully in human-computer interaction. He writes [Ibid., p. 9]:

1. The communications device must be as available (in every way) as a slide rule.
2. The service must not be esoteric to use. (It must be learnable in private.)
3. The transactions must inspire confidence. (“Kindness” should be an integral part.)

This quote shows clearly how Alan Kay anticipated the field of user interface design back in 1969. At Xerox PARC he worked on Smalltalk, and developed windows and menus for the graphical user interface as we know them today. David Canfield Smith and Larry Tesler, also at PARC, are mainly responsible for the idea of consistency and modelessness in interface design. A small generic set of operations should be sufficient to interact with the computer. Together with the conception of a physical office this makes it less complicated for the average user to remember all the commands. The metaphor helps the user to build up a mental model of the system. As long as interaction is coherent with the model interaction can happen with ease.

It only depends on the point of view whether any of those pioneers belongs more to the discipline of hypertext – in the sense of dealing with interrelated text – or to the field of user interfaces. The common vision is to enhance the possibilities for humans to cogitate about the world. They want to build tools for people who think.

The second reason to discuss hypertext and graphical user interfaces side by side is the problematic situation that we are facing on our screens since the advent of the World Wide Web. A browser window opens on the WIMP-desktop and unfolds the worldwide space of information. Inconsistency crawls in as the rules inside the browser window follow a totally different scheme of interaction techniques as the ones that apply outside the browser window. In fact a new non-WIMP graphical user interface proliferates in the middle of the desktop environment. The Web interface neither uses the desktop metaphor, nor other WIMP ingredients as its main principles. The window of the browser application just frames the Web-site on the desktop environment and can therefore be neglected as a constitutional property. Menus are also playing a minor role, because

they are not part of the standard repertoire of interface elements. And icons, respectively images, are more or less just substitutes for textual hyperlinks.

Consistency is a high value in personal computing. Consistency leads to growing confidence in the tools. The user can predict the effect of a click on an icon – it gets selected. She can rely on the effect of a double click on a document icon – the document opens and displays the content in a window. The document can be edited until the window is closed. A window corresponds to a sheet of paper in the real world in such a way that it belongs to exactly one document.

Such basic WIMP interaction rules do not apply for the Web anylonger. A simple single click is sufficient to trigger a link. What follows is that the current document disappears and the content of another document is displayed at the very same place. This is the idea of browsing but it does not match the mental model that has been established with the desktop metaphor.

This is just one example to illustrate that the interference between WIMP interfaces and Web interfaces causes an awful and inelegant environment overall.

Analyzing the current situation is the first step to remove the obstacles in order to come to an interface that suits better the needs of conveying local content like personal documents as well as distant information on Web servers all over the world. Looking back into the history of modern interfaces should reveal the original ideas and principles. Only if we reconsider the visions and all the influences that have shaped these visions into existing products we can decide how to continue the development of interface technology.

The present thesis is divided into three main chapters entitled *Hypertext*, *Graphical User Interfaces*, and *Beyond the Desktop*. The chapters on hypertext and graphical user interfaces share the same structure. They begin with an overview of the development to provide the historical context for the presented systems. Following is a detailed discussion of the systems one by one. The focus lays on peculiarities from the perspective of the present hypertext system called World Wide Web, respectively the perspective of the WIMP desktop model of Apple Macintosh and Microsoft Windows. Before these concepts are summarized, some general theoretical approaches will be presented to obtain a common terminology. The chapter on graphical user interfaces will additionally provide some aspects of cognitive science. The closing sections *Provisions for the Future of the World Wide Web* and *Provisions for the Future of the Desktop Model* consider the potential impact of the historical visions on the current systems.

The chapter *Beyond the Desktop* unveils the inconsistencies between the desktop environment and the World Wide Web with respect to human-computer interaction and the utilized metaphors. The closing section *Provisions for the Future* argues for a fundamental approach to combine both worlds with each other. The deficiencies of each domain can be tackled by means of know-how and experience of the other side.

2 Hypertext

What is hypertext? As the smallest common denominator it can be said that hypertext is text, distributed to a set of discrete sections, with referential links in between.

According to Ted Nelson, hypertext frees the author of the obligation to create sequential text. In some cases it might be more adequate to the topic to choose a form of description that is not necessarily linear or hierarchical. Complex relations are better represented by a network of associated ideas. On the other hand the reader gains autonomy over the text as she is free to decide whilst reading where to proceed in the text when a hyperlink marker shows up.¹

A third dimension of hypertext is the possibility to rearrange existing material according to ones personal reflections on it. A hypertext-aware computer system can be a helpful tool to support the human ability to think – Doug Engelbart would say «to augment the human intellect».

2.1 History

The history of hypertext has seen many different systems. Just a few of them can be presented in this section. A more comprehensive overview has been compiled by Jakob Nielsen in *Hypertext and Hypermedia* [Nielsen 90], respectively in the extended edition *Multimedia and Hypertext* [Nielsen 95]. Many details are taken from this source as well as from *Elements of Hypermedia Design* by Peter Gloor [Gloor 97] and from James Gillies' and Robert Cailliau's history on *How the Web was Born* [Gillies/Cailliau 2000]. Another source about hypertext is the hypertext *Hypertext Hands-On!* It is written and published by Ben Shneiderman and Greg Kearsley using their own hypertext system Hyperties in 1989. [Shneiderman/Kearsley 89].

The trip back in time starts more than 50 years ago. *As We May Think* is an article by Vannevar Bush (*1890 †1974), that was published in 1945 [Bush 45]. During World War II Bush is Director of the Office of Scientific Research and Development and in consequence the highest-ranking scientific administrator in the US war effort. He coordinates the activities of about six thousand American scientists and is especially in charge of the Manhattan Project that develops the atomic bomb [Klaphaak 96], [Hegland 2000].

As We May Think is the description of a hypothetical system called **Memex** (cf. 2.1.1), that supports scientists in their daily work. Bush recognizes the situation that coping

¹ A footnote is a classical form of a hyperlink. It's up to the reader to read – as you have just done – or to skip it. In this sense hypertext can be seen as the “generalized footnote”, a metaphor taken from Jakob Nielsen's book *Hypertext & Hypermedia* [Nielsen 90, p. 2].

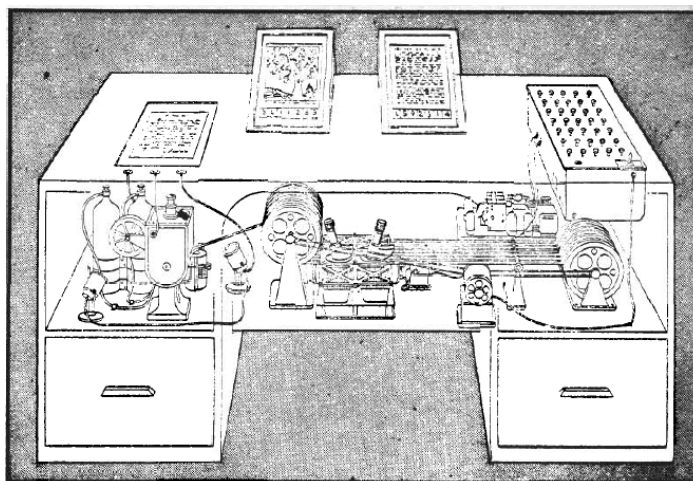


Fig. 2.1 Memex, as it was illustrated for LIFE Magazine, 1945. The desk contains two main microfilm projectors and mechanical apparatus to retrieve the pages for a given trail.

with an increasing number of scientific publications becomes a rising problem. Memex should archive all scientific journals and reports as well as all writings of the owner of the system on microfilm.

To keep track of all the data Memex offers to define trails through the stored articles. This creates sequences of pages that belong to a given chain of thought. Vannevar Bush points out, that classical filing methods like sorting by alphabetical order are artificial and do not correspond to the way humans think. A more natural approach would be to put the articles into context. A set of meaningful relations between the documents that map the associative style of the human mind.

The influence of Bush's article on the field of hypertext cannot be underestimated. Memex' trails count as the first sketch for the concept of hyperlinks.

The term hypertext was coined by Ted Nelson in the early 1960s. In his understanding hypertext stands for non-sequential writing. To the reader hypertext offers several different branches to assemble the meaning behind the written text. It is not possible to articulate orally two ideas at the same time. They have to be put in sequence to be properly told to the recipient of the message. The linear structure of printed books is just deduced from the linear structure of speech. The way such coherent texts are created by humans is far from linear. It is associative, as also Vannevar Bush has said twenty years before. Computers should be used to support the deep structure of thinking. Hypertext should free the author from the need of linearizing her text.

Xanadu (cf. 2.1.2) is Ted Nelson's own attempt for a hypertext system. It is more of a framework than a working program, although several aspects of Xanadu have exceeded the level of working prototypes. The incontestable merits of Xanadu is the influence it took on nearly all hypertext systems to come.

The Advanced Research Project Agency (ARPA) was established by the US Department of Defence in 1958 as a direct response to the Sputnik shock a year before. While NASA's brief was the mission to space, ARPA would initiate projects that had the chance of boosting America's defence-related technologies. The Information Processing Techniques Office (IPTO) was established in 1962. The goal of this ARPA office was to devise new utilization of computers other than plain computation. Joseph Licklider, the first director of IPTO, set the direction towards an «Intergalactic Computer Network» [Segaller 98, p. 39]. Bob Taylor, then a manager at NASA, shared Licklider's vision of the network. As Licklider's successor he devised the ARPAnet, the predecessor of the Internet today. They finally reached their goal to connect the first four nodes by end of 1969.²

After Doug Engelbart had finished his report *Augmenting Human Intellect: A Conceptual Framework* in 1962 [Engelbart 62], funding from NASA and APRA – later also from Air Force's RADC – grew the Augmentation Research Center at Stanford Research Institute (SRI-ARC). His research agenda exposes the design of a system that augments human mental abilities. An essential condition to reach this goal is to enhance the input and output channels of the computer. Engelbart realized, that computer screens can and should be used to display text. Until the late 1950s computer monitors are merely used to display radar data for the air-defence system SAGE [Friedewald 99, p. 95]. Engelbart felt the user should directly interact with the computer system, without dealing with punched-cards, teletype or any other means of batch processing.

The system NLS (cf. 2.1.3) was named after the literal meaning of being on-line with the computer – the oN-Line System – where “on-line” was not used with the sense of today to have a system connected to the Internet. There was no Internet yet. The meaning of on-line in the 1960s was to use the machine interactively. For SRI this was made possible by the use of one of the first time-sharing computers.

The public highlight of SRI was the presentation of NLS at the Fall Joint Computer Conference (FJCC) in San Francisco at December 9, 1968. This session is often referred to as “the mother of all demos”. Doug Engelbart and his team present the mouse, windows, interactive text editing, video conferencing and last not least the hypertext capabilities of NLS. In fact NLS is the first hypertext system that became operational.

The second hypertext system is HES (cf. 2.1.4), which stands for Hypertext Editing System. It was developed by Andries van Dam and Ted Nelson at Brown University in 1967 on an IBM/360 Model 50 mainframe within a 128K partition of memory. HES was actually used by NASA to write the documentation for the Apollo missions [van

² *Nerds 2.0.1* by Stephen Segaller gives an overview to the history of the Internet [Segaller 98]. The first four nodes of the ARPAnet are installed in 1969. The sites are the University of Los Angeles (September 1), SRI (October 1), the University of Santa Barbara (November 1) and the University of Utah (December 1). The fifth node is the company BBN itself, that built the Interface Message Processors (IMP) to connect the local mini computers to the net.

Dam 87].

After van Dam has witnessed the NLS presentation at FJCC he started the development of a new File Retrieval and Editing System. The design goal of FRESS (cf. 2.1.4) was to take the best from NLS and HES, and to overcome some limitations of the prior hypertext systems.

Other historical overviews do not mention **Smalltalk** (cf. 2.1.5) as a hypertext system. In this discourse it should not only be discussed as an important step for graphical user interfaces; Smalltalk also has qualities that make it worth to look at it from the perspective of hypertext. Smalltalk is a programming language that was invented and developed by Alan Kay and Dan Ingalls at Xerox PARC in the early 1970s. It stands in the tradition of SIMULA, the first object-oriented programming language. The structural similarity between referencing objects and hyperlinking will further be discussed in this chapter.

The 1980s was the decade when numerous hypertext systems were created and presented to the public. Workstations and Personal Computers came into widespread use. The IBM-PC with the text-oriented operating system DOS was first marketed in 1981. The first successful computer with a graphical user interface is Apple's Macintosh – introduced in 1984.

NoteCards (cf. 2.1.6) is originally a research project at Xerox PARC starting in the early-1980s. It uses a physical card metaphor; i.e. each card displays its content in a separate window. The system is designed to support information-analysis tasks, like reading, interpretation, categorization and technical writing [Shneiderman/Kearsley 89]. For that reason the focus lies on structuring and editing information compared to a more browsing and reading focus of Hyperties and Guide that run on less powerful personal computers. Consequently a special browser card displays an overview graph to illustrate the connections between the cards.

NoteCards is fully integrated with the InterLISP environment for Xerox workstations. This means that it is highly customizable for the skilled user.³

The field of online documentation is tackled by Symbolics Inc. Since 1985 the entire user manual for Symbolics' workstations has been delivered as an electronic edition. The applications program **Document Examiner** (cf. 2.1.7) is the browser – the corresponding editor **Concordia** is used to create a hypertext that consists of approximately 10,000 nodes and 23,000 links. According to Janet Walker, the designer of the system, this size corresponds to about 8,000 pages for a printed edition (*Document Examiner: Delivery Interface for Hypertext Documents* [Walker 87, p. 307]).

³ Shneiderman and Kearsley give the following example: A LISP program can be written that collects all bibliographic references, creates a card for each reference and connects all cards that cite the reference with the new bibliographic card. [Shneiderman/Kearsley 89]

Document Examiner and Concordia are implemented like NoteCards in LISP. This is no surprise, since the entire operating system for the Symbolics LISP Machine is also done in LISP.

Ben Shneiderman starts **Hyperties** (cf. 2.1.8) as a research project at HCIL around 1983. He takes a very simplified approach in browsing the hypertext in order to attract first time users. Especially museums discover the application of hypertext to support their exhibitions. For example “King Herod’s Dream” at the Smithsonian Museum of Natural History in 1988 or an exhibition about the history of Holocaust at the Museum of Jewish Heritage in New York [Shneiderman/Kearsley 89, p. 33]. A commercial version of Hyperties runs on MS-DOS and uses just a plain text screen. No mouse is necessary to operate the program, although it is possible to click on hyperlinks if a mouse or a touch screen is present.

The development of **Guide** (cf. 2.1.9) starts in 1982 at the University of Kent. Peter Brown has a first version running on a workstation one year later. In 1984 the British company Office Workstations Ltd. (OWL) gets interested and releases a Macintosh version in 1986. Soon thereafter Guide is ported to IBM-PCs. Guide becomes the first popular commercial hypertext system. It shall be noted here that OWL offers also the option to import SGML files into Guide’s hypertext format.

Bill Atkinson was one of the school kids that had contact with Xerox Alto computers and Smalltalk during the 1970s. Adele Goldberg and Alan Kay of Xerox PARC’s Learning Research Group (LRG) conducted a lot of courses for children to evaluate their conception of interaction principles. During the 1980s Bill Atkinson was working for Apple Computer. He was member of the team that designs the Lisa Desktop Manager, he created MacPaint and in 1987 **HyperCard** (cf. 2.1.10). HyperCard uses a cards metaphor. Each card has the same size to fit on the original 9" Macintosh screen. The cards are organized in stacks where the user can flip through. The hypertext functionality comes in as HyperCard is combined with HyperTalk, an easy to learn programming language. A rectangular region can be made sensitive for mouse clicks with a tiny piece of HyperTalk. Most of the time, a click triggers the display of another card in the stack.

The wide acceptance of HyperCard was based on a free copy that was bundled with every Macintosh starting in 1987. For many people, it is their first contact with the concept of hypertext.

Storyspace (cf. 2.1.11) was developed in 1990 by Mark Bernstein. It was initially released for Macintosh only, but a port to Windows also exists. Storyspace is much better suited for writing and reading hypertext than HyperCard is, in that it supports text links rather than just rectangular areas on top of the text layer, that need to be updated if the text underneath moves. Hyperlinks do not have to be coded, they are

created and directly manipulated with the mouse.

Several scientific texts have been written with Storyspace, because the diagram mode visually reveals the logical structure of arguments. Poets use it to write interactive fiction and poems. This is remarkable as non-technicians consort with the field of computer hypertext for the first time.

All of the presented hypertext systems – with the notable exception of Xanadu – are closed hypertext systems. All hypertexts are solitary. They are isolated from each other. Hyperlinking from one hyperdocument into another on the same machine is not possible – even when the creating applications program is the same. Hyperlinking between hypertexts on different computers is utopian. And by the way it is not a matter of missing wires. Nearly all local sites have their PCs connected to a local area network (LAN). And the Internet – then the ARPAnet – is operational since 1969 and hits the mark of 100,000 connected hosts by 1989.⁴ Electronic mail, newsgroups, FTP and remote login are the main services on the net. But hypertext systems that make use of the infrastructure are yet to be deployed.

Intermedia and Microcosm aim to bridge the border between programs and different file formats. **Intermedia** (cf. 2.1.12) was developed – once again – at Brown University in 1985. Its advanced concept to administrate links between documents of arbitrary programs is unique for the time. Intermedia is a model how to extend the operating system to provide a common API for all application programs to share linking information with each other. Unfortunately Intermedia did not get much attention because its target platform is A/UX, a Unix derivate for Macintosh that was not in widespread use. **Microcosm** (cf. 2.1.13) is a research project at the University of Southampton. Like Intermedia five years before, Microcosm is designed around the idea that material of diverse sources and formats should be tied together into one hypertext. To achieve this goal links have to be separated from the documents. They are stored and maintained in special link databases. In contrast to Intermedia, Microcosm plays the active role in gathering the necessary data out of the documents. For each topic it stores data on which documents contain information about it and also where inside the documents the information is located. This elaborated conception of links can be seen more as a kind of Memex' trails than simple one-to-one links. Given this structure dynamic linking becomes possible. This means that links can be associated with generic text strings. Wherever this string shows up a link is placed automatically. This has the effect that documents that are imported for the first time into Microcosm can have links immediately.

Microcosm becomes the prototype of an Open Hypermedia System (OHS) and is under continuous research and development until today.

⁴ ARPAnet switched to TCP/IP in 1983. Hence the Internet was born. The number of nodes increases rapidly: 1,000 in 1984, 10,000 in 1987, 100,000 in 1989; 1,000,000 in 1992. (Timeline in *Nerds 2.0.1* [Segaller 98])

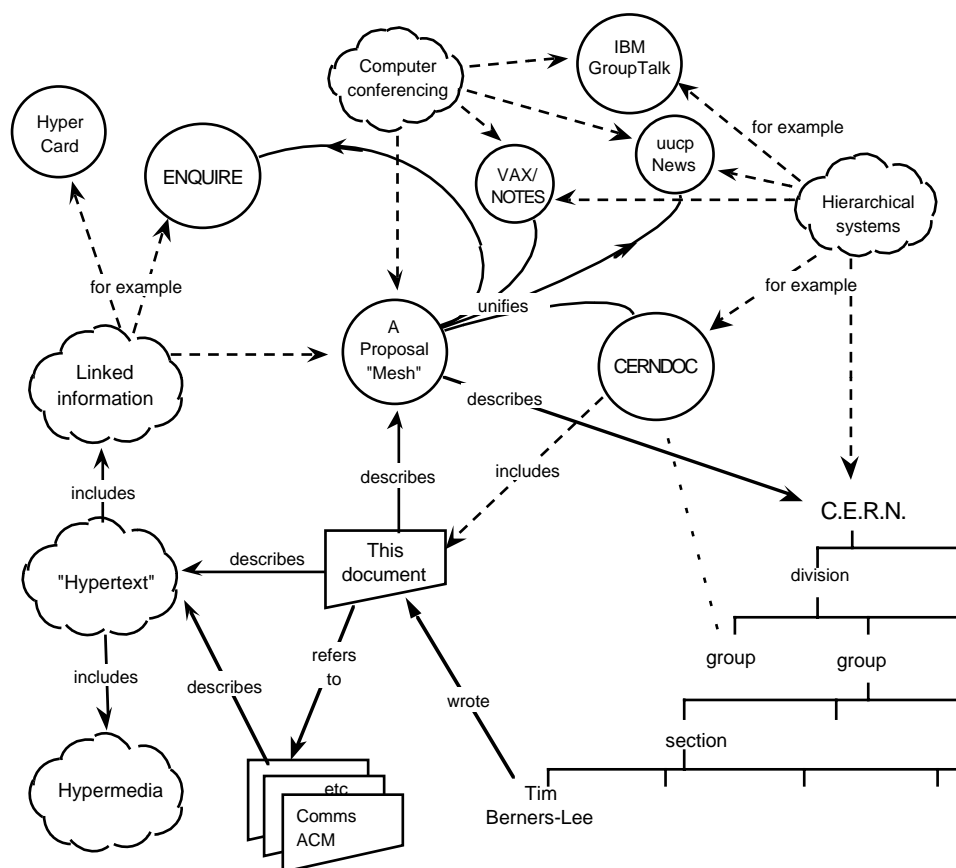


Fig. 2.2 Tim Berners-Lee's diagram for the World Wide Web, then named "Mesh", 1989

The **World Wide Web** (cf. 2.1.14) was born at the international laboratory for particle physics CERN in Geneva. CERN's scientific community is made up of several thousand people. They use a wide variety of different hardware and software. Keeping track of this organism is difficult; exchanging documents electronically is even more difficult because the incompatibilities between the systems are manifold. Furthermore many physicist are not located in Geneva itself. They work remotely from all over the world. Tim Berners-Lee and Robert Cailliau propose a distributed hypertext system to solve these problems. *Information Management: A Proposal* is written in 1989 [Berners-Lee 89], *WorldWideWeb: Proposal for a HyperText Project* a year later [Berners-Lee/Cailliau 90]. The first Web server becomes operational by end of 1990 with the URL address <http://info.cern.ch>. The first Web browser is called WorldWideWeb (cf. Fig. 2.12 on page 33), also working since December 1990.

Fig. 2.2 is taken from the cover of Berners-Lee's proposal. It depicts one of the reasons for the success of the World Wide Web. The Web unifies existing information networks as early Web browsers are capable of accessing services like UUCP newsgroups, FTP and WAIS. Just one applications program is needed to access all the diverse sources – a boost

in ease of use compared to the many different programs on many different platforms that were necessary before.

Furthermore the Web builds on existing technology, i.e. the hypertext transfer protocol (HTTP) builds on top of TCP/IP. Web pages are encoded with the hypertext markup language (HTML), a simple application of SGML, with the intended side effect that only ASCII characters are used. This makes it easy to edit and transfer HTML files with existing software between all platforms.

During the following years browsers are developed for all main operating systems, i.e. Mosaic for X-Windows by Marc Andreessen at NCSA released early in 1993, Macintosh and Windows versions follow by November the same year. Ten years after the first Web server has started at CERN the number of Web servers reaches ten million world wide.⁵ The genesis of the Web is described comprehensively by Tim Berners-Lee in *Weaving the Web* [Berners-Lee 99] and by Robert Cailliau in *How the Web was Born* [Gillies/Cailliau 2000].

Finally **Hyper-G** (cf. 2.1.15). This project starts also in 1989 at Graz University of Technology. Hermann Maurer and his team aim for a kind of networked version of Microcosm. Many of the flaws of the Web are identified and addressed. But when Hyper-G was presented in 1995 it was too late to get momentum for the new product. The Web was more exploding than growing and left no chance for Hyper-G. At least it will become interesting to contrast the World Wide Web with the competing approach of Hyper-G, respectively with HyperWave as the commercial version is named.

The following sections will take a closer look at the hypertext systems presented so far. The main focus lies on the question of what compelling concepts and features have been lost on the way to the predominant hypertext system World Wide Web.

2.1.1 Memex

Memex is designed with the scientific researcher in mind. Plenty of books, reports, magazines and newspapers are published every month. But classical methods of indexing stall at this amount of material. In *As We May Think* Vannevar Bush writes [Bush 45, p. 43],

Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.

⁵ The mark of 10 million is passed in February 2000. In July 2001 we have reached over 30 million Web servers. Source: *Hobbes' Internet Timeline v5.4* [Zakon 2001].

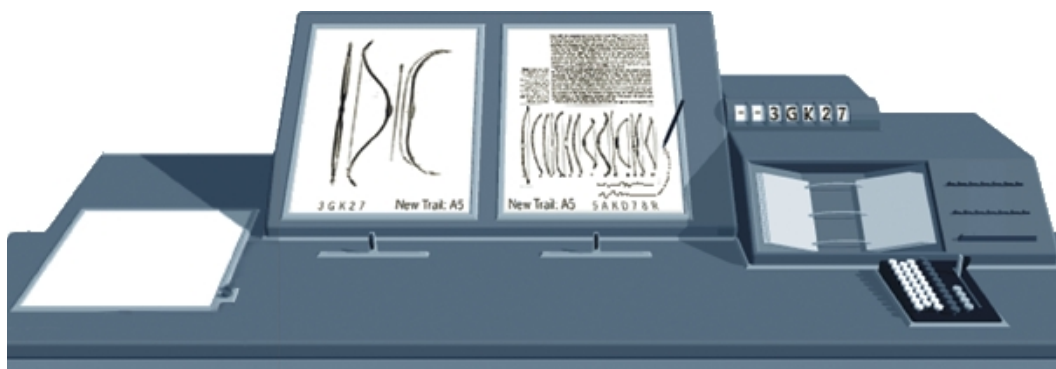


Fig. 2.3 This illustration is based on Bush's description of Memex: «On the top [of the desk] are slanting translucent screens, on which material can be projected for convenient reading. [...] At the bottom of each are a number of blank code spaces [...] The user taps a single key, and the items are permanently joined.» [Bush 45, p. 43-44]

Hierarchical filing is not sufficient to permanently enhance our ability to cope with the record. Bush takes the human ability of association as model to propose a better scheme. He continues [Ibid.],

The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain.

Another scheme is already in place that corresponds to human associations. As soon as a scientist gets interested in an article she might take the references as recommendations to get more material on the topic. Studying those papers leads to the next level, and so forth. During research she is forming a characteristic path through the literature. The creation of this path is the central idea behind Memex.

All articles are stored on microfilm inside the machine. They can be projected on some translucent screens on top. The user builds **trails** along the way of articles she reads. Two articles are connected by simply pressing a button. The entire trail is named and stored in a **code book** inside of Memex for later reuse. If our researcher comes across a page that is already part of another trail she can benefit from this and follow the other trail that might be related to her current interest.

Trails count as the first conception of hyperlinks. Although – considering the Web's understanding of links – one should better say the first conception of guided tours. Trails are a sweeping concept that is on a higher abstraction level than HTML hyperlinks.

Browsing through the pages of a book or the pages of a trail happens with breathtaking ease. A kind of joystick is used for this purpose. Once again Vannevar Bush [Ibid.]:

On deflecting one of these levers to the right he runs through the book before him, each page in turn being projected at a speed which just allows a recognizing glance at each. If he deflects it further to the right, he steps through the book 10 pages at a time; still further at 100 pages at a time. Deflection to the left gives him the same control backwards. A special button transfers him immediately to the first page of the index.

Not enough. Annotations and authoring are also possible. Therefore the projection screens can be photographed from underneath and the page is stored on microfilm in the repository. Such pages can become part of trails like any other page.

It shall be mentioned again that Memex has never been built. But the existing technology of 1945 – microfilm, dry photography, photocells – lead Vannevar Bush to the assumption that his concept is realistic for a future not far away. His judgment regarding to the evolution of technology was wrong. But his ideas became inspiring for the development of hypertext. The book *From Memex to Hypertext: Vannevar Bush and the Mind's Machine* by James Nyce and Paul Kahn [Nyce/Kahn 91] gives evidence for the profound influence of Bush's vision.

2.1.2 Xanadu

Ted Nelson strives for just a «decent writing system», as he says in his book *Computer Lib / Dream Machines* [Nelson 74, p. DM 59]. His central idea is a **docuverse**, a universe of documents, where a new form of literature can proliferate without the limitations of a linear medium like the printed book. A condensed version of his vision can be found in the preface of the 1993 edition of *Literary Machines* [Nelson 93, p. 10],

At your screen of tomorrow you will have access to all the world's published work: All the books, all the magazines, all the photographs, the recordings, the movies. (And to *new kinds* of publication, created especially for the interactive screen.)

You will be able to bring any published work to your screen, or any part of a published work.

You will be able to make *links* – comments, personal notes, or other connections – between places in documents, and leave them there for others (as well as yourself) to follow later. You may even publish these links. [...]

Any document may quote another, because the quoted part is brought – and *bought* – from the original at the instance of request, with automatic royalty and credit to the originator.

Xanadu is an continuously ongoing project since 1960. The system serves for Nelson as a framework to implement the concepts that constitute hypertext in the direction depicted above. Parts of this long term research project finally became public under the Open Source model in 1999. Other aspects have been demonstrated as mockups and working prototypes.

The fundamental problem of hypertext is, according to Ted Nelson in *Parallel Visualization: Transpointing Windows* [Nelson 98a], the ability to see connections side by side.

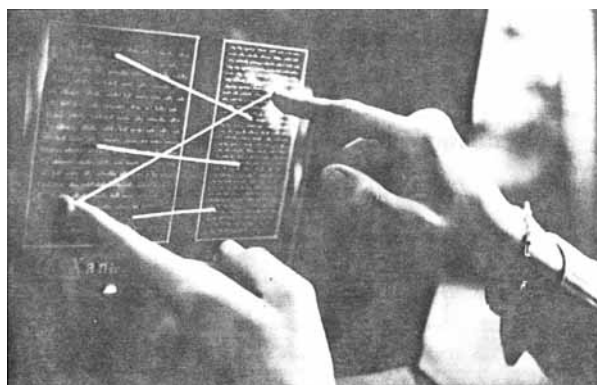


Fig. 2.4 A mockup showing *Parallel Textface*TM for the *Xanadu System*, 1972

The links themselves should be visualized on screen. For that reason early versions of Xanadu are designed around the concept of *Parallel Textface*TM. The mockup shown in Fig. 2.4 uses a cardboard and a celluloid picture to simulate a computer screen. It illustrates how connections between two columns of text should appear. The lines ought to be moving as either of the columns is scrolled by the user.

The correspondence between Xanadu's parallel text columns and Memex' two adjacent screens (as shown in Fig. 2.3) is obvious.

With the advent of windows of the graphical user interface the concept of *Parallel Textface*TM evolves to *Transpointing Windows*. Therefore the lines need to cross the window borders and bridge to the next window.⁶

Ted Nelson has a much deeper understanding of hypertext systems than just the user interface issues. He requires four fundamental qualities: Consistent hyperlinks between content, robust re-use of content, support of versioning and support for parallel documents.

The first two principles touch on connecting any two pieces of content, either by hyperlinking or by **transclusion** respectively. Transclusion means a sort of quotation without copying the content but borrowing it from the original source. None of these connections should ever break. This calls for a persistent identification mechanism for content, and the content itself should never be deleted from a xanalogical hypertext system. But of course changes are permitted. Hence the system has to cope with several revisions of the same content – versioning is the third principle. And finally, what makes up a parallel document? A flock of versions and variations of a single theme or idea is called a parallel document.

⁶ The Web editor Adobe GoLive implements such a behavior for the creation of HTML hyperlinks, *Adobe GoLive's Point & Shoot: an interface technique for creating hyperlinks* [Müller-Prove 99].

One of Ted Nelson's *Examples of Parallel Documents* [Nelson 98b] takes Hamlet. *Hamlet*, he says, is not just one fixed piece of text. Shakespeare himself has created several versions. The play has been translated to other languages and to other media, consider all the Hamlet movies acting Lawrence Olivier, Richard Burton, Mel Gibson and Kenneth Branagh. Hamlet has also been studied and therefore been annotated by English language and literature scientists. If one refers to "Hamlet" one means the wholeness of documents that have been created under the same idea.

Another example that might sound less offside is about email. It is inspired by Alan Cooper's book *The Inmates Are Running the Asylum* [Cooper 99, p. 61]. A conversation via email produces a bunch of single mails. It is desirable that the system can treat all related mails as part of a sequence – as one parallel document.

The four xanalogical principles above are discussed in *Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use* [Nelson 99a]. Many, if not all existing hypertext systems, fall short if they get evaluated against Nelson's criteria.

2.1.3 NLS/Augment

NLS is the first hypertext system that became operational, although it was never primarily designed as such. The objective was to build a new tool to "augment human intellect". The computer should be used for an interactive dialog with the user. Its flexibility and symbolic manipulation capabilities should support human idea processing. The interface related aspects will be discussed in 3.1.3 NLS/Augment (p. 56). This section will focus on the hypertext qualities of NLS.

NLS is implemented on one of the first time-sharing computers, an SDS 940. About six terminals were installed by 1968 (cf. Fig. 3.2 on page 57). An NLS process is directed for each terminal, which has access to the same set of files.

Each file in NLS is highly hierarchically structured. It is segmented into **statements** of limited size, that get an identifier according to its serial location in the text, e.g. 1, 1a, 1a1, 1a2, 1a2a, 1a2b, 1b, 1b1, etc. The identifiers can be hidden. And many other commands are provided to change the view. For example it is possible to display the outline of a file down to the second level with just 3 lines of text for each statement.

Identifiers can be used to link to the corresponding statements. But they are vulnerable against file manipulation. If the content of a file is modified, the identifiers might change as well and the link points to a wrong statement. Doug Engelbart and his team are aware of this difficulty and offer several more options to specify the destination for a hyperlink. They are presented in *Authorship Provisions in Augment* [Engelbart 84, p. 112, 6]⁷. NLS has three ways to address statements directly. First are **structural statement numbers** – this are the identifiers mentioned above. Second are **statement iden-**

tifiers (SID). SIDs are unique integers that are assigned in ascending order to statements once they are created. They stay unchanged as long as the statement exists in the document. Third are **worker-assigned statement names** or labels. A word in parenthesis – or any other specified delimiters – is treated as a label for the statement and can be used to link to the statement.

In addition to the direct methods a statement can be identified by **text and content addressing**. Halasz and Schwartz give the example of a link to «the statement containing the word ‘pollywog’» [Halasz/Schwartz 94, p. 33].

Addresses can be extended with supplements. Most of the **relative address extensions** deal with the structure of the file, like UP A LEVEL, DOWN A LEVEL or SUCCESSOR AT SAME LEVEL. A peculiar kind of extension is the **indirect link referencing**. Once a statement is found NLS looks for the first (or second, ...) link in the statement and follows it to the next statement. It is possible to do this kind of detour several times in succession.⁸

The linking mechanism of NLS does not stop at file boundaries. Other files in the same directory or in different directories can be specified in similar fashion.

Finally in an interview with Frode Hegland, Doug Engelbart recounts on **implicit links** in NLS [Hegland 2000, Hypertext/Hypermedia]. Wherever a special term like e.g. “mouse” shows up in the text a macro command can be provoked to automatically jump to the corresponding description in a glossary file. Thus links can also be calculated by the system.

In *Literary Machines* Ted Nelson pays tribute to the invention of the **text link** by Doug Engelbart [Nelson 93, p. 7]. All link specifications that have just been presented can be attached to any sequence of text in a statement. A single click with the mouse to the text is sufficient to display the linked text passage on screen [Hegland 2000, section on Windows].

It is very important to Doug Engelbart that all files in NLS follow the same hierarchical structure with no exception from documents and e-mails to source code files. Every statement in any file can be referenced with the same mechanism. The automatically generated statement identifiers guarantee a high granularity to link any paragraph separately.

Electronic mail and the Journal are part of NLS since 1970. They fit into the second phase of Engelbart’s research program not only to augment individuals but also to augment working groups. The Journal is a permanent storage for NLS documents. Once a file is put into the Journal it gets a reliable address, that can be used by other

⁷ ‘6’ is the number of the statement in the Web edition of [Engelbart 84].

⁸ For example (5c “*D”.1) represents the path “go to statement 5c, scan for first occurrence of ‘*D’, then follow the next link found in that statement”. The syntax for addressing statements is described in detail in [Engelbart 84, p. 112, 6].

documents for linking.

According to *NLS Teleconferencing Features: The Journal and Shared-Screen Telephoning* [Engelbart 75, 7], about 30,000 items have been entered during the first five years. For 1984 the number has exceeded 100,000 entries [Engelbart 84, p. 122, 10C7].

Notably all e-mails at SRI-ARC are stored in the Journal and can therefore be referenced by other mails. Engelbart gives an example for such a message [Engelbart 84, p. 121, 10C4]:

“Frankly, John, I think your comment in (DDD,xxx,aa) is a mistake! Didn't you notice the earlier assumption in (DDD,xxx,bb)? Maybe you should go back to Tom's earlier requirements document [...] – especially at (EEE,yyy,cc).”

The expression (DDD,xxx,aa) represents a citation link to a file with the Journal item number 'xxx' in the Journal 'DDD'. The 'aa' part is the address pointing to a specific passage in that Journal file. A click on such a link brings up the original document.

The Journal is really remarkable. It shows how a community of people can benefit from a holistic approach that combines e-mail and hypertext.

2.1.4 HES and FRESS

The first major conference on hypertext was *ACM Hypertext '87* in Chapel Hill, North Carolina. The keynote speaker was Andries van Dam, who created the Hypertext Editing System (HES) together with Ted Nelson twenty years before at Brown University. In his keynote address he refers to the system and describes its qualities, *Hypertext '87 Keynote Address* [van Dam 87]. HES' two main objectives are the online production of printed documents and the exploration of the hypertext concept. The first goal is met insofar as NASA used the system successfully to produce the user manuals for the Apollo mission to Moon. Invisible control information allows the printing of the entire hypertext in linear form.

The second goal is strongly influenced by Ted Nelson. HES and Xanadu share the same approach in memory management. Instead of directly editing raw text, edits are done by pointer manipulation to text fragments. This architecture promotes the distinction between **inclusion** and **reference** in HES. Inclusion, respectively Ted Nelson's term **transclusion**, uses instances of text in several places. If one instance of text is modified the changed text will show up in all other places as well. Van Dam comments nonchalantly, «Instances are a standard idea from computer graphics – no big deal.» [van Dam 87, p. 889]. This “standard idea” was invented by Ivan Sutherland in the early 1960s and will be presented in 3.1.2 Sketchpad (p. 54).

After attending Doug Engelbart's legendary NLS demo in 1968, Andries van Dam and his team start all over. Their second hypertext system FRESS – the File Retrieval and Editing System – builds on their experience with HES and incorporates also concepts

from NLS. FRESS does outline processing like NLS, but is less rigid. In particular, no limitation in statement size restrains the user. For that reason the system allows «a more freeform editing style» [Ibid., p. 890]. FRESS is also more responsive than its predecessors. The speed is about the same whether the user is working on a 2-page or on a 200-page document.

The link handling is also improved in FRESS. For the first time bi-directional links are implemented. They are called **jumps**. Uni-directional links are called **tags** and have a special purpose. Nicole Yankelovich explains the difference in *Reading and Writing the Electronic Book* [Yankelovich et al. 85, p. 23]:

A tag—a one-way link—indicated a connection to a single element such as an annotation, definition, or footnote. When a reader pointed to a tag [...], the associated text appeared in another window [...] for reference while the reader remained in the main document. Unlike a tag, a jump—a bidirectional link—indicated a path to another document. By following a jump, the reader was transferred from one document to another [...]. Since cross-reference markers (destinations of links) were displayed in the text, readers could backtrack through a sequence of links [...].

Links are not stored within the documents. In a collaborative environment the external link database can be used to show or hide the links that have been defined by other users. Hyperlinks are also typed. That means that they can be labelled with keywords. Links with the same label form a path that can be followed like Memex' trails.

There was no mouse at the IBM/360. A light pen was used instead to point on hyperlinks. James Gillies and Robert Cailliau write, «It was more like point-and-kick than point-and-click: you would point with the light pen and click with a foot pedal.» [Gillies/Cailliau 2000, p. 104].

FRESS provides also annotation capabilities. Van Dam recalls that this led to the first «electronic graffiti» in an English poetry course where FRESS was used in the early 1970s [van Dam 87, p. 891].

2.1.5 FLEX and Smalltalk

Alan Kay saw the idea of object orientation several times during the 1960s. He describes in *The Early History of Smalltalk* [Kay 96] how he had to analyze an ALGOL60 program that turned out to be the programming language SIMULA, «the documentation read like Norwegian translated into English, which in fact it was.» [Ibid., p. 516]. SIMULA67 is formulated by Kristen Nygaard and Ole-Johan Dahl as an extension to ALGOL60. It introduces the concepts of classes and instances to programming – then called activities and processes. SIMULA67 becomes the first object oriented programming language. Ellis Horowitz gives an introduction to ALGOL and SIMULA in *Fundamentals of Programming Languages* [Horowitz 84, p. 251].

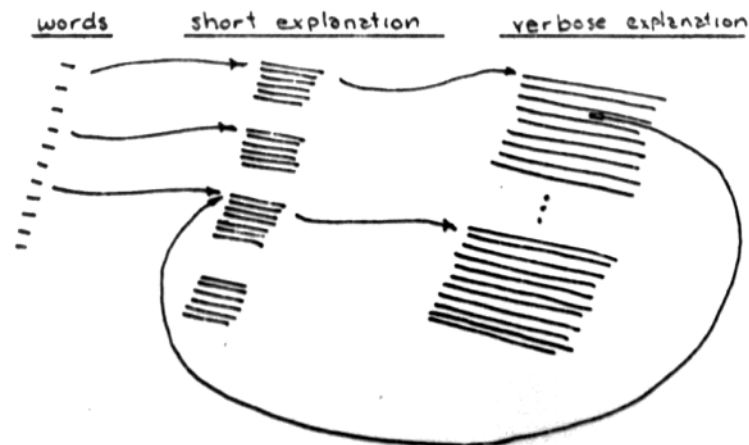


Fig. 2.5 Sketch for an interactive version of Webster's Dictionary for the Flex Machine by Alan Kay, 1969

Another influential step on the way to Smalltalk is Kay's design of the Flex Machine in his doctoral thesis *The Reactive Engine* [Kay 69]. As an example utilization for the Flex Machine Kay presents an interactive version of Webster's Dictionary, [Ibid., p. 157]:

The desired word can be [...] pointed by the stylus. The display can then show a condensed explanation of just the word pointed to. If more information is needed, the stylus can again be used, to cause an expanded "encyclopedia" entry to be displayed. More tricks can be played. Suppose a word in the explanation is not understood. Does the user have to retreat to the top level to select the entry for the new word? Not if vocabulary words are linked together. Then the stylus can be used to point at the word *in the explanation itself*^[9] and its entry will then be displayed.

To implement such functionality Alan Kay devises a new programming language called *Flex – A flexible extendable language* [Kay 68]. It follows the tradition of ALGOL and EULER, but goes beyond those in object-oriented aspects. FLEX is interpreted and therefore highly interactive like JOSS by RAND Corporation. But JOSS has deficiencies in dynamic simulation and extensibility which are addressed in FLEX [Kay 96, p. 517]. New operators may be declared directly and even FLEX itself can be modified using FLEX. The user interacts with the Flex system only by using FLEX and all characteristics of the system are expressed in FLEX itself. This peculiarity is called homoiconic. That means that the internal and external representations are essentially the same and the system uses only one language to interact with the user.

FLEX shares these characteristic features with Smalltalk, which make it to a direct predecessor for Smalltalk.

⁹ This phrase was originally underlined. Back in 1969 it was not only for Alan Kay's typewriter impossible to create italics typeface.

Alan Kay and Dan Ingalls develop Smalltalk in 1972 at Xerox' new research center in Palo Alto – abbreviated to Xerox PARC. The founding principle for Smalltalk is that every item is an object, from numbers to windows to projects up to the entire system itself. Objects refer to each other and messages are sent to and fro to change the internal states of the objects.

The Smalltalk environment is not a dedicated hypertext system. But referential connections between objects of all kind, i.e. between text objects, belong in the category of hypertext. Alan Kay writes in an e-mail to the author [Kay/Müller-Prove 2001]:

All of the Smalltalks at PARC had hyperlinks, not just between “content”, but between “projects” (the GUI there was not just the first overlapping window interface, it also had what we would call today “multiple desktops” that were connected via hyperlinks.)

If an ENTER message is sent to a project the corresponding desktop area opens and displays the current state of the project, i.e. «the windows for the tasks involved in that project», as Larry Tesler explains in *The Smalltalk Environment* [Tesler 81, p. 144]. References to projects can be placed anywhere in the system.

The relevance of Smalltalk for graphical user interfaces will be discussed in section 3.1.5 Xerox Alto, the Interim Dynabook and Smalltalk (p. 62).

2.1.6 NoteCards

One of the developers of NoteCards, Frank Halasz, describes the hypertext system in *Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems* [Halasz 88]. Halasz and his team at Xerox PARC use the physical world of index cards as metaphor. A card corresponds to a window, i.e. each card has its own window. Hyperlinks are references to other cards, but not to special locations within the cards. The framed title of a card is used as a link marker to the card. Clicking the box does open a new window to display the content of the destination card – or the window comes to front and gets activated if the card was already open.

In addition to the standard text and graphic cards, NoteCards has more card types built in. These are **filebox** and **browser cards**. They support the user in sorting and categorizing the content cards. A filebox is a simple way to collect cards that have some aspects in common. Fig. 2.6 shows two filebox cards in the lower left corner. Fileboxes can also contain other filebox cards and provide to such an extent a hierarchical structure among the cards. A browser card shows a diagram based on the link structure between the cards. The large window in Fig. 2.6 is an example for this.

Cards are interconnected by typed links. That means a label is assigned to a link to describe the kind of relationship between two cards.

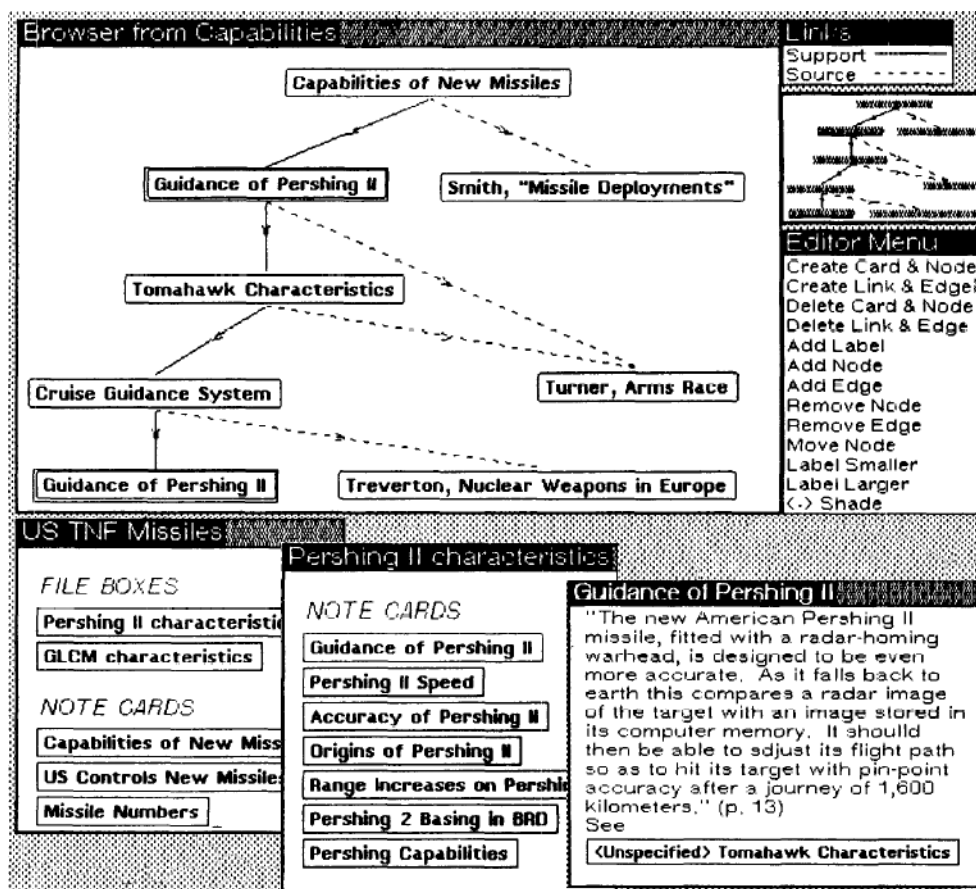


Fig. 2.6 *NoteCards*. The browser card shows a structural diagram of nodes and typed links. 2 filebox cards (bottom left and center) hold links to other cards. The standard note card (bottom right) contains a link to a card entitled "Tomahawk Characteristics". ('<Unspecified>' probably means that the link type is yet undefined.) The example is taken from the working materials of a graduate student in history, who has used *NoteCards* to structure the domain of the research topic [Halasz 88, p.

Frank Halasz continues his paper with the discussion of 7 issues, that are crucial in his opinion for the next generation of hypertext systems [Ibid., p. 841] (respectively his presentation at the first hypertext conference 1987 with the same title [Halasz 87, p. 352]). Four points will be presented here.¹⁰

Search and Query in a Hypermedia Network. Sophisticated search capabilities should extend and complement the navigational character of link following. This should act against increasing problems with the user's orientation in hypertext. Content searches like «all the nodes containing the string "hyper*"» [Halasz 88, p. 842], as well as structure searches should be provided. An examples for a structure search would be, «all subnetworks containing two nodes connected by a [link of the type SUPPORTS], where

¹⁰ The skipped items are Issue 4: *Computation in (over) hypermedia networks*, Issue 6: *Support for collaborative work* and Issue 7: *Extensibility and Tailorability*.

the destination node contains the word “hypertext”» [Ibid.]. Another example would be the search for «a circular structure containing a node that is indirectly linked to itself via an unbroken sequence of “supports” links» [Ibid.]. This query would reveal circular arguments.

Composites – Augmenting the Basic Node and Link Model. Filebox cards and browser cards serve a special purpose in NoteCards. They are temporary provisions for a missing concept in the ordinary hypertext model with nodes and links. Dealing with a set of nodes should be well integrated into the model. For example a specific group of hypertext nodes describes various aspects of a juristic case. It should be possible to refer to the case as a whole rather than linking to all nodes individually.

Virtual Structures for Dealing with Changing Information. Not all kinds of data fit into the static structure of hypertext. Information is in flux and requires the author to update the network ceaselessly. Just take a news ticker as an extreme example for a source of continuous flow of information. Future hypertext systems should be able to manage this kind of situations.

Versioning. The area of versioning has already been mentioned as one of the xanalogical conditions (cf. page 15). Halasz carries on with versioning for changing link structures. And he rises the question of semantics: Is it correct to automatically link to an updated node without verifying the link?

Many of Halasz’ statements sound also familiar in the Web context, albeit they have been written nearly fifteen years ago and long before the Web took off. *Search and Query* – Search engines that index and categorize Web services like Google and Yahoo play an important role for the World Wide Web. *Composites* – just the concepts of filebox cards and browser cards integrated into the Web would boost the current appearance of the Web. Possible areas for application are guided tours, site maps, and bookmark lists. Established interface standards for composites could support the user in navigating the Web. *Virtual Structures* – Content Management Systems deal with databases and XML templates to dynamically create Web pages. Finally *versioning* is still an unsolved but important problem.

2.1.7 Symbolics Document Examiner & Concordia

For the first time browsing and authoring hypertext is handled by two different application programs. Symbolics Document Examiner is used for viewing, while Concordia is specialized on authoring Symbolics’ hypertext. The reason might be the kind of text, that is published by Symbolics in hypertext form. As a computer manufacturer they provide the user with documentation for their workstations. And user manuals change reasonable rarely.

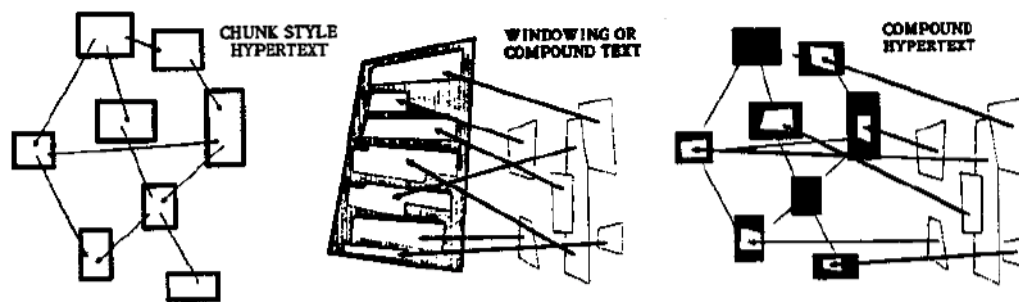


Fig. 2.7 Sketches by Ted Nelson for chunk style hypertext, windowing text, and compound hypertext

The architecture of Symbolics' hypertext seems to be inspired by Doug Engelbart's NLS and by Ted Nelson's Xanadu, respectively HES. The basic unit for Symbolics' hypertext is a **record**. It has a title and contains the description for a specific topic. Keywords and a oneliner should also be provided by the author. Each record has a unique identifier with the same persistent qualities as the statement identifiers in NLS (cf. page 16). The records are stored in a central database on the Symbolics workstation.

Just linking the records would not make up a proper online manual. The experience for such purpose should be founded in the book metaphor. A document-like flow of text is created by assembling a sequence of records into a single window. In *Literary Machines* Ted Nelson calls this technique **compound text**; respectively he prefers his newly coined term **windowing text** [Nelson 93, p. 1/15]. An interconnected network of windowing text documents is called **compound hypertext** [Ibid., p. 1/16].

The records in Symbolics' hypertext are glued together by **inclusion links**. Inclusion and three other forms of linking are explained by Janet Walker in *Document Examiner: Delivery Interface for Hypertext Documents* [Walker 87, p. 310]:

Inclusion. An inclusion link specifies that the content fields of the record referred to are to be included at that location when a reader is reading the document.

Precis. A precis link specifies that the title and oneliner fields of the record are to be included at the location of the link.

Crossref. The result of a crossreference link is to insert a conventional crossreference at the location of the link, for example, "See the section Combatting Gnats."

Implicit. As writers create the material, they can enclose the names of some topics in implicit name links.

Symbolics' hypertext browser Document Examiner divides the screen in four panes. The content area, the Candidates and Bookmarks pane and the command region below.

The content area normally displays the documents, but right now in Fig. 2.8 a graphical overview of the documents is presented.

A click on any link does not immediately jump to the destination; instead the link is

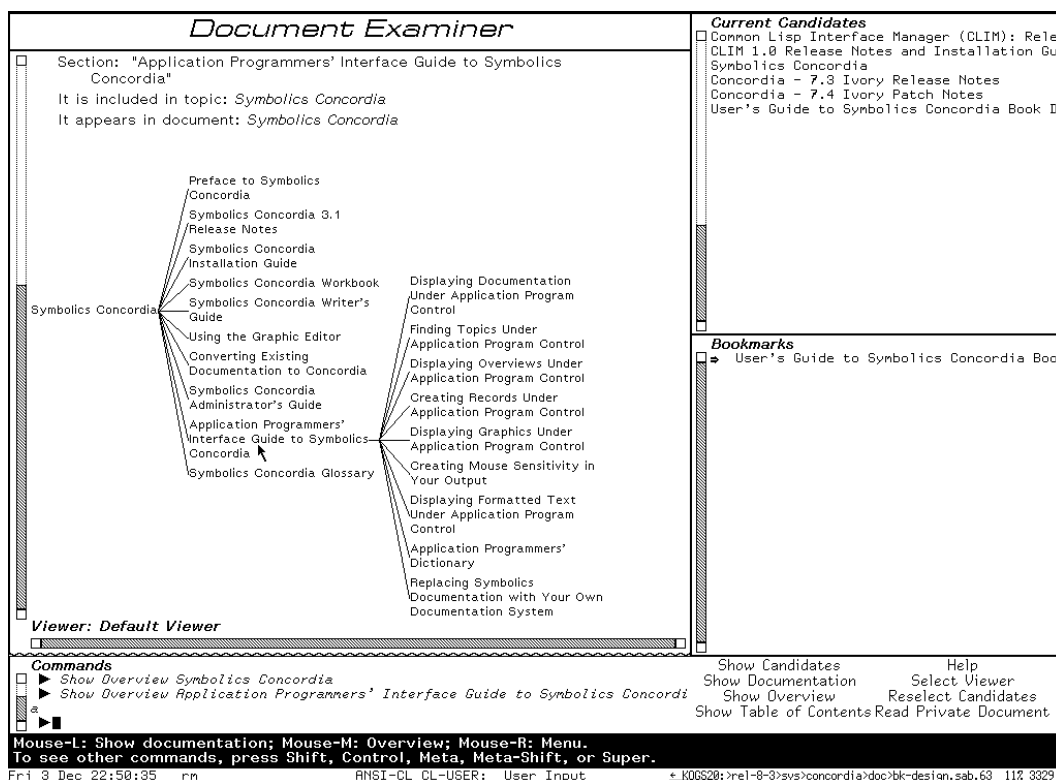


Fig. 2.8 Symbolics Document Examiner with a diagram displayed in the main content pane. 6 links are currently listed in the Candidates pane (top right), but only one bookmark link. (right).

added to a list of candidates. Eventually a click on a link in the Candidates pane opens the document in the content area. This behavior has turned out to be useful in the application of online help, because users are looking for information and like to preselect some topics that might solve their problems. The result of search operations is also displayed in the Candidates list.

References to documents can also be saved as **bookmarks** – a concept also based on the book metaphor. A dedicated Bookmark pane holds the links for later use.

Annotations capabilities are underdeveloped in Symbolics' hypertext system and call for improvement. All preceding hypertext programs offer an integrated reading and writing environment. But the differentiation between the read-only browser and the editor Concordia has taken the editing features from Document Examiner. Furthermore the concept of annotations fits elegantly into the book metaphor. Janet Walker points out the necessity to integrate annotations with versioning. Annotations need to be maintained for each new release of the user manual in hypertext form [Ibid., p. 321].

Janet Walker nominates a second topic that is of interest for future research. It should be possible to constrain the context for search operations. Readers like to define the

boundaries before they start a full-text search or keyword query [Ibid.] (cf. the “7 Issues” by Frank Halasz on page 22).

2.1.8 Hyperties

Hyperties takes a very simplified approach in browsing hypertext. The commercial version for IBM-PCs from 1987 displays each article on the entire screen. And all interactions can be performed with the arrow keys. The user moves the cursor until the link of interest is highlighted. Pressing ENTER causes a jump to the link target. Jacob Nielsen reports, that this special usage of arrow keys as jump keys is significantly faster than the same operations performed with the mouse [Nielsen 90, p. 120].

A node in Hyperties is called **article**. It has a title and a short description about its content. The title is used to automatically place links wherever the very same text phrase appears in other articles. This is very restricting because no other way of creating links is possible. The description is used as a preview for a link. If the user highlights a link the description is displayed at the bottom line of the screen. Most of the times this is sufficient to decide whether to jump to the link or not [Ibid., p. 89].

2.1.9 Guide

Guide is probably the first hypertext system that has underlined hyperlinks¹¹. It offers three different forms of links: jumps, pop-ups and replacements. The links look initially the same; yet they can be distinguished because the mouse cursor changes its shape accordingly.



Fig. 2.9 Special mouse cursors indicate reference jumps, pop-up notes, open and close of inline replacements. Further to the right comes the standard Macintosh arrow cursor and the link cursor used by the Web browser Netscape Communicator.

Jumps are hyperlinks to other nodes, but they can also refer to locations within the same node. The mouse cursor changes to an arrow. Today’s browser use a pointing forefinger for this purpose. **Pop-up links** display a tiny extra window as long as the mouse button is pressed down. Fig. 2.10 illustrates this behavior that corresponds directly with the classical function of footnotes. The asterisk cursor is also derived from this metaphor. **Replacement links** reveal a more extensive description of the topic at the position of the hyperlink marker. The text gets stretched out. The reverse operation cuts down the text again.¹²

¹¹ According to Harald Weinreich and Hartmut Obendorf in *The Look of the Link – Concepts for the User Interface of Extended Hyperlinks* [Weinreich et al. 2001], different text styles like bold and italic are also possible.

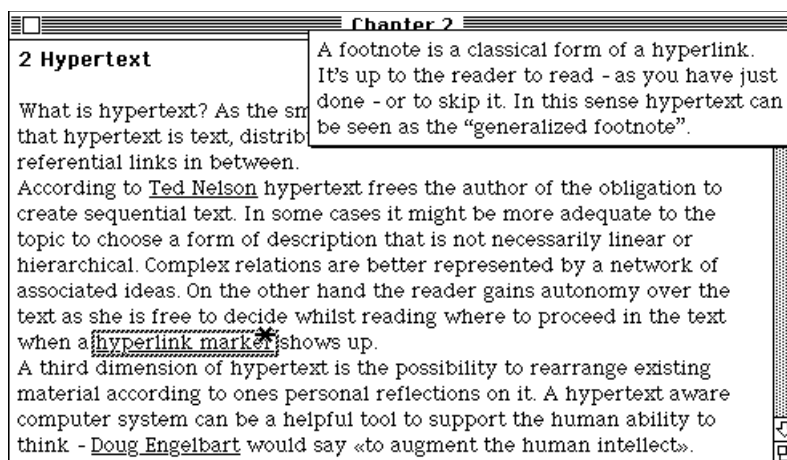


Fig. 2.10 A typical Guide window. The mouse button is currently pressed to display a pop-up note.

Jakob Nielsen compares replacements with NLS' feature to show and hide statements of a lower hierarchical level [Nielsen 90, p. 91]. And in fact Guide can mimic this behavior with replacement links. But the effect is more general and is a variation of Ted Nelson's concept of Stretchtext. Nelson gives the following example [Nelson 74, p. DM 19]:

Stretchtext is a form of writing. It is read from a screen. The user controls it with throttles. It gets longer and shorter on demand.

would elastically expand to:

Stretchtext, a kind of hypertext is basically a form of writing closely related to other prose. It is read by a user or a student from a computer display screen. The user, or student, controls it, and causes it to change, with throttles connected to the computer. Stretchtext gets longer, by adding words and phrases, or shorter, by subtracting words and phrases, on demand.

Ted Nelson promotes Stretchtext as a form of hypertext with less chance to get lost.

2.1.10 HyperCard

Apple HyperCard for Macintosh PCs is based on the index cards metaphor. But in contrast to Xerox PARC's NoteCards, where a window corresponds to each card, the cards are always organized in **stacks**. As a consequence windows play a marginal role in HyperCard. Just one main window shows the front most card of a stack. All cards have the same fixed size to fit on the original 9 inch Macintosh screen.

¹² This description follows Jakob Nielsen [Nielsen 90, p. 91]. It is in conflict with the definition given by Jeff Conklin. He says that replacement links completely swap the content of the entire window while the standard case of reference jumps opens new windows (in *Hypertext: An Introduction and Survey* [Conklin 87, p. 32]).

The stack provides a standard order for the cards and even without any hyperlinks the user is able to flip through the cards of the stacks. One card has a special status. It serves as the home for the stack. The home card is accessible from any other card and acts as a landmark for orientation. It is also possible to show a gallery with thumbnails of all cards to gain an overview.

Cards are made up of three layers. The background layer is the canvas for the cards. It contains basic artwork that is common for the cards of a stack. The foreground layer contains the text and individual illustrations. The buttons layer is on top of the foreground layer. It can contain active regions that trigger scripts written in HyperTalk. This layer model causes serious problems. Whenever the text is relayouted the buttons need to be aligned again to cover the underlying text. Link marker and button are not connected with each other.

In spite of all shortcomings HyperCard was very successful. Reasons for that are ease of use, availability – it was bundled for free with every Macintosh – and the possibility to share stacks with friends and in online communities.

2.1.11 Storyspace

Michael Joyce wrote the first electronic hypertext novel *Afternoon, a story* with Storyspace in 1990 [Joyce 92]. It is composed of 539 nodes and more than 900 links. *Afternoon* is an established classic in the new literary genre of hypertext fiction. Michael Joyce is also, together with Mark Bernstein, co-developer of Storyspace. The discussion in this section is based on his paper *Storyspace as a hypertext system for writers and readers of varying ability* [Joyce 91]. Another source is the user manual *Getting Started with Storyspace for Macintosh 1.5* [Bolter et al. 96].

A hypertext in Storyspace consists of a network of **writing spaces**. A writing space has a title, a text space and furthermore a topographic space. For the main writing space just this last mentioned space is used to spatially arrange the nodes of the hypertext that are for their part again writing spaces. The enfolded writing spaces offer the same functionality as the top level space. This leads to a recursive segmentation of the entire hypertext graph. Nonetheless this approach does not constrain the graph to a tree structure, because nodes inside a writing space can link to any other writing space, no matter whether it resides in the same writing space or not. It is rather motivated by the same idea as Frank Halasz' composites are (cf. page 23). It offers an additional dimension to classify the nodes of an hypertext. It supports the user in creating different levels of abstraction of the topic and in keeping the structures handy. In accordance with Halasz it is also possible to refer to an entire writing space by hyperlinking.

If also the starting point of a link is a writing space the link is called basic. These basic links have no representation in the text space. A special button in the tool bar – the Navigate tool – is needed to follow basic links.

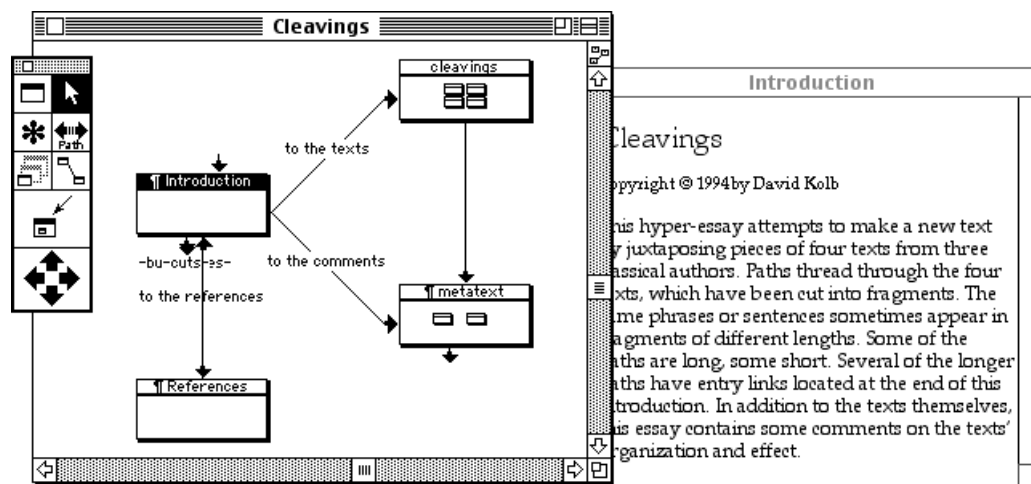


Fig. 2.11 The main writing space for the Storyspace hyperdocument "Cleavings" contains four writing spaces. Two of them contain writing spaces on themselves. The second window shows the text space of the "Introduction" node.

The other component of writing spaces is the text space. It contains a standard text editor with basic layout capabilities. Images can be placed into the flow of text. Text links are created for the current selection with a menu command. This provokes a line to be drawn between the origin of the link and the position of the mouse cursor until the user clicks a target [Müller-Prove 99].

Links are not highlighted. They do not distinguish from the rest of the text until the reader presses the OPTION-CMD keys. Then the links get surrounded by a solid frame and can be clicked to jump to the target writing space. In *Afternoon*, where nearly every word is a hyperlink, a double click is used instead of the modifier keys.

A peculiarity in Storyspace is that links can point to several targets at the same time. On activating such a link marker a modal dialog presents the descriptions for all links in question. It is also possible to attach conditions to links. For example a condition can be set up to only activate the link if a specific node has been visited before.

2.1.12 Intermedia

Intermedia is a prototype under Apple's version of Unix A/UX. It looks like a set of application programs that can be launched by the Macintosh Finder. In fact it is just one program that simulates the desktop environment with folder and document icons and implements application-like components instead of stand-alone application programs. The suite of pseudo-applications of Intermedia contains InterWord for text processing, InterMail for electronic mail, InterDraw for graphics editing, InterPlay and InterVideo for animations and movies, and InterVal for scheduling events. All these applications share access to a common link database and can exchange linking information about documents related to their associated file types.

Intermedia is a model for an operating system service of link management. The authors of Intermedia – Nicole Yankelovich, Norman Meyrowitz, Paul Kahn and Bernard Haan (among others) – aim for an integration, «where linking would be available for participating applications in much the same way that copying to and pasting from the clipboard facility is supported in the Macintosh and Microsoft Windows environments», *IRIS Hypermedia Services* [Haan et al. 92, p. 38]. Also the interaction of link creation follows the copy & paste pattern. Yankelovich writes in *Intermedia: The Concept and the Construction of a Seamless Information Environment* [Yankelovich et al. 88, p. 82]:

If links are to be made frequently, they must be a seamless part of the user interface. In any document, users can specify a selection region and choose the Start Link command from the menu. In any other document, regardless of type, users can define another selection region and choose one of the Complete Link commands.

Consequently, like the standard ‘Edit’ menu an ‘Intermedia’ menu is available in all application programs.

Link data is stored separately from the user’s documents in a database. It is segmented into **webs**, that contain a collection of links. For the user a web represents a network of references from the perspective of a specific context [Haan et al. 92, p. 43]. Webs can be individually activated one at a time (although it had turned out that the limitation to one open web is too restrictive). This provokes a different set of connections to be displayed between the documents.

The separation between links and content is a necessary condition for this quality (cf. FRESS on page 18). Frank Halasz points out that alternative named versions of webs can also be utilized to achieve a versioning of the link structure (cf. page 23 and [Halasz 87, p. 361]).

Intermedia’s approach will be continued by Microcosm and leads to the conceptual framework of Open Hypermedia Systems (OHS) (cf. 2.2.3).

2.1.13 Microcosm

Wendy Hall recalls the motivation that led to the development of Microcosm at the University of Southampton as follows [Gillies/Cailliau 2000, p. 128]:

I’d got the challenge from our activists, who said “We’ve got all this stuff about Lord Mountbatten and we want to be able to link it all together.”

Whatever system I was using, whether it was Word or a database or a spreadsheet or whatever, I wanted links that went across those processes, across applications. So I thought of links as being separate entities that you could apply.

What if someone’s written an essay or a criticism, or there’s a textbook about Mountbatten? We want to link to that as well, you know. Those were the problems I was trying to solve.

In contrast to the ideal laboratory situation of the Intermedia team, Wendy Hall has neither influence on the file formats, nor on the application programs. She has to cope with a real life situation of heterogeneous data formats. The approach she took, like Intermedia's, is based on the separation between content and linking information. The link data is stored in link databases, called **linkbases** for short, and can be overlaid on arbitrary document types [Lowe/Hall 99, p. 333].

Microcosm supports three kinds of links with fixed destinations. They are defined by Hugh Davis in *Towards an Integrated Information Environment with Open Hypermedia Applications* as follows [Davis et al. 92, p. 184]:

The *specific link* is a link from a particular object at a *specific* point in a source document that connects to a particular object in a destination document.

The *local link* is a link from a particular object at *any* point in a *specific* document that connects to a particular object in a destination document.

The *generic link* is a link from a particular object at *any* position in *any* document that connects to a particular object in a destination document.

A local link can connect entire documents with each other. This is similar to basic links in Storyspace (cf. page 28). Generic links are a generalization of Hyperties' exclusive linking mechanism. Wherever a dedicated marker shows up a link to the corresponding destination is created.

Microcosm provides also functions for **dynamic linking**. Dynamic hyperlinks are links without fixed source or destination. The target is computed based on text retrieval algorithms like GREP, or is based on proximity between vocabulary of the two documents. Gillies and Cailliau continue with Halls scenario [Gillies/Cailliau 2000, p. 129]:

[A] request to Microcosm to find links about [Mountbatten's Burma] campaigns could take you to the precise page [...]. And if you try the same request some time later, after someone has added a map of the Burma campaigns, dynamic linking means that Microcosm will now find that too.

How is it possible for Microcosm to implement this set of features for standard applications on standard operating systems? «Microcosm is best understood as a set of autonomous communicating processes which supplement the facilities provided by the operating system» [Davis et al. 92, p. 183]. A protocol is defined that uses the Dynamic Data Exchange service on Windows (DDE), Apple Events on Macintosh, and Sockets on Unix. Fully aware Microcosm applications use the protocol to pass information about the documents to Microcosm's linkbases, and to receive commands to highlight special passages if a document is requested as a link destination. Partially aware applications implement at least the menu for Microcosm, e.g. Microsoft Word can be extended to be partially aware by a special plug-in.

It is a problem for Microcosm, if a document is edited by an applications program that is not aware of the linking facilities of the system. Links might get out of date. It is also unfavorable if documents are moved or renamed in the file hierarchy.

Microcosm follows the Open Hypermedia Model that will be presented in section 2.2.3 Open Hypermedia Systems (p. 43).

2.1.14 World Wide Web

«Vague but exciting...», commented Mike Sendall[†] on Tim Berners-Lee's proposal of the World Wide Web in 1989. Sendall, then head of CERN's on-line computing group, decided to support the project.

Tim Berners-Lee and Robert Cailliau present the key concepts of the Web in *The World-Wide Web* for the Communications of the ACM [Berners-Lee/Cailliau et al. 94]. They introduce an universal address system, a network protocol for Web servers, and a markup language.

The address system defines Universal Resource Identifiers (URI). They should be unique and globally persistent for each object they refer to. URIs are concatenated strings of network protocol, server name and parameters to identify the object. For example

```
http://www.w3.org/History/1989/proposal.html
```

is the URI to the file 'proposal.html' on the Web server 'www.w3.org'. Slashes represent a hierarchy space in the file directory structure of the server. It is possible to attach further information to the URI to specify a predefined anchor position.

URIs do not necessarily have to identify a specific file. They can also depict a query to a server. The result is recalculated each time the URI is used.

```
http://www.google.com/search?q=MueLLer-Prove
```

is an example to show how the '?' is used to separate the query from the server address. The network protocol defined for Web servers is the Hypertext Transfer Protocol (HTTP). Other protocols for valid URIs might be FTP or NNTP for news. But HTTP offers some features not otherwise available. It is «a protocol for transferring information with the efficiency necessary for making hypertext jumps» [Ibid., p. 78]. HTTP is original stateless, which means, that the TCP connection is closed after each transmission.¹³ GET and POST methods are implemented that allow to transfer files from the Web server and to upload new files to the server.

Web pages are encoded with the Hypertext Markup Language. HTML is a formal language with a conforming SGML Document Type Definition (DTD). Special HTML elements are used to tag headlines, lists, tables, etc. Media files like images can be

¹³ States are introduced for HTTP 1.1.

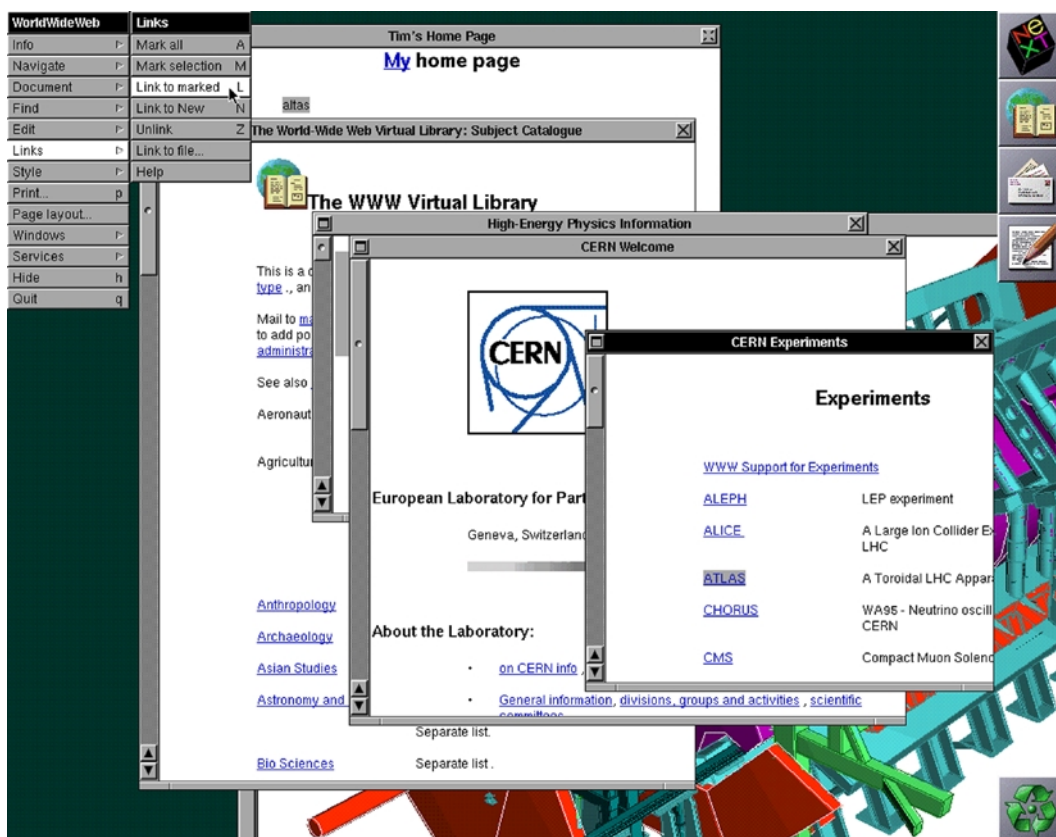


Fig. 2.12 *The Web browser WorldWideWeb on NeXTStep as it looked in 1993. In contrast with the original version from 1990 this version is capable of displaying images within the text flow. The first version was opening separate windows for images.*

included and hyperlinks can be attached to pieces of text. The markup for a link would be:

```
<a href="http://www.mprove.de">Matthias' Home Page</a>
```

The text “Matthias’ Home Page” is the visible marker for a link to the URI ‘http://www.mprove.de’.

A Web client is used to display HTML pages. The first is *The WorldWideWeb browser* by Tim Berners-Lee [Berners-Lee 93]. Not all of the features implemented in the early 1990s have survived the evolution to the present browsers Netscape Communicator and Microsoft Internet Explorer. The next paragraphs will outline the capabilities of the browser WorldWideWeb. In order to avoid confusion with the abstract information space of the World Wide Web the browser program was later renamed to Nexus.

Still today links are typically blue and underlined. But unlike today URIs are not displayed by WorldWideWeb/Nexus. They are considered as too technical and disgusting to be revealed to the average user [Gillies/Cailliau 2000, p. 206]. If a link is followed the resulting page opens in a new window. Initially also images are displayed

in separate windows. This has the advantage that they stay visible whilst the user scrolls through the text.

The Navigate menu has BACK and NEXT and PREVIOUS commands. The last two should not be mixed up with the FORWARD command of today's browsers. They do not reverse the BACK operation – they mean to go back a step and then take the next or previous link from the same page [Berners-Lee 93]. NEXT and PREVIOUS commands make it possible to leaf through the Web like a book page by page with a single keystroke. In *Hypertext in the Web - a History* [Cailliau/Ashman 99], Robert Cailliau and Helen Ashman compare this feature with Memex's trails. A Web page stores a sequence of arbitrary references to other pages, that can be displayed in succession with ease.

Tim Berners-Lee's browser offers no bookmark capabilities. It has a different notion of 'home page' instead. A home page used to be a private HTML page where references can be stored. URIs are captured for later use with COPY & PASTE from any Web page currently displayed to the personal home page. Our current understanding of a home page was known as a 'welcome page' back then.

WorldWideWeb/Nexus is an integrated environment for browsing and editing Web pages following the WYSIWYG paradigm. It is as easy to read pages as to write them. For example just one command with the shortcut CMD-SHIFT-N is needed to create a new HTML page and link to it from the previous text selection. The user is prompted to specify a new URI and the page content is sent to the server with the HTTP POST method.

Especially the last presented feature is a missing quality of all commercial browsers today. Just Amaya is still a combination of Web browser and Web editor. Amaya is being developed by the World Wide Web Consortium W₃C as an Open Source project.

Recent research activities at W₃C strives to separate content, structure and style of Web pages from each other. Cascading Style Sheets (CSS) and the Extensible Markup Language (XML) are two specifications that lead into this direction. Also linking information should be stored outside of the Web pages in separate files. XPointer is the formalism developed by W₃C for this purpose.

2.1.15 Hyper-G/HyperWave

According to Hermann Maurer, Frank Kappe, and Keith Andrews, the World Wide Web belongs together with WAIS and Gopher to the first generation of information systems on the Internet. The research team around Maurer argues, that Hyper-G has qualities that classify it for a second generation system. In *The Hyper-G Network Information System* [Andrews/Kappe/Maurer 95] they uncover the top six shortcomings of the Web. Hyper-G will be presented on the basis of these points.

First on *hyperlinking* [Ibid., p. 2]:

[The Web]¹⁴ does not provide any information structuring facilities beyond hyperlinks; its links are one-way (there is no way of determining which other documents refer to a particular document, leading to inconsistencies when documents are moved or deleted – the frequent “dangling links”) and embedded within text documents (there are no links from other kinds of documents).

Hyperlinks in HTML are absolute addresses to pages on other Web servers. There is no automatism to update these links if the destination page changes in name or location. This leads to “broken links” and annoying “HTTP error 404 – file not found” messages. Hyper-G automatically maintains the consistency of links. Therefore Hyper-G servers exchange linking information between sites. Much like Microcosm a database is used on each Hyper-G server to store linking data separate from the documents.

On *searching* [Ibid.]:

Like Gopher, [the Web] has no native search facilities, but relies on external search engines such as WAIS, leading to patchy server-by-server provision of search facilities by individual sites and no real-time cross-server searches (searches in previously generated cross-server indices are available).

Google and Yahoo have been mentioned above. But services like these can neither cope with millions of Web pages nor guarantee that they are all-embracing for a specific server. The user is left in uncertainty whether a negative result to a search means that there are really no pages that match the text query. Hyper-G in contrast has sophisti-

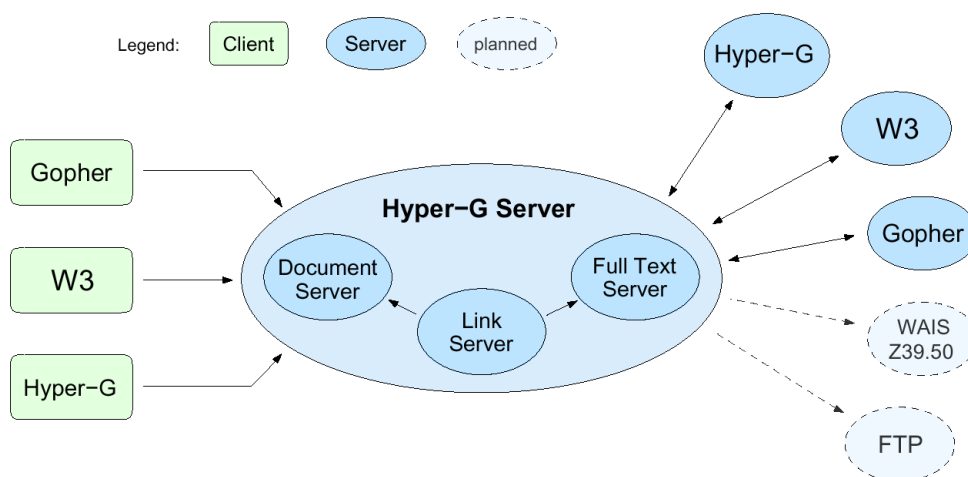


Fig. 2.13 The architecture of Hyper-G. Web- and the Hyper-G clients have Internet access to a Hyper-G server. The server itself exchanges information with other Hyper-G servers. In addition it can also receive data from Web servers and other resources.

¹⁴ Actually “W3” is used here. I have changed it to “the Web” for better readability and because of consistency.

cated search capabilities built in. The Hyper-G protocol allows to initiate searching processes on several Hyper-G servers simultaneously. The results can be presented in best suited layout depending on the preferences of the user. This leads to consistency in the user interface no matter where the search is performed.

On *dynamic content* [Ibid.]:

The flexibility provided by CGI is achieved at great cost: the uniformity of the interface disappears, different [Web] servers behave differently – resulting in the “Balkanisation” (to quote Ted Nelson) of the Web into independent “W3 Empires”.

The syntax for URIs allows to specify queries instead of just fixed pages. The ‘?’-example on page 32 has shown how extra information can be transmitted to the server. The protocol used here is the Common Gateway Interface (CGI). A CGI script can instruct the Web server to start an arbitrary application program to calculate something, for example to connect to an external database to retrieve some data. Finally the results are presented on an HTML page.

This point is related to the next on *large structures* [Ibid.]:

Also, there is little support for the maintenance of large datasets, so it is not uncommon to see several [Web] servers within a single organisation, each a fundamentally separate interactive context.

Hyper-G provides an architecture that allows to define structure independent from the distribution of the files among the servers. Files can belong to **collections**, which may themselves be part of other collections. Each file is member of at least one collection, but it can be associated with several others at the same time. A special kind of collection is a **cluster**. It is used to group a set of files as a logical unit. It can be used e.g. to refer to a document and its translated versions as one entity or to form multimedia aggregates like movies with an associated text transcription file. Collections are well integrated into the data model. This means especially that they can also be the target object for hyperlinks in Hyper-G.

Barry Fenn and Hermann Maurer add **tours** as another form of aggregation. A tour is «a scripted sequence of links, activated between a series of documents or clusters, which the user may just sit back and watch like a television program» (in *Harmony... on an Expanding Net* [Fenn/Maurer 94, p. 35]). The user can stop at any time and continue to explore the content on her own behalf. Tours evolve to a specialization of collections and are called **sequences** then [Maurer et al. 98, p. 284]. They are collections with a sort order imposed on its members. Navigation through sequences can use the order of items to offer commands like NEXT, PREVIOUS, FIRST and LAST. Note that sequences are orthogonal to link structures. They also match Memex’ notion of trails much better than the concept of hyperlinking does.

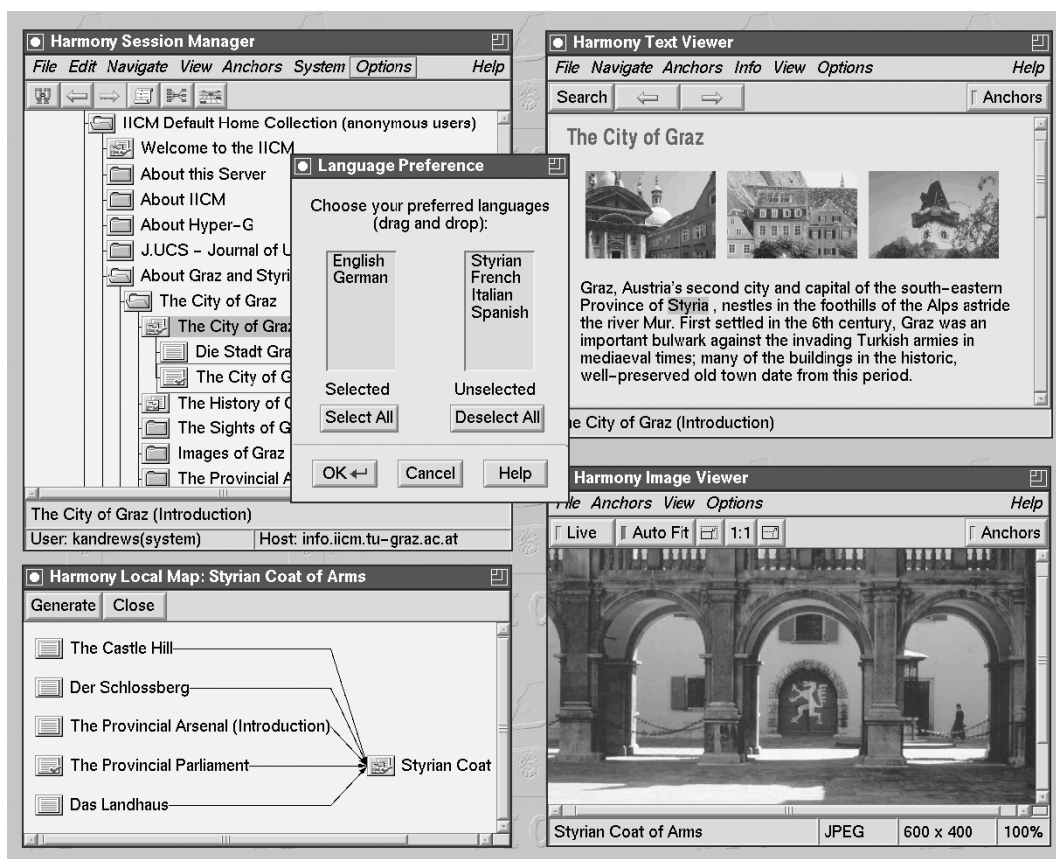


Fig. 2.14 *Harmony, the Hyper-G browser and editor for X Windows. The Session Manager (top left) shows the hierarchy of collections. The Text Viewer (top right) uses an SGML parser to display HTML or HTF documents. Images can be displayed inline or separate with the Image Viewer (bottom right). The Local Map (bottom left) shows on request the incoming and outgoing links for a given document.*

On *authoring* [Andrews/Kappe/Maurer 95, p. 2]:

The Web today is very much “read-only”, in the sense that information providers prepare data sets in which information consumers can generally only browse.

The client application program for Hyper-G is Harmony. Fig. 2.14 shows the Session Manager, the Text Viewer, the Image Viewer and the Local Map window. Hyper-G has its own native markup format for hypertext files called HTF. It is, like HTML, an application of SGML, and the Text Viewer can display both formats. Moreover audio files, MPEG movies, 3D scenes, and PostScript files are directly supported by Harmony.

Unlike the dominating Web browsers Microsoft Internet Explorer and Netscape Communicator the Hyper-G client Harmony is also an authoring tool. Links can be created and followed for all documents of file formats just mentioned. Images, movies and 3D scenes can contain links in Hyper-G.

Instead of HTTP the Harmony Document Viewer Protocol (DVP) is used between Hyper-G servers and clients. This is necessary to provide the power for various browsing, editing, and link functions.

On *scalability* [Ibid.]:

Finally, although its URL mechanism endows [the Web] with scalability in terms of number of servers, it is not scalable in terms of number of users. Extremely popular [Web] servers [...] can often become overwhelmed by tens of thousands of users, necessitating their physical mirroring to many alternative sites.

Replication mechanisms and caching is built into the architecture of Hyper-G. The servers can adapt to variable user traffic and are capable to reroute the load to less used servers.

2.2 Theory of Hypertext

Many different hypertext systems have been presented in the previous section. And each of them introduced new concepts. Even more confusing, each system comes along with its own terminology and might have a slightly different understanding of already existing concepts. For example a hypertext node is called (in alphabetical order) article, card, compound text, document, file, hyperdocument, node, page, parallel document, windowing text and writing space. The same variety of terms can be found for hyperlinks. A clear terminology is necessary to compare the presented hypertext systems with each other and to recognize significant variations.

A couple of approaches have been taken to classify the species of hypertext systems. First is Jeff Conklin 1987 with *Hypertext: An Introduction and Survey* [Conklin 87]. He presents a table that compares several Hypertext systems against a possible set of features.

The second approach is *The Dexter Hypertext Reference Model* [Halasz/Schwartz 94]. It grew out of a series of small workshops on hypertext between 1988 and 1990. The initial workshop was held at Dexters Inn in New Hampshire, hence the name for the model. Among the participants were Doug Engelbart (NLS/Augment), Frank Halasz and Randall Trigg (NoteCards), Janet Walker (Symbolics), Catherine Plaisant (Hyperties), and Norman Meyrowitz (Intermedia).

The most recent theory is on Open Hypermedia Systems. Intermedia, Hyper-G and especially Microcosm belong into this category.

2.2.1 Hypertext Feature Matrix

This section builds on top of Jeff Conklin's record and extends it for the hypertext systems presented in this discussion. Conklin's feature dimension consists of [Conklin 87, p. 21],¹⁵

	Memex	Xanadu	NLS/Augment	HES	FRESS	Smalltalk	NoteCards	Symbolics	Hyperties	Guide	HyperCard	Storyspace	Intermedia	Microcosm	World Wide Web	Hyper-G
Hierarchy		○	●	●	●	●	●	●	○		①	●	●		○	●
Graph-based	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Link Types	○	●	●		●		●	○	○		○	●	●		○	
Attributes	○	●	●				②	○	○		○		●		○	
Paths	●	●	●				○	●	○	○	①		○		○	●
Versions		●	●				○	○	○		○	○	③		○	
Procedural Attachment	○	○	●			●	●	○	○	●	●		○	○	●	○
String Search	○	○	●				●	●	○		●		●		○	●
Text Editor		any	custom			Smalltalk	InterLisp	Concordia	basic	basic	self	self	custom	custom	custom	Harmony
Concurrent Multiusers	○	○	●			○	●	○	○	○	○	○	●		●	●
Graphical Browser	○	○	○			○	●	●	○		○	●	●	●	○	●
Structured Nodes	○	●	●		●		○	○	○	○	○	○	●	●	●	●
Separation of content and links		●	○		●		○	○	④		○	○	●	●	○	●
Windows	⑤	⑤	⑤			●	●	○	○	●	○	●	●	●	⑥	●

Tab. 2.1 Matrix of Hypertext systems and their features. ('●' is used if a hypertext system has a feature; '○' if not. Empty entries indicate an unknown or not applicable quality.)

- ① A linear order is given by the Stack structure.
- ② Just the nodes in NoteCards can have attributes.
- ③ Intermedia's webs can be utilized for version control of the link structure.
- ④ Hyperties has implicit links. Keywords refer to nodes, anywhere they appear.
- ⑤ Memex uses at least two projection screens, Xanadu has two parallel windows, NLS initially has one screen that can be split horizontally into windows.
- ⑥ The standard implemented by Netscape Navigator and Microsoft Internet Explorer is a browsing mode. The next page completely replaces the previous page in the very same window.

¹⁵ The question whether the hypertext system supports images will be skipped.

Hierarchy: Is there specific support for hierarchical structures?

Graph-based: Does the system support nonhierarchical (cross-reference) links?

Link Types: Can links have types?

Attributes: Can user-designated attribute/value pairs be associated with nodes or links?

Paths: Can many links be strung together into a single persistent object?

Versions: Can nodes or links have more than a single version?

Procedural Attachment: Can arbitrary executable procedures be attached to events (such as mousing) at nodes or links?

String Search: Can the hyperdocument be searched for strings (including keywords)?

Text Editor: What editor is used to create and modify the content of the nodes?

Concurrent Multiusers: Can several users edit the hyperdocument at the same time?

Graphical Browser: Is there a browser which graphically presents the nodes and links in the hyperdocument?

These categories will be extended by the following questions:

Structured Nodes: Can links refer to specific location within the nodes?

Separation of Content and Links: Is the linking information stored separately from the documents?

Windows: Do nodes open in new windows?

Jeff Conklin could only take into account existing hypertext systems up to 1987. From his table the matrix in Tab. 2.1 takes information about Xanadu, NLS/Augment, NoteCards, Symbolics, Hyperties, and Intermedia. The list will be complemented by Memex, HES, FRESS, Smalltalk, Guide, HyperCard, Storyspace, Microcosm, Hyper-G, and the World Wide Web. Tab. 2.1 shows the new matrix.

Just a few of Conklin's statements needed to be updated. Symbolics Document Examiner has a graphical browser as can be seen in Fig. 2.8 on page 25. The corresponding editor is Concordia instead of 'None'. The other update touches upon Intermedia's support for versioning. As we have seen the concept of webs can be utilized for versioning of the link structure.

2.2.2 The Dexter Hypertext Reference Model

Frank Halasz and Mayer Schwartz present the model in *The Dexter Hypertext Reference Model* [Halasz/Schwartz 94]. The model aims to deliver a common terminology that covers existing and future hypertext systems. The abstraction leads to a model with

three layers. The storage layer describes the network of nodes and links. The run-time layer covers the interaction of the user with the system. The third layer is the within-component layer. It describes the internal structure of hypertext nodes.

Dexter's response to the variety of different names for nodes and links is the introduction of a new term.

The basic entity of the storage layer is a *base component*. (1)

A node in the Dexter model is an **atomic component**. But the internal structure of components is purposefully not specified. This allows the forming of **composite components**, that can recursively contain other components. Composite components can be used to describe for example the writing spaces of Storyspace or the documents of Symbolics' hypertext system.

Every component has a *globally unique identifier* (UID). (2)

This means that a component can be unambiguously addressed not only from the hypertext where the component belongs to, but from any other hypertext. Some hypertext systems provide indirect specification of link destinations. NLS for example can jump to «the statement containing the word 'pollywog'» [Ibid., p. 33]. The storage layer is responsible to resolve such component specifications into UIDs.

The mechanism to address content within components is called **anchoring**.

An *anchor* is a pair of anchor ID and anchor value. (3)

The anchor ID is valid within the scope of a component and is used to identify a position within the component. The anchor value can be any arbitrary value and has only meaning to the owning application program of the component. It is used to specify the position within the component. If the content of the component is edited the application has to update the anchor values for the component accordingly. The anchor IDs stay fixed to provide a reliable address for incoming links. This distinction between anchor ID and anchor value is necessary to integrate diverse data files as nodes for the hypertext.

A globally unique component specifier together with an anchor ID forms one side of a hyperlink. Two more fields are added for the definition of a specifier in the Dexter model.

A *specifier* is a tuple of UID, anchor ID, direction and presentation specification. (4)

The direction value can be FROM, TO, BIDIRECT, or NONE. The common case of unidirectional text links is modeled in the way that the first specifier has the direction value

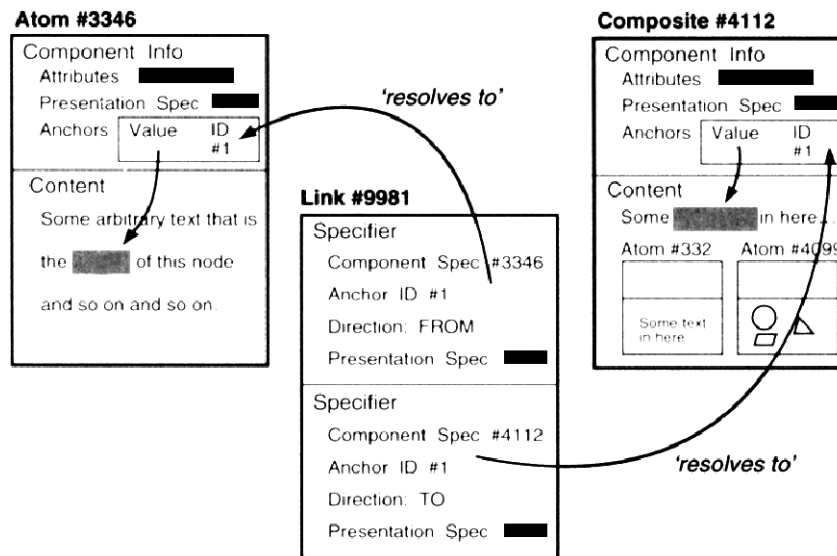


Fig. 2.15 This diagram shows the relations between 5 components. An atomic component to the left. The composite component to the right contains 2 atomic components. The link in the middle is the fifth component. (The component info block for the link component is omitted.)

set to FROM and the second specifier the direction value set to TO.

The purpose of the presentation specification is to store data about how to display the link marker for the user interface.

Now it is possible to define the link for the Dexter model.

A *link* is a component that contains a sequence of at least 2 specifiers. (5)

Concepts like trails, paths, tours or sequences can be represented by Dexter links with more than 2 specifiers.

Now the suite of components is complete.

A *component* is a pair of base component and component information. (6)

A *base component* is either atomic, composite or a link component. (7)

The component information stores the sequence of anchors for the related base component. Further on the component information contains attribute-value pairs that can be used to describe the base part of the component. For example meta-data like keywords for hyperlinks can be attached as special attributes of the component information section for link components. Third, data stored in the component information is a presentation specification that contains instructions for the run-time layer how to display the content of the component in the user interface.

The run-time layer manages the presentation of the hypertext structure for the user interface.

An *instantiation* is the activation of a component for the run-time layer. (8)

A *link marker* is an instantiated anchor. (9)

Halasz and Schwarz continue their tour through the Dexter Model with the presentation of general functions that are necessary to operate the hypertext model. They will not be presented here.

In order to conclude the Dexter Hypertext Reference Model two qualities shall be highlighted. The model is capable to represent higher order aggregations of nodes, as has been demanded by Frank Halasz in his “7 Issues” [Halasz 87], [Halasz 88], (cf. page 23). And second, it is possible to specify links to links. The model is therefore powerful enough to describe a hypertext that refers to itself.

2.2.3 Open Hypermedia Systems

The focus of Open Hypermedia Systems (OHS) is the integration of content files of arbitrary formats with hypertextual linking structures. The presented hypertext systems Intermedia, Microcosm and Hyper-G are designed with OHS principles in mind.

David Lowe and Wendy Hall define a hypermedia system as open if it conforms with the following four conditions, *Hypermedia & the Web: An Engineering Approach* [Lowe/Hall 99, p. 338]:

The hypertext link service should be available across the entire range of applications available on the desktop.

The hypertext service has to deal with application programs that are unaware of the link management. As a consequence of this linking data and content have to be separated, because the original data format cannot be extended to store information about links and anchors.

The link service must work across a network of heterogeneous platforms. [Ibid.]

Especially the Internet is a medium that should be utilized to connect Open Hypermedia Systems with each other. The tools and services should be available for all major operating systems.

The architecture should be such that the functionality of the system can be extended. [Ibid.]

This calls for a modular software architecture of the system with a well defined programming interface (API) for new extensions and plug-ins.

There should be no artificial distinction between author and reader. [Ibid.]

A hypertext system is considered as open, if the user is able to change and modify the linking structure on his own. This does not imply that a user is allowed to change every aspect of the document and to alter how the document appears for someone else. But

a user should have full access in her private domain. Furthermore the hypermedia system has to manage user lists and access rights in order to offer a service to share user changes with other users.

2.3 Provisions for the Future of the World Wide Web

This section outlines a scenario for the World Wide Web. How could the Web be if ideas of other existing hypertext systems are incorporated?

The outline is arranged along the three interrelated topics hypertext concepts, user interface of Web clients, and integration with the operating system.

2.3.1 Identification of Nodes

It is rather simple to keep hyperlinks valid if the hypertext is limited in size and a single author is editing at one terminal only. In this case the entire hypertext is stored on one machine and one program can update the links if any resource is moved or renamed. The global hypertext system World Wide Web is a different case. The data is distributed over millions of servers and no mechanism is reporting broken links to keep the Web free of hyperlink errors.

What can be done to get robust identification of hypertext nodes for the Web? Tim Berners-Lee's original concept talks about Universal Resource Identifiers (URI) instead of Uniform Resource Locators (URL) [Berners-Lee 99, p. 62]. URIs are meant to stay constant as long as the designated resource exists. But they are vulnerable against location change of the file in the server's file hierarchy. Hence the weaker concept of Uniform Resource Locators that contains the path to the resource that might change. URLs drop the quality to be persistent and universal in any means. This alone might be no problem, but the Web offers no compensating methods to regain stability in addressing Web pages.

Two different solutions have been proposed. The definition of really persistent identifiers like URIs, and the introduction of a functional layer that translates persistent IDs to real addresses. The first direction is taken by Xanadu. Xanadu introduces a global address scheme that guarantees for every character a unique and persistent address. «The central [...] secret this all relied on [is] the freezing of content addresses into permanent universal IDs» [Nelson 99a, p. 9]. Documents become sequences of pointers to the global address space. HES has actually implemented this data model.

The second approach has been taken by Microcosm and Hyper-G. Linking information is stored in link databases external to the files. The Hyper-G protocol allows to broadcast update information between Hyper-G servers world-wide in order to synchronize the data.

2.3.2 Groups of Nodes

The Web lacks power of expression to refer to a group of nodes. For good reason many hypertext systems have implemented means to deal with higher order entities of nodes. NoteCards has filebox and browser cards and Frank Halasz in his "7 Issues" calls for composites to augment the basic nodes and links model. Hyper-G has collections, clusters and sequences that are well integrated into the user interface of the Hyper-G client Harmony. The Dexter Hypertext Reference Model addresses the need by the definition of composite components.

The purpose of groups for the Web is manifold. For example it could change the implicit notion of Web sites into an explicit concept. The assumption that a Web site is equivalent to a set of files that reside on the same Web server does not hold under all conditions. Given the knowledge of the Site structure Web clients can offer navigational aids to the user.

During the recent years the W₃C has agreed on a formal language to encode semantic data. The Extensible Markup Language (XML) can be used to adopt the concept of aggregations of nodes for the Web.

2.3.3 General Hyperlinks

Hyperlinks can be either text links or basic links in the notion of Storyspace. That means that the starting point can be attached to a piece of content or the entire node serves as the starting point. Hyperlinks can either point to a node, or to a position or spawn within the node. Hyperlinks can be unidirectional or bidirectional. Furthermore, according to the Dexter Model links can connect more than just two nodes with each other.

The Web supports unidirectional text links only. But the W₃C has a working group to develop a generalization of links. XPointer is based on XML and incorporates the presented ideas. Also link types are possible with XPointer's syntax.

2.3.4 Browser

Vannevar Bush and Ted Nelson emphasize the importance of parallel visualization of documents. At least two independent areas on screen are necessary to successfully handle text online. Windows are developed by Doug Engelbart in the 1960s at SRI and by Alan Kay in the early 1970s at Xerox PARC. They offer a flexibility to the user that also bears an additional complexity for the interface. Windows usually overlap and cover each other to a high percentage, because the desktop metaphor mimics working with one sheet of paper. A designated secondary window is not part of the WIMP interface. It is on the user's behalf to arrange the windows to see the content of two windows side by side.

It is also Ted Nelson who suggests the use of animation for the interface. He argues [Nelson 74, p. DM 53]:

The text moves on the screen! [...] Note that we do *not* refer to here to jerky line-by-line jumps, but to smooth screen motion, which is essential in a high-performance system. If the text does not move, you can't tell where it came from.^[16]

This sort of behavior could support the orientation in hypertext. It would be obvious to distinguish between local links and links to a different page. It would also be obvious whether a local links points upwards or downwards in the current page.

Finally something about the relation between browser and Web page. Today an HTML page has no access to user interface elements outside the content of the browser window. But consider a Web site with control to the menu bar. It could provide information to the browser to add a standardized menu with landmark pages like GOTO HOMEPAGE OF THIS SITE, or SHOW IMPRINT.

Another effective point of control for a Web site is the cursor shape. The Guide example has illustrated how user interaction can take advantage of different mouse cursors (cf. Fig. 2.9 on page 26).

2.3.5 Integrated Browser/Editor Environment

Tim Berners-Lee recalls [Berners-Lee 99, p. 157],

I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse, but to create.

Consequently the first implementation of a Web client WorldWideWeb/Nexus was capable of browsing and editing HTML pages in WYSIWYG mode. The united approach got lost as the print publishing industry discovered the Web. NCSA, who developed the early Web browser Mosaic, showed no interest in building an editor for HTML. Netscape and Microsoft didn't shift the focus back to editing and «the Web became another consumer medium with many readers but relatively few publishers» [Gillies/Cailliau 2000, p. 243].

Nearly all hypertext systems presented in this chapter strive for an integrated environment between reading and writing. No artificial borders should hamper the user from editing and commenting existing content. Just Symbolics' hypertext system wilfully separates the tasks between the application programs Document Examiner and Concordia. But it became apparent that even in the context of online documentation annotation capabilities are desirable.

If the Web should be used to support creative knowledge workers, flexible and easy to use editor capabilities are necessary. A single application program has the advantage to

¹⁶ Undelining replaced by italics typeface.

offer a consistent user interface. Existing WYSIWYG Web authoring tools like Adobe GoLive can at least soothe the process of editing HTML markup source code.

The new WebDAV protocol, which stands for Web-Based Distributed Authoring & Versioning, offers many features that can improve the current generation of Web authoring. Consequent implementation of WebDAV can bring the user experience near to the original vision.

2.3.6 Separation between Content and Appearance

The Web requires pages to be HTML encoded. Files of different format – for example plain text or images – might be nodes of the Web, but cannot be used as starting points for hyperlinks. HTML files contain the content, linking information, structural information and definitions how to display the page in the browser.

The current approach taken to fight the overload of HTML files is the separation of content and appearance. A first step is the external definition of styles for HTML files using cascading style sheets (CSS). Just content and structural information remains in the HTML file. The next step is the abstract and formal representation of content and structure in XML encoded format. The extensible style language (XSL) and CSS is used to transform XML back to a form that can be displayed on screen. Linking structure is going to be encoded as XPointers; also a form based on the XML syntax.

The discussion of Open Hypermedia Systems has shown that a different approach is possible. The separation between content and linking structure has a lot of advantages for the OHS model. No markup is imposed on the files. Any file can contain links. And the system takes care for link consistency.

Ted Nelson goes even beyond. In *Embedded Markup Considered Harmful* [Nelson 97b] he argues against any inherent form of hierarchical structure for content. Any markup, if based on SGML, curtails the flow of thought. Text and markup for structure have to be kept separated. Markup for style forms a third layer. Nelson summarizes the three layers [Ibid.]:

A content layer to facilitate editing, content linking, and transclusion management.

A structure layer, declarable separately. Users should be able to specify entities, connections and co-presence logic, defined independently of appearance or size or contents; as well as overlay correspondence, links, transclusions, and “hoses” for movable content.

Finally, a *special-effects-and-primping layer* should allow the declaration of ever-so-many fonts, format blocks, fanfares, and whizbangs, and their assignment to what’s in the content and structure layers.

Nelson's model solves the following problem. A quote, like the three paragraphs above, is taken by copy & paste from journal's Web site. This breaks the software-based connection to the original text. You can look up the reference for '[Nelson 97b]' in the appendix, find in this case a URL, and with some luck the corresponding page on the Web still exists. Nelson's concept of transclusion would maintain the link to the cited piece of text all the time. Moreover the separate layers would make possible the integration of the quote into the new context. Appropriate formatting could also be assigned.

2.3.7 Integration of Hypertext facilities into the Operating System

Early hypertext systems like NLS exploit the resources of the mainframe computers in such a way that they interact with the hardware quite directly. But none of the hypertext systems thereafter has ever intruded the level of operating systems.¹⁷

Open Hypermedia Systems define the foundation for such a service. But to achieve a robust and consistent user interface, integrated into the whole environment, support by the operating system is inescapable. Renaming or moving of files has to induce the update operations to keep the link data consistent.

This chapter shall close with a quote from Tim Berners-Lee [Gillies/Cailliau 2000, p. 195]:

'Picture a scenario in which any note I write on my computer I can "publish" just by giving it a name. [...] In that note I can make references to any other article anywhere in the world in such a way that when reading my note you can click with your mouse and bring the referenced article up on your machine. Suppose, moreover, that everyone has this capability.' That was the original dream behind the Web.

¹⁷ Sun's Link Service provided an operating system level service for Sun workstations. It is described by Amy Pearl in *Sun's Link Service: A Protocol for Open Linking* [Pearl 89]. The objective behind the development during the late 1980s was «that if a link service was a standard feature of the operating environment, then all serious applications would be written to make use of this feature» [Davis et al. 92, p. 185].

3 Graphical User Interfaces

The evolution of human-computer interaction can be classified into four main stages. According to Andries van Dam in *Post-WIMP User Interfaces* [van Dam 97], respectively in *Post-WIMP User Interfaces: the Human Connection* [van Dam 2001], the first period reigns during the 1950s to the 1960s. Computers are operated in batch mode, and stacks of punched-cards are fed into card reading devices. The second period is the era of timesharing. It starts in the early 1960s and lasts until the early 1980s. The dominating interaction method is manual command line input on mainframes and mini-computers. In the early 1970s starts the third period that is still going on. Computers are raster-graphics-based networked workstations, microcomputers and PCs. They are equipped with a mouse as a graphical input device (GID). The graphical desktop metaphor with windows, menus and icons is the prevailing paradigm. The fourth generation has just begun. Van Dam calls it Post-WIMP. «These don't use menus, forms, or toolbars, but rely on, for example, gesture and speech recognition for operand and operation specification» [van Dam 97, p. 64]. Raj Reddy coined the term SILK interfaces in 1996. They utilize Speech, Image, and Language understanding, and are driven by Knowledge bases [Ibid., p. 67].

This chapter focuses on the third generation of user interfaces. On graphical user interfaces and their history. At a symposium to honor Doug Engelbart in 1998 Ted Nelson has pointed out that 'GUI' does not depict one single user interface. He said, «there are so many millions of graphical user interfaces possible and yet we are stuck with one in which we have a single fixed little area called the desktop», *The Unfinished Revolution and Xanadu* [Nelson 99b, p. 4].

Investigating the history of graphical user interfaces might reveal some insights about original objectives that are forgotten nowadays.

3.1 History

Remarkably few books and articles cover the history of graphical user interfaces. Many more are success stories of companies, or biographies of their CEOs. A few salutary exceptions to this are *A brief history of human-computer interaction technology* by Brad Myers [Myers 98], *A History of Modern Computing* by Paul Ceruzzi [Ceruzzi 98], *Dealers of Lightning – Xerox PARC and the Dawn of the Computer Age* by Michael Hiltzig [Hiltzig 99], and *Der Computer als Werkzeug und Medium* by Michael Friedewald [Friedewald 99]. Many details for the following sections are taken from these resources.

The advent of Apple Macintosh in 1984 has initiated the desktop publishing revolution. Computer based type setting using the paradigm of WYSIWYG – What you see is what you get – and laser printers for high quality paper output made it possible for everyone to create professional looking documents. The Macintosh introduced the mouse as a standard input device to double-click folder and document icons, to arrange windows and to pull down menus.

The entire scenery on screen follows the so-called desktop metaphor. All objects are designed to fit into the virtual world of an office. The desktop world literally covers the engineering aspects of the computer. This is to create a familiar and friendly environment for people who are not experts in computer hardware and operating system design. The user deals with documents instead of files. Directories are called folders and look and behave similar to their physical counterparts. They can be opened and documents can be filed into. And most prominently an icon of a trash can is used to delete objects. Document icons are dragged onto the trash can icon where they stay until the trash gets emptied.

The Macintosh was the first computer with a graphical user interface that was commercially successful. But the development starts long before Apple Computer was even founded. This section presents a trip through the history of the modern user interface for personal computers.

The journey starts in 1960. Joseph Licklider (*1915 †1990) wrote the article *Man-Computer Symbiosis* [Licklider 60] in which he proposes interactive computing as a new paradigm to make use of the computer. Two years later he became the first director of the Information Processing Techniques Office (IPTO) at ARPA. IPTO's objective was to devise new utilization of computers other than plain computation. Especially the military was looking for computer systems that support decision processes with short response times. To Licklider this was pretty much the same as his vision of **Man-Computer Symbiosis** (cf. 3.1.1). During the 1960s IPTO funded several research projects to develop time-sharing computer systems and information processing projects.

MIT Lincoln Laboratory was one of the institutes that was supported for their research work on interactive computer graphics. Ivan Sutherland presented 1963 in his Ph.D. thesis *Sketchpad: A Man-Machine Graphical Communication System* [Sutherland 63a] a working program to interactively edit vector based illustrations with a light pen directly on screen. Also the concept of a window was first used in **Sketchpad** (cf. 3.1.2). The user can zoom into a drawing area and all graphical elements are clipped against the edges of the screen. Sutherland's ground breaking work is the starting signal to develop interactive user interfaces with graphical aspects for the decades to come.

The Augmentation Research Center at Stanford Research Institute (SRI-ARC) has already been mentioned in the previous chapter about hypertext. Now the system NLS (cf. 3.1.3) will be examined from the perspective of interaction techniques. Doug Engelbart's long term objective is the augmentation of human intellect. His fundamental research lead especially to the invention of the mouse in 1963. Windows, interactive text editing, the five-finger chording keyset input device, and video conferencing are other development projects at SRI.

Xerox Inc. grew by the commercial success of the Xerox copier machines in the 1960s. In 1970 they founded a new research laboratory. The mission for Xerox Palo Alto Research Center (PARC) was to explore the opportunities of new computer systems for office appliance. With this step Xerox tried to be prepared for the dawning age of personal computing. Alan Kay has formulated this idea as, «The best way to predict the future is to invent it» (e.g. in [Frenkel 94, p. 22]).

One of the things that was developed at Xerox PARC was the Laser Printer with the two components Scanning Laser Output Terminal (SLOT) by Gary Starkweather, and the Research Character Generator (RCG) by Butler Lampson and Ron Rider. SLOT is a technique that uses a laser beam to transfer an image to the xerographic drum, while RCG is a way to create a bitmap of text in computer memory.

The **Alto computer** (cf. 3.1.5) is the ancestor of modern PCs. It was principally designed by Butler Lampson and Chuck Thacker and had a 72 DPI bit-mapped graphic display and a mouse. Ethernet was invented by Bob Metcalf in cooperation with David Boggs, Chuck Thacker, Butler Lampson and others in 1973. This new technology made it possible for PARC to connect all the Altos to the first local area network (LAN). All these parts together were the ingredients to EARS, which stands for Ethernet-Alto-RCG-SLOT. When EARS became operational in Autumn 1974, the system started immediately to attract people at Xerox PARC. Everyone wanted to have an Alto of her own. The prototype for an office of the next decade was established by the word processing applications program Bravo by Charles Simonyi, the newly developed paradigm of WYSIWYG, and the possibility of printing documents in high quality to the shared SLOT terminal. About 50 units of the Alto I and more than 1,500 units of the Alto II were produced until 1980 [Friedewald 99, p. 275].

The path towards personal computing was not taken by chance. Alan Kay directed the Learning Research Group (LRG) at PARC. His master and doctoral thesis at the University of Utah about the **Flex Machine** (cf. 3.1.4) became the research program for the group. Many aspects of the Alto computer are directly influenced by Kay's vision of the **Dynabook** (cf. 3.1.4), *A Personal Computer for Children of all Ages* [Kay 72a]. Alan Kay and Adele Goldberg took the Alto computer as an Interim Dynabook into the classroom. The article *Personal Dynamic Media* [Kay/Goldberg 77] shows that school kids were able to program the computer with the newly developed object-oriented program-

ming language **Smalltalk** (cf. 3.1.5). The work with children revealed many interesting insights into the field of human-computer interaction.

The Alto had always been an experimental prototype. The project of a commercial system that builds on top of all the research results of PARC started in 1975 and led to the development of the Xerox 8010 Information System. The **Xerox Star** (cf. 3.1.6), as this system was called for short, was presented in 1981. It introduced a consistent graphical user interface that covers the operational tasks of the computer with the notion of an office environment. The desktop metaphor was born, although it was originally called the physical-office metaphor.

While Xerox PARC is located in California the Architecture Machine Group was working at MIT in Massachusetts. It was formed in 1967 by Nicholas Negroponte and got financial support from the Cybernetics Technology Office (CTO), another division at DARPA.

In the mid-1960s Negroponte was a student of architecture. His dream was to have a machine to support the creative design process of architects. He was convinced, that for such a device to be truly helpful, it would have to be intensely interactive with the human user, *The Media Lab: Inventing the Future at M.I.T.* [Brand 87, p. 137]. This explains the heritage of the name of the Architecture Machine Group, and also the devotion for human-computer interaction.

Throughout the 1970s, the Architecture Machine Group explored chances of a computer-based medium. The Aspen Movie Map for example is a video disk application, that offers a virtual tour through the city of Aspen, Colorado. The user is free to move in any direction and even to change the season at any point of the session [Nielsen 95, p. 40]. Other projects use speech or holographic images. But especially Richard Bolt's work on the **Spatial Data Management System** (cf. 3.1.7) is of interest for the history of graphical user interfaces. One of SDMS's components is called Dataland. It is the projection of data to a two-dimensional field. The user can move and zoom into the Dataland to enlarge thumbnails of documents until the content becomes accessible.

It is often rumored, that Apple Computer, Inc. took initial inspiration for the **Lisa** (cf. 3.1.8) and **Macintosh** (cf. 3.1.9) projects from a visit at Xerox PARC. This visit did actually happen in November 1979, but proposals for Lisa and Macintosh are dated respectively to October 1978, and May of 1979.¹⁸ A demo of Smalltalk was given to Apple's engineers that consolidated their understanding of a graphical user interface with windows, menus and the mouse as graphical input device. Another important influence

¹⁸ In *Holes in History* Jef Raskin, head and father of the Macintosh project, passionately argues against such simplified chronicles [Raskin 94a]. Raskin cites an e-mail conversation with Robert Cringely: «As for all the business of what project started when, whether Lisa started before or after Steve [Jobs, founder of Apple Computer, Inc.] visited PARC, whether the Mac had already begun or not, well I don't think that it really matters very much. My attempt was to EXPLAIN [...], not to be a historian.» How an author can hope to explain what happened if he doesn't even know what happened eludes me.» [Ibid., p. 15]

for the Lisa team was MIT's Spacial Data Management System. The way Lisa, and later the Macintosh user interface deal with spacial arrangements of icons has their roots in SDMS Dataland. The Lisa was presented in 1983 but like the Xerox Star before the Lisa was not commercially successful. The interface concepts were too ambitious for the performance for the hardware of that time.

Finally with the Macintosh, Apple managed to offer a system with an intuitive user interface for an acceptable price. Together with the Apple LaserWriter and a built-in network, the Macintosh turned out be extremely popular and commercially successful.

No historical overview can mention all the contributions that led to the graphical human-computer interface that we are all using today. The following sections will present some astonishing concepts and ideas that are in some cases more than 40 years old but not outdated in any respect.

The discussion could have started again with Vannevar Bush. But a cross reference to section 2.1.1 Memex (p. 12) shall do.

3.1.1 Man-Computer Symbiosis

Bob Taylor describes the first generation of human-computer interaction in the following way – cited by Stephen Segaller [Segaller 98, p. 39]),

“[In] those days to work with a computer you had to go punch a bunch of holes in either paper tape or cards. Then you had to take these cards to the computer room and turn them over to someone usually with a white coat on. That's called batch processing.”

Mainframe computers used to be the size of rooms. Frequently turn around cycles for the user took entire days, sometimes only to figure out that the program contained a syntax error. This depicts the atmosphere in which Joseph Licklider envisions a tight cooperation between human and computer. In his article *Man-Computer Symbiosis* [Licklider 60] he adopts the biological term symbiosis for the relation, because each part has qualities that the other lacks. A computer can solve faster and more accurately mathematical and logical tasks. It can better perform mechanical routine tasks like sorting and searching for information. On the other hand humans are superior at redundant natural languages. Many typical tasks of a scientist include determining the logical consequences for a given situation and preparing the arguments that support a theory or a new insight. A symbiotic system is beneficial for both participants and has qualities that neither of the parts alone is able to do. Licklider envisions an interactive ensemble between human and computer. He calls for a computer that supports the scientist in a way that is much more direct and intensive than anything that seems to be possible with the mainframes of the 1950s and 1960s. Licklider writes [Ibid., p. 133]:

One of the main aims of man-computer symbiosis is to bring the computing machine effectively into the formulative parts of technical problems. The other main aim is closely related. It is to bring computing machines effectively into processes of thinking that must go on in “real time,” times that moves too fast to permit using computers in conventional ways.

He continues [Ibid.],

To think in interaction with a computer in the same way that you think with a colleague whose competence supplements your own will require much tighter coupling between man and machine [...] than is possible today.

It should not be neglected that Licklider gives also the example of a military commander, who has to make critical decisions in less than 10 minutes. A tactical analysis with a response time of days is unusable in this situation.

3.1.2 Sketchpad

Lincoln Laboratory at MIT had one of the first transistorized computers, a TX-2 with 69,632 words of core memory to 36 bits each. One of the projects for this machine was Sketchpad by Ivan Sutherland in the early 1960s. Sketchpad is the first program to interactively create line drawings of striking complexity on a computer screen.

The console to operate the Sketchpad system is shown in Fig. 3.1. The central device is a 7 by 7 inch scope with a 1024 by 1024 raster. A light pen is attached to directly point to the screen. The position is tracked whilst the light pen is moving. A termination flick event is created when the pen is abruptly moved away from screen. The second hand is used to press command buttons. A box holds about 40 buttons that correspond to operations like DRAW, MOVE and DELETE. Sutherland explains how to draw a line with Sketchpad [Sutherland 63a, p. 2],

If we point the light pen at the display system and press a button called “draw,” the computer will construct a straight line segment which stretches like a rubber band from the initial to the present location of the pen [...] A sudden flick of the pen terminates drawing [...].

In order to draw a circle [Ibid.]

[...] we place the light pen where the center is to be and press the button “circle center,” leaving behind a center point. Now, choosing a point on the circle (which fixes the radius) we press the button “draw” again, this time getting a circle arc, whose angular length only is controlled by the light pen position [...]

Pressing a button labeled “Stop” has the same effect as a termination flick of the light pen. The current position is taken as final and the drawing of the line or circle is completed.



Fig. 3.1 Ivan Sutherland's Sketchpad console, 1962. Sketchpad is operated with a light pen and a command button box (under left hand). The four black knobs below the screen control position and scale of the picture.

Other command buttons are used to apply constraints to the drawing. “Horv” for example constrains a line to be horizontal or vertical. Other constraints can be more complex. Constraint P for example defines the relation between four points: «Line from first to second would be parallel or perpendicular to line from third to fourth» [Ibid., p. 74]. Many of such constraints can be applied to a drawing. If the user moves one point the algorithms of Sketchpad try to adapt the lines according to the conditions.

Once a picture is stored on magnetic tape of the TX-2 it can be used as a master graphic for new instances. One master graphic might be a bolt that is used at many places, in different angles in a building plan. If the image of the bolt is modified, all instances change as well. This is the earliest application of object-oriented concepts for a computer graphics editor. It is clear right now, that Sketchpad can be taken as the ancestor of CAD application programs par excellence.

Sketchpad does not display any user interface controls like menus and scrollbars. The presence of the console indicates that all such elements are still external to the screen. The box with the buttons holds the commands. A bank of toggle switches to the left is used to turn functions on or off.

Sketchpad is also capable of zooming and scrolling the picture. In other words the actual drawing area could be much larger than the screen. The screen behaves as a window into this area, which can be moved and zoomed with the four black knobs just above the table. If the picture becomes too large to fit onto the screen an algorithm calculates the clipping with the edges of the screen. Sketchpad III by Timothy Johnson¹⁹ is an extension of Sketchpad for the construction of 3-dimensional objects [Sutherland 63b, p. 344]. Johnson uses Sutherland's clipping algorithm to tile the screen into four independent quadrants. Fig. 3.1 shows the ancestors of windows for graphical user interfaces.

3.1.3 NLS/Augment

Doug Engelbart and his team at SRI did a lot of research work on the design of in- and output devices. The main design principle was to continuously improve the physical and mental relation between human and the machine. The term bootstrapping is coined by Engelbart to describe, that all systems are not only developed but also used extensively by the members of the Augmentation Research Center (ARC). All source code, reports, internal documentation, e-mails and printed letters at ARC are written with NLS in order to discover improvements for the next generation of the system. The text editing capabilities of NLS have already been discussed in 2.1.3 NLS/Augment (p. 16).

By the end of the 1960s the output screen is a conventional TV monitor (cf. Fig. 3.2). An inhouse cable TV network is used to deliver the signal to the offices of the ARC members. The original proposed 17" computer monitors would cost about \$15,000 to \$20,000. Too much for ARC's budget. Smaller 5" monitors are affordable but not sufficient for an ergonomically designed computer workplace. That's why Doug Engelbart chose the indirect way. The time-shared mainframe computer generates images on a couple of 5" raster-scan displays. A high-quality video camera is mounted in front of each screen, takes the picture, amplifies the signal and transmits it to the TV sets in the offices [Engelbart 88, p. 197].

Furthermore this detour offers the advantage to experiment with the video signal. A simple inversion leads for the first time to a positive display of text on screen. Equipment to mix different channels was also reasonably cheap. The famous NLS demo at

¹⁹ The subtitle for Fig. 3.1 in *Medien – Kunst – Geschichte* by Hans-Peter Schwarz on page 61 reads «Ivan Sutherland, ›Sketchpad-Programm, 1962». Compared to Alan Kay's article on *The Early History of Smalltalk* [Kay 96, p. 515, Fig. 11.3], it is more likely that it is Timothy Johnson, instead of Ivan Sutherland, who is using Sketchpad III.

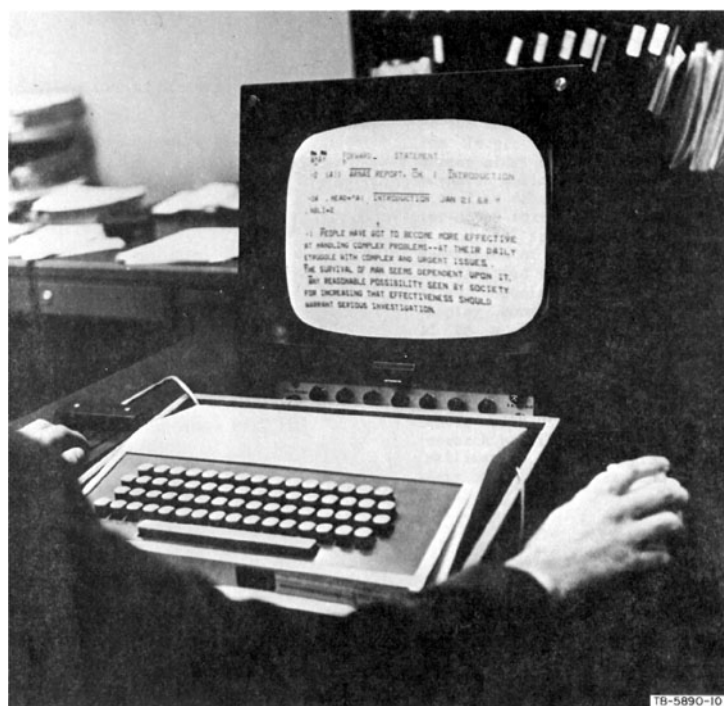


Fig. 3.2 Doug Engelbart's NLS workstation around 1967. The right hand uses the mouse while the left hand lays on the chording keyset.

Fall Joint Computer Conference December 1968 makes frequent use of split screens and overlay effects.

Fig. 3.2 shows Engelbart's NLS workstation around 1967. With his right hand he holds a mouse, and under his left hand is a five-finger chording keyset. We will look now at these two instruments for user input.

The evaluation of graphical input devices for text editing compared the light pen, with joysticks, and with a new development called the mouse. Bill English describes the experiments in *Selection Techniques for Text Manipulation* [English/Engelbart/Berman 67]. The statistical results indicate that the mouse is faster and more accurate than any other device. The original mouse is a little wooden box that «is constructed from two potentiometers, mounted orthogonally, each of which has a wheel attached to its shaft» [Ibid., 2D2]. The motion of the mouse on the table is interpreted as cursor movements on the display. A button on top is used for the select function.

Like the mouse the five-finger chording keyset is also invented at SRI-ARC. It is usually operated with the less dominant hand. The five keys can be pressed not only one by one, but like chords on a piano keyboard several at the same moment. These 31 possible combinations are mapped to the letters of the alphabet. The thumb alone produces an 'a', the index finger a 'b', thumb and index finger together produce a 'c', middle finger a 'd', middle finger and thumb an 'e' and so on. In order to generate more than just the

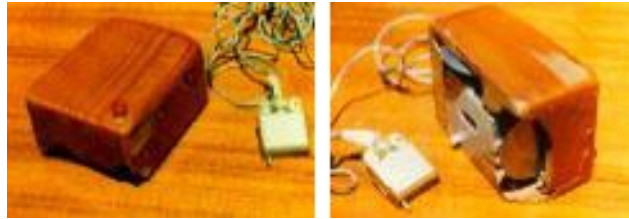


Fig. 3.3 The first wooden mouse, invented by Doug Engelbart and Bill English at SRI in 1963

lower case letters the mapping is systematized to five cases. This is described in *A Research Center for Augmenting Human Intellect* [Engelbart/English 68, 3B5]. The first case contains lower case letters, the second has the upper case letters. The third and fourth case contains numbers and punctuation. Finally the fifth case is reserved for control functions like BACKSPACE or UNDERLINE. To change between the cases a particular chord has to be pressed to change the mode to the control case from where the other four cases can be reached.

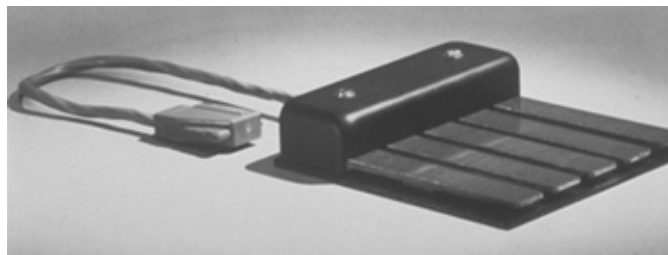


Fig. 3.4 The Five-Finger Chording Keyset

The chording keyset is an equivalent alternative to the QWERTY keyboard. To the NLS system it is the same whether a character is generated by the standard keyboard or by the chording keyset.

The advantage of the chording keyset becomes clear if we consider the motion of the hands during the usage of a mouse-equipped computer. The hands travel back and forth between the keyboard to enter some text and the mouse to locate a new cursor position. The combination of mouse and chording keyset provides a two-handed, higher-speed option. The to and fro of the hands does not happen that often anymore. Words and short phrases can be entered without moving the hands away from mouse and keyset. Engelbart explains, [Engelbart 88, p. 220]

[We] aren't limited to doing things sequentially, but can do things with both hands, concurrently. This stimulated the chord-keyset option.

Longer text passages can of course be entered with the standard keyboard.

Moreover the chording keyset is best suited for NLS' command architecture. All commands are 2-letter combinations. The first letter specifies the operation, the second specifies the type of the operand. For example the input 'CW' stands for COPY WORD, whereas 'CS' is the COPY STATEMENT command. The following click with the mouse selects the target of the operation, in our example either the word or the statement [Friedewald 99, p. 415].

For the members of ARC the collection of input devices has turned out to be very useful and efficient. Once they are learned they can hardly be surpassed by other input methods. But especially the chording keyset is a high threshold for the average user. The invisible modes and artificial mappings are hard to learn and to remember. Thierry Bardini describes the bewilderment of the outside user in *Bootstrapping – Douglas Engelbart, Coevolution, and the Origins of Personal Computing* [Bardini 2000, p. 145]

Because of the highly original nature of the system, some of the practices also were unique to the ARC community and tended to separate that community from outsiders. Some astonished visitors reported that the ARC members had strange codes or habits, such as being able to communicate in a “weird” sign language. Some staff members occasionally communicated across the distance of the room by showing the fingers position of a specific chord entry on the keyset.

3.1.4 Flex Machine and Dynabook

«Machines which do one thing only are boring», says Alan Kay in *The Reactive Engine* [Kay 69]. With the Flex Machine Kay aims in the late 1960s for a tool that can support human cognition processes. His starting point is the human language. People describe the world with words. They use language to communicate with each other. Language is most basically used to think about the world. It follows a special form – it's syntax. And it has a meaning, carried representations of the world and of abstract concepts – the semantics of language.

These considerations call for a computer, that should participate in an interactive dialog with a human, that the interface needs to be flexible enough to express new ideas; as Kay formulates, «it has to be able to form the abstractions in which the user deals» [Ibid.].

The Reactive Engine outlines the system design of the Flex Machine. The computer is equipped with a standard keyboard as well as a five-finger chording keyset like Engelbart's NLS system. A graphic tablet is installed for 2-dimensional input. Output is to be drawn on a calligraphic 1024 by 1024 pixel CRT.²⁰ Like Sketchpad, the actual drawing area is much bigger than the screen size. The drawing algorithms of the Flex Machine can map any rectangular area of a virtual screen to the monitor. And the monitor can

²⁰ A calligraphic cathode ray tube is a special vector display. Bitmapped displays are to be invented at Xerox PARC about four years later.

even be divided between several virtual screens. However window controls like scroll-bars to manipulate the region of the virtual screens are still not employed.

The Flex Machine ought be so compact to fit on a desktop. An utopian concept for the computer technology of the late 1960s. And this goal could never be reached for the Flex Machine, although various components have been implemented on several mini computers. For example, compilers for FLEX have been developed for the Univac 1108 and Doug Engelbart's SDS 940 at SRI.

The FLEX language is integral part of the machine (cf. 2.1.5 Flex and Smalltalk (p. 19)). The user interacts with the system entering FLEX statements that get executed immediately and the results are displayed on the monitor.

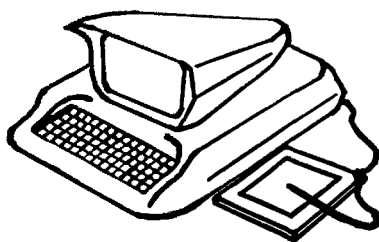


Fig. 3.5 Alan Kay's vision of the Flex Machine, 1969

With the Flex Machine Alan Kay lays the foundation for Personal Computing. With the Dynabook his vision takes full shape. The Dynabook is inspired by the invention of the flat-panel display, although the prototype of 1968 is just a 1 square inch piece of glass. Alan Kay, now at the Xerox' newly founded research laboratory in Palo Alto, starts to ponder about a conjunction between the Flex Machine and the flat-panel display. Two papers present the resulting concept, *A Personal Computer for Children of All Ages* [Kay 72a] and *A Dynamic Medium for Creative Thought* [Kay 72b].

The Dynabook is the distilled vision of a «personal, portable information manipulator» [Kay 72a, p. 1]. It is a device useful and powerful enough to support adults in their daily work, and it should be simple enough to be used by children for playing and learning.

Fig. 3.6b shows a mockup of the computer. The shape of the case should be similar to an ordinary notebook. The Dynabook has a keyboard, and a stylus is used as a pointing and drawing device directly on the surface of the display. Alan Kay would have liked to have a black and white flat-panel display of 8H by 11 inch with a resolution of at least 100 DPI. This high resolution is necessary to compete with paper for legibility of text. The option of several different bit-mapped fonts and font faces is crucial to make the Dynabook a personal device.

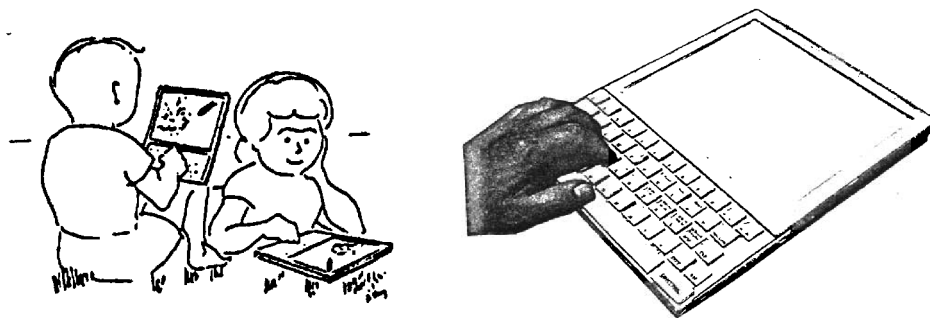


Fig. 3.6 (a) Children playing with Dynabooks and (b) a mockup of the Dynabook from 1972

To illustrate how a Dynabook can be used, Alan Kay portrays a scenario with two children. They have connected their Dynabooks to play the multi-agent game *Spacewar*. After they lost interest in playing they discuss how to incorporate the attractive force of the sun. They know how to reprogram the game, because they have written it themselves. But they have no idea about the mathematical formulas for gravity. They decide to ask their teacher for help. The teacher connects her Dynabook to the local library to look for information about our solar system. She copies articles that match her criteria to her Dynabook and gives some hints to the waiting children how they can improve their *Spacewar* program.

The scene touches on many important aspects for personal computing. First of all everyone has its own computer. Kay foresees a price tag that is in the same range with a television set. This alone is pure science fiction in the early 1970s when minicomputers start at \$10,000. But a low price is necessary to make the Dynabook affordable to everyone. Dynabooks should also be portable and independent from wired networks for data transmission and power supply. All this is merely a matter of time and technological progress. Really important are the implications of the personal computer as a new medium. Alan Kay defines [Kay 72a, p. 3]:

What then is a personal computer? One would hope that it would be both a medium for containing and expressing arbitrary symbolic notions, and also a collection of useful tools for manipulating these structures, with ways to add new tools to the repertoire.

The children in the scenery above use their Dynabooks as a personal medium in the way that they use it for reflexive communication. They are author and audience of the medium and use the Dynabook like a conventional notebook – although as a slightly magically enhanced one.

Alan Kay continues that it is essential for personal computers to be «superior to books and printing in at least some ways without being markedly inferior in others» [Ibid.]. The teacher connected her Dynabook with ease to a local library. Entire books can be transferred to the Dynabook for study and leisure. This is also an indication for a new

kind of medium as Marshall McLuhan argues in *Understanding Media: The Extensions of Man* [McLuhan 64], that new media adopt the content of previous media first. But a computer can give access to the content of a book in a more flexible way than its printed counterpart can do. Kay contends, «It need not be treated as a simulated paper book since this is a new medium with new properties» [Kay 72b, p. 3]. From here to Ted Nelson's vision of a global form of hypertext is just a small step.

Why children? The reason why Alan Kay took two children for his introductory scene is more profound than just to accent the Dynabook's easy and straightforward user interface. Alan Kay is impressed by the psychological works of Jean Piaget of the 1920s and Jerome Bruner's of the 1960s. Both scientists have studied children to figure out how they learn. The result is a model of human learning and cognition processes that forms the theoretical background for Alan Kay's Learning Research Group at Xerox PARC (LRG). Kay and his team manage to develop principles for human-computer interaction that are in accord with these psychological models of human cognition (cf. 3.2.2 Three Stages of Human Development (p. 78)).

3.1.5 Xerox Alto, the Interim Dynabook and Smalltalk

The Alto is the first computer with a size small enough to fit beneath a desktop table (cf. Fig. 3.7a). It is equipped with a mouse, additionally it had a regular keyboard with the option to connect a five-finger chording keyset. The new 600 by 800 pixel bit-mapped graphic screen can display a full-page with 8G by 10H inch at 72 DPI. Network capabilities are also built-in for the new Ethernet technology. The parameters are described in detail by Butler Lampson in the Xerox Inter-Office Memorandum *Why Alto* in 1972 [Lampson 72]. It is the first time that a research agenda conceives the development of computers to be used by one person each. Time sharing on minicomputers was standard for that time. Personal computing was envisioned by just a few people. Among them are Butler Lampson and Alan Kay.

As soon as the first Altos became operational in 1974 the bootstrapping power unfolds. The network of Alto computers became a perfect testbed for numerous software projects at Xerox PARC. For example Charles Simonyi and Tom Malloy developed Bravo, the first WYSIWYG word processing application. (Simonyi later went on to Microsoft to create Microsoft Word.) Markup and Draw are two application programs to create illustrations and images. Electronic mail could be handled with Laurel, and Neptune was the program to manage files. Documentation on these application programs are part of the *Alto User's Handbook* [Taft 79]. The spectrum of capabilities is also described by Thomas Wadlow in *The Xerox Alto Computer* [Wadlow 81]. He mentions that the Alto computers have also been used for a bunch of games. Trek and Mazewar are among the favorites, because several players with their personal Altos could participate simultaneously on the local network.

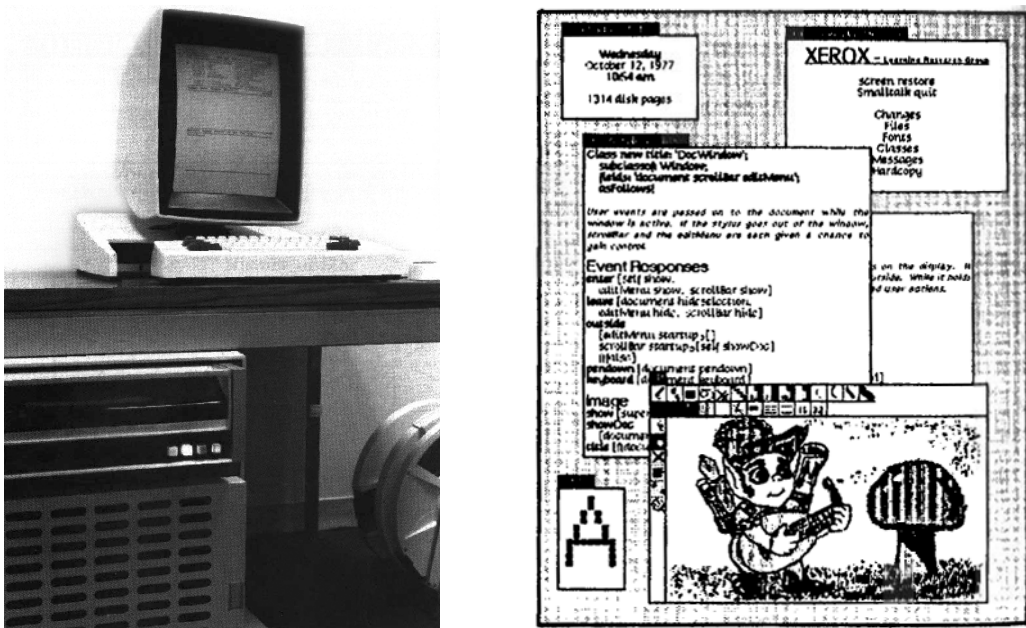


Fig. 3.7 (a) The Xerox Alto II personal computer, ca. 1975. (b) Typical screen of Smalltalk-76. The windows behave like overlapping sheets of paper.

The 72 DPI graphic display of the Alto computer offers an acceptable resolution to bring into use typographic bit-mapped fonts. Together with the Research Character Generator (RCG) and the Scanning Laser Output Terminal (SLOT) it becomes possible to display text with different fonts, spacing and layout on screen much alike the printed output. This relation is the core of “What you see is what you get”, which is abbreviated as the WYSIWYG paradigm.

To Alan Kay the Alto means a step towards his vision of the Dynabook. Consequently he calls the Alto an «Interim Dynabook» (e.g. in *Personal Computing* [Kay 75, p. 5]). Together with Dan Ingalls he develops Smalltalk in the tradition of the object-oriented programming language Simula by Kristen Nygaard and Ole-Johan Dahl, the interactive language Logo by Seymour Papert and the homoiconic language FLEX by Kay himself (cf. 2.1.5 Flex and Smalltalk (p. 19)).

Windows as we know them today are invented for Smalltalk (cf. Fig. 3.7b). In the 1960s Sutherland’s Sketchpad, Engelbart’s NLS and Kay’s Flex Machine could map rectangular regions of a larger virtual screen to non-overlapping areas of the display. But it is up to Smalltalk to draw a frame around the windows and to actually overlap them like real sheets of paper. A click with the mouse to a visible part of an overlapped window is sufficient to move it to front. Once the window is front and activated, it can be dragged with the mouse to any position on screen. The size can also be changed with the mouse and scrollbars allow to shift the visible portion of the view. Overlapping windows have a couple of positive qualities for graphical user interfaces.

First of all they generate a familiar environment in that they build on the metaphor of overlapping sheets of paper. Secondly, swapping the front window eliminates a mode from the interaction with computers. The user does not have to quit one application program to open another for a different document or piece of information anymore. A simple click to a partly covered window activates the document and the associated program. This interaction mode is called *modeless* because the mode switch between different programs is transparent to the user. A third advantage of overlapping windows is the subjective gain in real screen estate. For example the content of the windows in Fig. 3.7b would not fit on the same screen without overlapping facilities. Even though they are not completely visible they are accessible at any time with a simple click. An acceptable trade-off for more flexibility in window space handling, and a more economical use of screen space.

The introduction of overlapping windows solves the dilemma of preemption between application programs. Larry Tesler goes further and tracks down the mode problem for the domain of text editing. Modes like INSERT, REPLACE, DELETE, and SEARCH had to be specified explicitly before these operations could be performed. Consider for instance NLS. The command letter 'D' puts the system into the mode of deleting. The next letter and the following click define the operand for the command; e.g. 'DW' and a click deletes a word, while the same click deletes the line of text if the given command was 'DL' [Friedewald 99, p. 415].

Tesler's solution holds until today. He introduces the selection of text and reverses the order of command and operand. The border case of a zero-length selection defines a new kind of cursor next to the mouse cursor, namely the text cursor. The right field of Fig. 3.8 shows the text cursor between the 'd' and the comma.

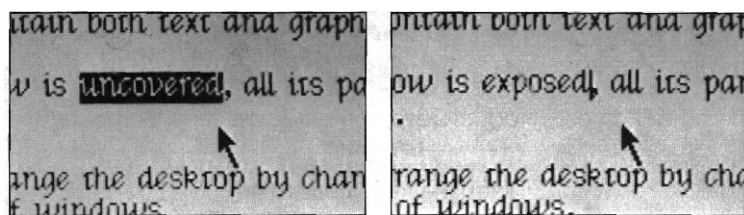


Fig. 3.8 The mouse is used to select a piece of text in the Smalltalk environment. Typing always replaces the selection. «Thus, the usual insert, append, and replace modes are folded into one mode – replace mode – and one mode is no mode at all.» [Tesler 81, p. 104]

Text is selected with the mouse. A single click positions the text cursor between two characters. A double click selects a word, and a click-drag gesture selects a passage. Commands like CUT, COPY and PASTE can be chosen from a pop-up menu.

Overlapping windows, modeless text manipulation and the WYSIWYG paradigm form the user illusion of directly interacting with real objects.

3.1.6 Xerox Star

Development of the Star started in 1975. It was Xerox' intention to exploit the scientific results of Xerox PARC to produce and market a personal computer for office application. The Star workstation is presented in April 1981 as the Xerox 8010 Information System.

The hardware architecture is an improved version of the Alto computer. The paper *Designing the Star User Interface* [Smith et al. 82] by David C. Smith, Charles Irby, Ralph Kimball, William Verplank, and Eric Harslem mentions that the typical memory configuration is 512 KB RAM and a harddrive with 10 MB capacity, optionally 29 MB. The Star has Ethernet built-in to connect the computer to a local areal network with access to a laser printer, shared file servers and other Star workstations. The Star is equipped with a much larger screen than the Alto. The bit-mapped, black and white 17" screen can display 1024 by 808 pixels at 72 DPI. A mouse with two buttons is attached as graphical pointing device. In order to reduce complexity this is one button less than Alto's mouse has. For the same reason the five-finger chording keyset is no longer an option. But the keyboard is extended by a moderate number of function keys to the left, right and top.

Star's interface team presumed that an integrated system would be much better suited for an office environment than the current software situation at Xerox PARC. Many application programs like Bravo for WYSIWYG text layout or Draw for illustrations are impressive if they are evaluated one by one. But they lack a common approach in interface design. Each program employs a different set of interaction techniques. Alan Kay has demonstrated how the dilemma of preemption can be solved for the Smalltalk environment. But several programs are based on different programming languages and cannot make use of overlapping windows.

In order to gain a better understanding of the real situation of their target audience the interface team arranges several site visits at business offices. User studies and task analysis become for the first time an official part of a development process. Some early results from 1976 are published by William Newman and Timothy Mott in *Officetalk-Zero: An Experimental Integrated Office System* [Newman/Mott 82]. Typical tasks include document editing, page layout and illustration. Documents like memos, presentations and letters are filed into folders and retrieved on demand. Personal records are processed. Electronic mail is also an office task of growing importance. The insights are used to develop the prototype Officetalk-Zero. It is field-tested extensively at PARC and at other Xerox departments outside the laboratory in 1978. One of the results of the Officetalk studies indicate difficulties with overlapping windows. Users

seem to waste too much time adjusting size and position of windows [Ibid., p 329]. This leads to the design decision to use non-overlapping windows for the Star interface. The only exception will be property sheets, that are – similar to modal dialogs nowadays – only displayed for a short period of time.

David Canfield Smith joins PARC and the interface team of the Star after he has finished his Ph.D. in 1975 and after a short period at Engelbart's team at SRI thereafter. His doctoral thesis was about a Smalltalk program called Pygmalion – the ancestor for the field of programming by demonstration, respectively for visual programming. One of the main results is the conception of icons. Icons are analogical representations with a well defined semantics. The advantage compared to textual representations is, that they can share functional similarity with structures and actions in the context being modeled. For the domain of office application Smith develops his icon idea into those which represent documents, folders, file cabinets and mail boxes. Fig. 3.9 shows the final version of the icons as they appear on the Xerox Star desktop. Icons are more than plain graphical illustrations. They stand for data and behavior. Operations that are performed on icons result in the manipulation of the data structures they represent. In *Pygmalion: An Executable Electronic Blackboard* [Smith 93] Smith argues that,

Analogical representations suggest operations to try, and it is likely that operations applied to analogical representations would be legal in the other context, and vice versa. This is the philosophical basis for the design of the Xerox Star and, ultimately, Apple Macintosh “desktop” user interfaces; they are a metaphor for the physical office [...]. Being able to put documents in folders in a physical office suggests that one ought to be able to put document icons in folder icons in the computer “desktop,” and in fact one can.

To interact with the system the Star interface offers a reduced set of commands, that is MOVE, COPY, OPEN, DELETE, SHOW PROPERTIES, COPY PROPERTIES, AGAIN, UNDO, and HELP [Smith et al. 82, p. 307], [Friedewald 99, p. 346]. These commands are permanently assigned to function keys on the keyboard. They are sufficient to operate all aspects of the system, because they are used in a generic fashion. For example, if the user wants to file a document into a folder, she has to select the corresponding document icon with a single click, and has to press the function key MOVE. The next click on a folder icon determines the target folder and moves the document. If she wants to print the document, she has to use the MOVE command once again. But the next click on a printer icon activates in this case the printing process of the document.

The limited set of generic commands avoids the complex command structure of other programs. Together with the relation between icons and terminology to the physical office environment it helps the user to build a conceptual model of the system. This model is referred to as the physical-office metaphor, or the desktop metaphor. It turns out to be essential especially for casual users, because its consistency allows to draw

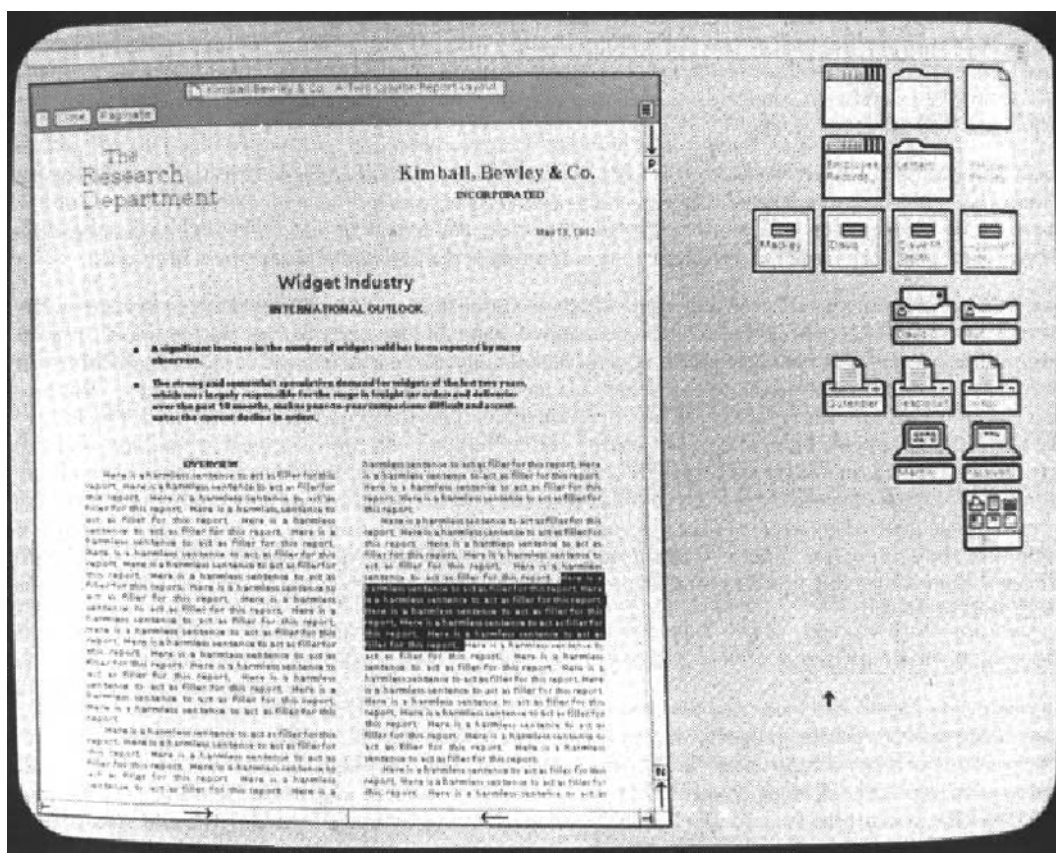


Fig. 3.9 Xerox Star, 1981. A document window with layout using multiple fonts. On the Desktop are icons representing a record, a folder and a document (first row). The third icon in the second row without a frame is the open icon for the document window. It is followed by four file drawer icons, then electronic mail – in and out, three printers, two terminals, and finally a directory of a remote network. The white bar on top is a status bar – it shouldn't be considered as a menu bar.

analogical assumptions that are in fact valid operations in the context of the environment. A deep technical understanding of a computer is no longer necessary to use a computer.

3.1.7 Spatial Data Management System

According to Nicholas Negroponte and Richard Bolt, the Spatial Data Management System (SDMS) was motivated by two ideas [Brand 87, p. 138]. First is the psychological notion of motor-memory reinforcement. For instance, to overcome the uncertainty whether one has locked a door or not, one can remember the cold perception and the weight of the key in ones hand. To find a book one can remember the raised and stretched out left arm, that was used to put the book on the top of a shelf. A final example to illustrate the importance of physical involvement is about two scientists, arguing a topic in front of a blackboard. They «will refer each other to diagrams, equations, and terms on the basis of where they had been written, even long after they have

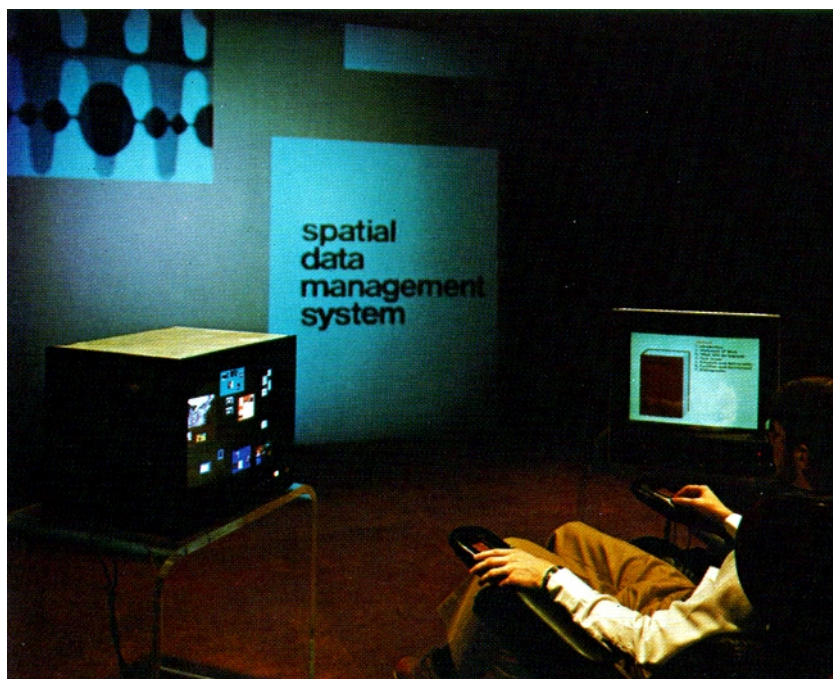


Fig. 3.10 *The Media Room of the MIT Spatial Data Management System, around 1977. The user sits in a comfortable armchair with an integrated joystick and touch-sensitive pad. Two monitors and a wall-size projection screen are used as displays.*

been erased» [Ibid.].

The second idea is inspired by the ancient Greek poet Simonides of Ceos. Simonides should have been famous for his ability to recite epic poems entirely from memory.²¹ Negroponte and Bolt explain how he managed to remember such long tales without making any mistakes [Ibid.]:

His secret was to tie each successive part of a to-be-remembered poem or speak to a specific locale within the mental floor plan of either an actual or imagined temple. ... For each successive subsection of the talk to be given, the orator would mentally walk from place to place within the temple, rehearsing the appropriate material before some specific piece of statuary.

The psychological explanation to this phenomenon is, that humans learn without effort about the relationships of objects in space. Simonides employed this quality to attach names, things, and even abstract ideas to spots of his mental temple.

The first motivating idea reflects on our physical interaction with the world, while the latter accentuates our mental preference for spacial relations. With SDMS Bolt and

²¹ Simonides has a name as the founder of mnemotechnics, the art of memory. In *Informations- und Wissensorganisation anhand räumlicher Ordnungsmodelle: Das Spatial Data-Management System der Architecture Machine Group als Fallbeispiel* Kirsten Wagner argues, that mnemotechnics has first been applied to computer science by Negroponte and Bolt [Wagner 2000]. Mnemotechnics is named after Mnemosyne, the personification of memory in Greek mythology.

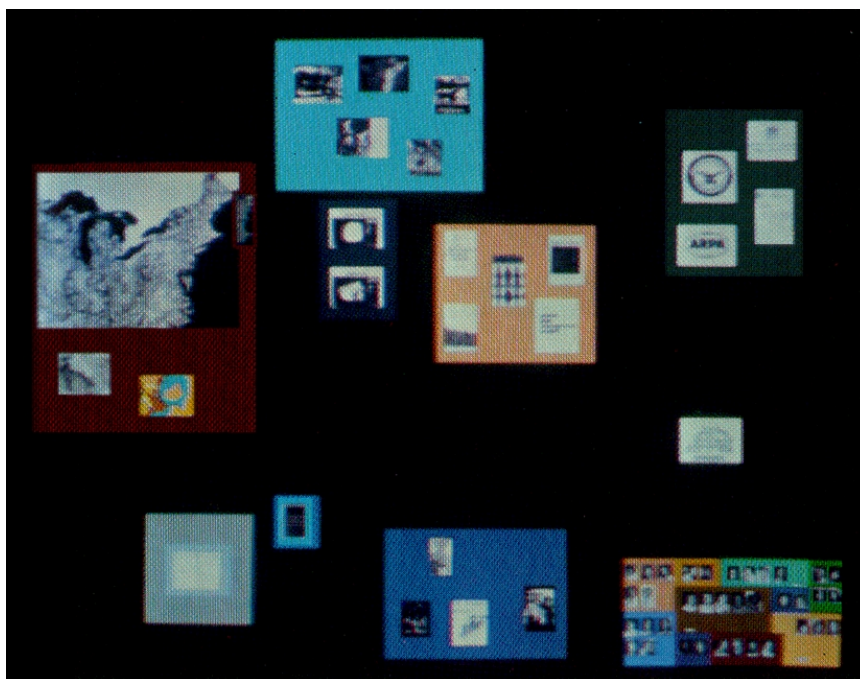


Fig. 3.11 *The Dataland of the MIT Spatial Data Management System. The light translucent overlay (bottom left) represents the section that is currently displayed on the main screen.*

Negroponte devise a human-computer interface that takes into consideration both aspects. Objects like books, letters, and telephones are spatially arranged in a 2-dimensional Dataland. The positions are persistent in order to allow humans to become used to them.

Most exceptional is SDMS' working environment. The user sits in the center of the Media Room (cf. Fig. 3.10). Two small monitors are set up front left, respectively front right. Between them opens the view to a wall-size projection screen. The left monitor always contains an overview of the Dataland (cf. Fig. 3.11). The user can point with a finger to the touch sensitive screen, or navigate with a joystick over the Dataland to define a section that should be blown up for the main projection screen. In *The Human Interface: Where People and Computers Meet* [Bolt 84, p. 12] Richard Bolt points out, that

[...] the items in Dataland are *facsimile* in nature: books look like books, calendars like calendars, and so on. Dataland is not a map of the data. It is the data.

SDMS does not distinguish between icons and document windows, like Xerox Star will do a few years later. The user can zoom into the minimized thumbnail world of material until an item becomes clear and legible. The monitor to the right will display control elements to edit the document in focus. For example, if the main screen displays a calculator in full-size the secondary monitor offers numerical and mathematical func-

tion buttons. If the user zooms in a book, the monitor will display the table of contents. The book can be read on the main screen. To turn a page the user has to perform a diagonal top-right to bottom-left finger stroke on the touch-sensitive pad, that is built into the arm rest. The turning of the page is accompanied by a page-flipping animation. Richard Bolt argues [Ibid., p. 15],

[...] This page-turning animation visually separates one page from the next and gives readers a sense of where they are in the material in a way that endlessly scrolled text would not.

On long jumps that are directly issued at the table of contents, the animation takes longer to indicate more skipped pages. This artificial delay – computers could do it almost instantaneously – is the equivalent to the physical steps of opening a book on a specific page. It helps to later recall the position of a paragraph of interest.

Later versions of SDMS and military implementations introduce a hierarchy of Datalands. A new item called port connects Datalands with each other. If the user focusses on a port and continues to zoom in the Dataland of the next level unfolds.

It shall also be mentioned, that the Architecture Machine Group did research on the field of multi-modal input. A third methods of navigation in SDMS is «voice travel». For example, the spoken words “Take me to the calculator.” would focus the main screen accordingly [Ibid., p. 13]. The project Put-That-There of 1980 utilizes speech recognition and pointing gestures to move items in Dataland [Bolt 84, pp. 35].

3.1.8 Apple Lisa

Apple Lisa is much more than just a computer between Xerox Star and Apple Macintosh. In fact the development of Lisa was almost independent from Xerox Star and evolved parallel to Apple’s Macintosh project.

The Lisa computer is assembled to fit into the same case as its 12 inch monitor. The screen has 720 by 364 pixels with a horizontal resolution of 90 DPI [Schreiber 90, p. 291]. The small vertical resolution of about 60 DPI is caused by rectangular pixels. Some psychological experiments had indicated, that the horizontal resolution is more important for legibility than the vertical value [Friedewald 99, p. 382].

The final design of the mouse has just one button. It was quite vividly argued about the correct number of buttons. «What ensued became known as the “button wars”», recall Roderick Perkins, Dan Keller, and Frank Ludolph in *Inventing the Lisa user interface* [Perkins et al. 97, p. 46]. Several user tests and the aim to make the system easy to use for naïve office employees lead finally to the decision of one button.

Lisa is equipped with 1 MB RAM, a 5 MB harddrive, and a powerful 5 MHz Motorola 68000 processor. The capabilities of the processor strengthened the impression of the design team, that «the Lisa would be so fast that it would be waiting on the user most

of the time! The idle time could then be used to drive a more elaborate user interface» [Ibid., p. 44].

The top maxim of Lisa's interface design was that it must be fun to use the computer. The design team started with the filing functions in the early 1980s. The questions to solve are [Ibid., p. 47]:

- How are documents created or destroyed?
- How are they located?
- How are they returned to their filing homes?
- How should their attributes be displayed?

Several prototypes worked quite well, but they didn't pass the test to be fun to use. One of the ideas that was fun to use was inspired by MIT's Spacial Data Management System. Bill Atkinson developed a functional prototype that was capable of displaying a desktop with spatially arranged icons. Roderick Perkins recalls [Ibid., p. 50]:

Bill [Atkinson] adapted this idea to the filing problem by creating an enormous virtual desktop, perhaps a mile square, and then providing methods for very quickly moving around and zooming in or out. Documents were represented as small icons that could be organized spatially, with related documents placed near each other.


This attempt was rejected, because it didn't fit into an office environment. It was compelling for technicians but not easy to understand for novice users. What remained was the appreciation for spacial quality.

The final design of the Lisa Desktop Manager uses icons for documents and folders. It introduces the trash can for the desktop. Icons can be moved with drag & drop to folders or to the trash can, where they remain before they are finally deleted. This supports a feeling of confidence for the user that all filing operations can be undone. A double click is utilized as a shortcut for selecting an icon and choosing the most likely OPEN command from the menu. If a document is opened the icon on the desktop or in folder changes to a shadow icon. In *The Lisa User Interface* Frank Ludolph and Roderick Perkins argue [Ludolph/Perkins 98, p. 18]:

Since users never saw both document icon and window at the same time, a window was perceived as just a form of presentation, not as an independent object with its own edit state.

A Xerox Star document can contain all datatypes collectively. Especially text and image can be freely arranged and edited in place. Unfortunately this advantage for the user causes a closed software architecture. It is a problem for third parties to develop programs that interact seamlessly with the existing programs and documents. Lisa on the other hand has an open software architecture. Apple offers documentation how to utilize the Desktop Libraries for custom development. Also a style guide enti-

tled *Lisa User Interface Standard* was written in 1980 in order to achieve a consistent “look and feel” among all Lisa application programs [Friedewald 99, p. 387].

Lisa introduces the menu bar with pull-down menus. The bar is located at the top edge of the screen (cf. Fig. 3.12). It is shared between Lisa programs, that hook their commands into the structure of the menu bar. Some frequently used commands can be selected with a keystroke combination out of a special Apple command key together with a character – e.g. ‘D’ duplicates a document.

The Lisa has a document-centric user model. That means that the user should not notice anything about application programs. They are called tools and reside somewhere on the hard disk – but that’s all. It is even not possible to launch a tool directly. They get called automatically by the Desktop Manager if the user wants to open an associated document. Tools are also terminated by the Desktop Manager if the document is closed and another tool needs memory resources. Consequently tools on Lisa have no QUIT command.

Tools on Lisa have no NEW DOCUMENT command, either. To create a new document of a specific type the user has to use a stationery pad. This is a special kind of document that acts like a template for new documents. Opening a stationery creates a new document by duplicating the content, and automatically naming the new document like the stationery with the current date appended.²²

Saving a document is an uncritical action, as the Lisa system continuously saves all changes to an invisible ‘suspend’ file. The user can be sure that everything is kept by the system, even if she is interrupted by a different task. Documents can be temporarily set aside without the need to save them. In this case an icon remains on the desktop, that can be called up at any time to open the document in the same state. Finally if the user decides to close the document a modal dialog asks whether the changes should be saved permanently or not. The following table is based on Arthur Naiman’s *Introduction to the Lisa* [Naiman 84]. It should convey an impression of available operations for docu-

File/Print	File/Print
Set Aside Everything	Set Aside Everything
Set Aside	Set Aside
Save & Put Away	Save & Put Away
Open "*"	Save & Continue
Duplicate	Revert to Previous Version
Tear Off Stationery	Format for Print...
Make Stationery	Print...
Monitor the Printer	Monitor the Printer

Tab. 3.1 *Lisa’s File/Print menu for the Desktop Manager (left) and LisaWrite (right)*

²² Even folders have to be created via a stationery pad for new folders [Naiman 84, p. 77].

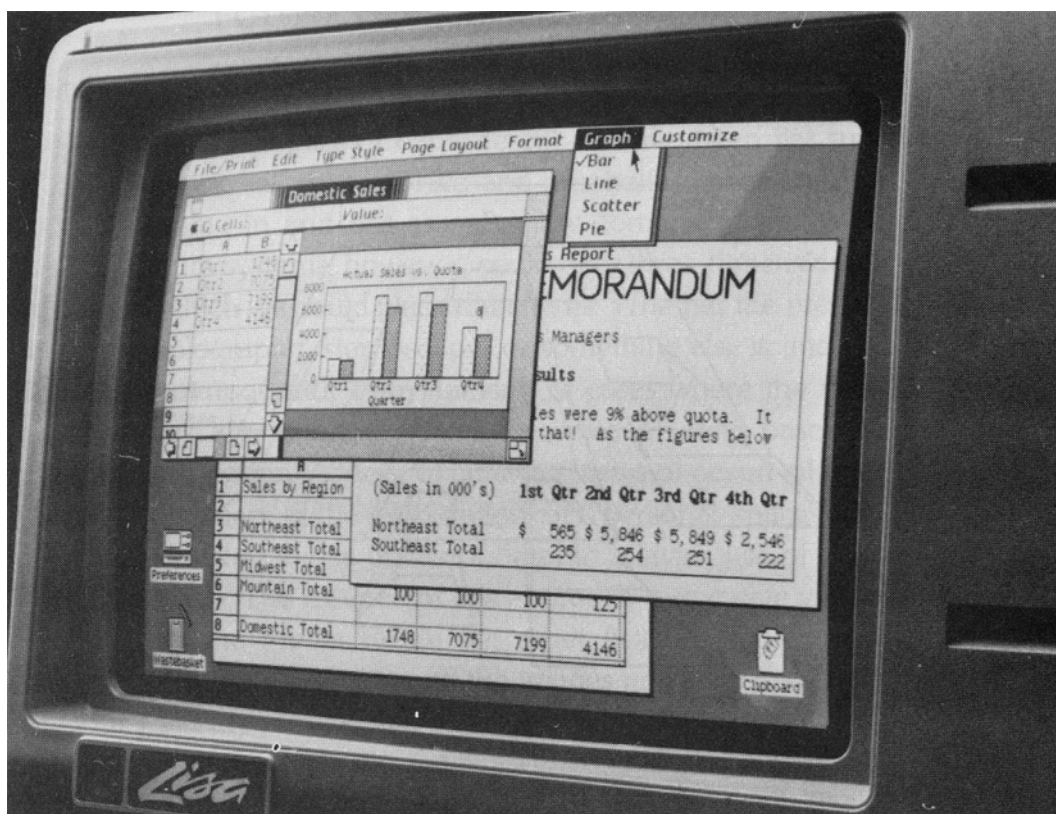


Fig. 3.12 Apple Lisa, 1983. Menu bar on top of the screen. The “Wastebasket” is bottom left.

ments on the Lisa system. The Open command reflects the name of the currently selected item.

The designers of the Lisa expanded this special saving approach even to switching the computer off and on. Arthur Naiman explains [Ibid., p. 83],

[...] to finish a work session with the Lisa [...] all you have to do is ... turn the machine off. It doesn't matter whether you're in a window or on the Desktop—the Lisa saves onto the disk all the changes you've made to any document. [...] The Lisa also remembers what windows were open, their size, shape and location on the screen, etc., etc. When you turn the Lisa on again, it puts you back in exactly the same place, so you can begin working right where you left off.

This very impressive behavior of a system that is nearly twenty years old can only be compared with the sleep mode of modern PCs.

The estimate that the 68000 processor was powerful enough for this sort of interface turned out to be false. Interacting with the Lisa required a lot of patience.

3.1.9 Apple Macintosh

The Apple Macintosh computer starts to sell for \$2,500 in 1984.²³ The low price could be reached because the computer is equipped less generously than Apple Lisa. It uses a Motorola 68000 processor with a clock speed of 8 MHz. But it has only 128 KB RAM – yet a 64 KB ROM for core segments of the operating system. The monitor has 9 inch in diagonal and offers 512 by 342 (squared) pixels at 72 DPI. The Macintosh ships with a 400 K floppy disk drive – a hard disk is not required.

Early Apple recognizes the importance of software for the new machine. It is crucial to have a wide range of application programs to compete against IBM PCs market share. Especially an early agreement with Microsoft guarantees the availability of programs like MS Word for the Macintosh. For instance Microsoft Excel is originally developed on the Macintosh platform. The constellation of Macintosh, AppleTalk network, Apple LaserWriter, and the WYSIWYG layout program Aldus PageMaker established the Macintosh as the computer of choice for the desktop publishing revolution [Friedewald 99, p. 404]. Soon the hardware becomes more powerful – so the model of 1984 is just the first in a long and successful product line of Apple Macintosh computers. This thesis for example is written on a Macintosh PowerBook with 192 MB RAM and a processor speed of 400 MHz.

What the Lisa Desktop Manager is to the Lisa, is the Finder to the Macintosh. Icons are used to represent disks, hard drives, folders, application programs, and documents. Items are stored in a hierarchy of nested folders, and a trash can completes the desktop metaphor. Overlapping windows are used to display the content of folders.

The Finder release of 1991 introduces the notion of an alias. An icon of type alias is a reference to an arbitrary Finder item, that can be used as a substitute for the original item. An alias can be created with ease to virtually store the item at a second location in the tree structure of folders.²⁴

The original Macintosh design is restricted to run a single application program at a time. If the user opens a document the Finder is terminated before the program launches to display the document window. This is a concession to the limited memory resources of the Macintosh in the mid-1980s. Almost all current graphical user interfaces still suffer from those early years of Macintosh, when the application-centric user model was perpetuated. Lisa's document-centric user model was carefully tested, but it

²³ Lisa's original price tag was \$9,995. With the introduction of the Lisa 2 in 1984 it was reduced to \$3,500. Source: *Apple Confidential: The Real Story of Apple Computer, Inc.* by Owen Linzmayer [Linzmayer 99, p. 61, 77].

²⁴ Microsoft will implement this concept as 'shortcuts' in Windows 95.

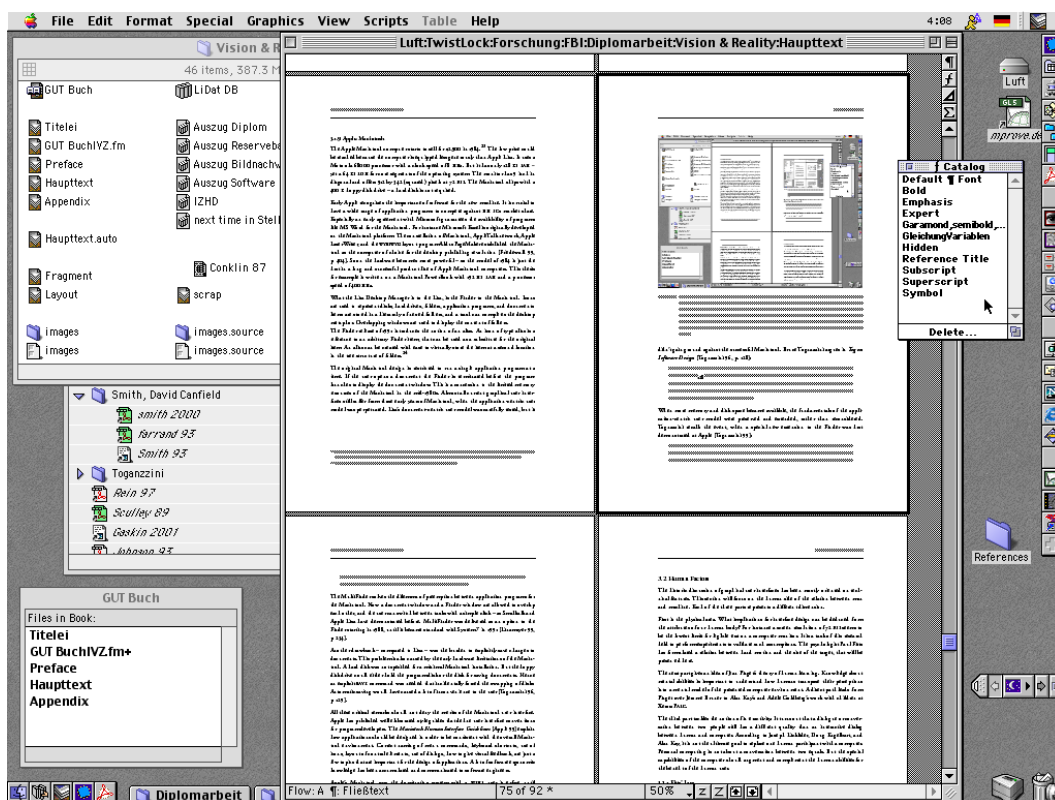


Fig. 3.13 Self-referential screenshot of this document in Adobe FrameMaker under Mac OS 9.0. The tool window “f Catalog” offers predefined character styles for text edits. The menu bar is located at the top edge of the screen. The two inactive windows (top left) belong to the Finder – “GUT Buch” is a FrameMaker’s project window, that assembles the chapter documents into a book. To the right are a few desktop icons: A hard disk, an alias, a folder, a desktop printer, and the trash can. The run of small icons at the right edge and bottom left give access to frequently used application programs.

didn’t gain ground against the successful Macintosh. Bruce Tognazzini argues in *Tog on Software Design* [Tognazzini 96, p. 128]:

Why don’t we have the compound document model in use today? Because the Macintosh was a 128-kilobyte machine with a single disk drive. A user couldn’t possibly have more than one tool in the computer at any time because there wasn’t room. Since only one tool could be used with a document, then the tool might as well handle the opening and closing of that document.

When more memory and disk space became available, the fundamentals of the application-centric user model were preserved and extended, rather than reconsidered. Tognazzini recalls the event, when a special new extension to the Finder was first demonstrated at Apple [Tognazzini 99]:

[They] started up the Mac, showing the Finder, then launched MacWrite. MacWrite opened up with a new, full-screen document, as it always did. They typed a few lines, then grabbed the mouse and headed for the size box, down at the corner of the win-

dow. When they shrank the window back, the [Finder], with all its files and folders, was revealed beneath. We were totally blown away. Totally.

The MultiFinder solves the dilemma of preemption between application programs for the Macintosh. Now a document window and a Finder window are allowed to overlap each other, and the user can switch between tasks with a simple click – as Smalltalk and Apple Lisa have demonstrated before. MultiFinder was delivered as an option to the Finder starting in 1988, until it became standard with System 7 in 1991 [Linzmayr 99, p. 234].

Another drawback – compared to Lisa – was the burden to explicitly save changes to documents. This problem is also caused by the early hardware limitations of the Macintosh. A hard disk was not specified for a minimal Macintosh installation. But the floppy disk drive could either hold the program disk or the disk for saving documents. Hence an explicit SAVE command was needed that incidentally forced the swapping of disks. Automatic saving would have caused a lot of inconvenience to the user [Tognazzini 96, p. 129].

All these critical remarks should not deny the merits of the Macintosh user interface. Apple has published well elaborated style guides that define user interface conventions for program developers. The *Macintosh Human Interface Guidelines* [Apple 95] explain how applications should be designed in order to be consistent with the overall Macintosh environment. Correct naming of menu commands, keyboard shortcuts, use of icons, layout of control elements, use of dialogs, how to give visual feedback, are just a few topics that are important for the design of applications. A lot of software ergonomic knowledge has been accumulated and communicated to software engineers.

Apple's Macintosh was the dominating system with a WIMP user interface until Microsoft turned into a serious competitor with the introduction of Windows 3 for PCs during the early 1990s and its successor Windows 95. Apple continuously lost market share, and Microsoft Windows became the prevailing operating system with a WIMP-based graphical user interface.

Apple's current operating system is Mac OS 9. It is a direct descendant of the original version from 1984. With respect to the graphical user interface Mac OS 9, Microsoft Windows, and even the new Mac OS X build on the same principles that have been established for the first generations of Macintosh computers during the 1980s. Neither Apple, nor any other manufacturer of graphical user interfaces for personal computers has made an successful attempt to solve such obstacles in the human-computer interaction that have been described above. Nearly all concepts are faithfully copied from the original Macintosh interface design.

3.2 Human Factors

The historic discussion of graphical user interfaces has been mostly oriented on technical features. This section will focus on the human side of the relation between man and machine. Each of the three parts represents a different dimension.

First is the physical area. What implications for interface design can be deduced from the attributes of our human body? For instance a screen resolution of 72 DPI seems to be the lowest limit for legible text on a computer monitor. It is a task of this research field to perform experiments to validate such assumptions. The psychologist Paul Fitts has formulated a relation between hand motion and the size of the target, that will be presented here.

The next part gives an idea of Jean Piaget's theory of human learning. Knowledge about mental abilities is important to understand how humans transpose their perceptions into a mental model of the presented computer environment. A direct path leads from Piaget over Jerome Bruner to Alan Kay's and Adele Goldberg's work with children at Xerox PARC.

The third part tackles the notion of interactivity. It turns out that a dialogue or conversation between two people still has a different quality than an interactive dialog between human and computer. According to Joseph Licklider, Doug Engelbart, and Alan Kay, it is not the ultimate goal to replace one human participant with a computer. Personal computing is not about a conversation between two equals. But the special capabilities of the computer should augment and complement the human abilities for the benefit of the human user.

3.2.1 Fitts' Law

In the 1950s the experimental psychologist Paul Fitts has discovered a mathematical relation between the time it takes to acquire a target with the hand and the distance and size of the target area.²⁵ The relation can be expressed as

$$T_{pos} = c_1 + c_2 \cdot \log_2\left(2\frac{d}{w}\right)$$

where c_1 and c_2 are device dependant constants. d is the distance to the target of width w .

In the words of Stuart Card, Thomas Moran, and Allen Newell [Card et al. 83, p. 53]:

²⁵ The original papers that present Fitts' Law are *The information capacity of the human motor system in controlling amplitude and movement* by Paul Fitts in *Journal of Experimental Psychology* (Volume 47 p. 381-391, 1954) and *Information Capacity of Discrete Motor Responses* by Paul Fitts and J. Peterson Ibid. (Volume 67 p. 103-112, 1964).

[The] time to move the hand to a target depends only on the relative precision required, that is, the ratio between the target's distance and its size.

An obvious conclusion to be drawn for interface design is the rule to make distant items large, e.g. icons and buttons. Or, as Bruce Tognazzini says in *Tog on Interface*, «close targets are faster to acquire than far ones» [Tognazzini 92, p. 206].

Another direction to exploit Fitts' Law is to measure and compare the values of c_1 and c_2 for different graphical input devices. This opens the way to a qualitative analysis of mouse, touch screen, graphic tablet, etc.

Due to Ronald Baecker, Fitts' Law has first been applied in the field of human-computer interaction at Xerox PARC by Stuart Card, Thomas Moran, Bill English and Alan Newell in the 1970s [Baecker et al. 95, p. 470]. Their studies provided qualitatively founded guidance for interface design and laid the foundations for applied cognitive psychology. Card, Moran, and Newell developed a sound scientific framework that was published as *The Psychology of Human-Computer Interaction* [Card et al. 83].

3.2.2 Three Stages of Human Development

Jean Piaget spent his life working with children to figure out how they learn. His theory of learning explains why children of different age take a different approach in learning a new topic. Piaget argues that human development proceeds in a sequence of stages. The presentation here follows Alan Kay's paper *A Personal Computer for Children of All Ages* [Kay 72a], respectively his essay *User Interface: A Personal View* [Kay 90].

The *sensorimotoric stage* covers about the first 18 months. During this period the child's behavior is mainly based on reflexive actions. The child can distinguish between objects. The sensorimotoric stage is followed by the *preoperational stage* that lasts until four years. Speech starts. The child develops an understanding for size of objects, but volume and mass is still out of reach. The child plays and touches the objects physically. Between the age of four and eight comes the *concrete operational* or *visual stage*. The child paints images. Trial and error is the way how the child explores the world in this stage. The fourth and final stage of development is the *formal* or *symbolic stage*. It is characterized by logic, hypothesis and deductions, theories, and thinking in abstractions.

Jerome Bruner, also a psychologist, has repeated many of Piaget's experiments and confirmed the results. But he concludes a more far-reaching theory than Piaget's stage model of human learning. He structures cognitive abilities into a set of mentalities. The three main areas are the *enactive*, the *iconic*, and the *symbolic* mentality. To each mentality he finds a corresponding stage in Piaget's model. The early playful stage matches the enactive mentality. With David C. Smith's words, «Learning is accomplished by doing. A baby learns what a rattle is by shaking it. A child learns to ride a bicycle by riding one» [Smith 93]. The phase until about eight years is dominated by

the iconic mentality. «Learning and thinking utilizes pictures. A child learns what a horse is by seeing one or a picture of one», says Smith [Ibid.]. The logic and formal stage is in accord with the symbolic mentality.

The main conclusion drawn from Piaget's and Bruner's work is, that it doesn't make any sense to try to teach abstract mathematical concepts to children before they have reached the third stage of formal and symbolic reasoning. In consequence Seymour Papert's design of the programming language Logo is mainly driven by insights of Piaget's and Bruner's theories – as portrayed in *Mindstorms: Children, Computers, and Powerful Ideas* [Papert 80]. The concept of the turtle attracts children immediately because it is well designed to fit into the first stage of curiosity and play and at the same time the second stage of visual and iconic thinking as the turtle is used to draw images.

During the video lecture *Doing With Images Makes Symbols: Communicating with Computers* [Kay 87] Alan Kay points out that mentalities of human cognition do not supersede each other. If a new stage unfolds the previous stage remains active. It is a layered model of learning for children as well as for adults. Although adults tend to neglect the early stages as childish. This turns out to be imprudent, because the relation between practical experience, and a more figurative, iconic thinking on the one hand and logical reasoning on the other hand is the vivid source for creative thought. This is illustrated with a quote from Albert Einstein. His introspective description of how he thinks harmonizes in remarkable manner with Bruner's model of mentalities [Smith 93]:²⁶

The words of the language, as they are written or spoken, do not seem to play any role in my mechanism of thought. The psychical entities which seem to serve as elements in thought are certain signs and more or less clear images which can be 'voluntarily' reproduced and combined. [...] This combinatory play seems to be the essential feature in productive thought—before there is any connection with logical construction in words or other kinds of signs which can be communicated to others. [...] The above mentioned elements are, in my case, of visual and some of muscular type. Conventional words or other signs have to be sought for laboriously only in a secondary stage, when the mentioned associative play is sufficiently established and can be reproduced at will.

Einstein even refers to the enactive physical layer – «some of muscular type» – that plays an important role for his process of thought.

Software design should consider to serve all mental layers. The desktop metaphor and WIMP graphical user interfaces are successful, because they offer direct manipulation of content that corresponds to the visual and physical layer. Icons and pull down menus

²⁶ Smith took the quote from Jacques Hadamard's book *The Psychology of Invention in the Mathematical Field*, Dover Publications, New York, 1945, p. 142-143.

	Doing	with Images	makes Symbols
Piaget's stages	kinesthetic / sensorimotoric	visual	symbolic / formal
Bruner's mentalities	enactive	iconic	symbolic
Kay's interface design	mouse	icons, windows	Smalltalk
	know where you are, manipulate	recognize, compare, configure, concrete	tie together long chains of reasoning, abstract

Tab. 3.2 *Doing with Images makes Symbols*

are used to convey abstract concepts of directories and algorithms. Kay's slogan «Doing with Images makes Symbols» [Kay 90, p. 196] wraps up Piaget's stage model, Bruner's mentalities and their application for user interface design.

The mouse is a physical extension of the hand to touch and manipulate icons and windows on screen. They are graphical representations for otherwise abstract concepts. Alan Kay's programming language Smalltalk can be used to formulate ideas of all kind, to express relations between those ideas, and to logically infer new facts. Tab. 3.2 gives a summary of the three models.

3.2.3 Interactivity

Personal computer systems ought to participate in an interactive dialog with the user. Reflecting on the real sense of interactivity reveals a disenchanting situation of human-computer interaction. Steward Brand has conducted an interview with Andrew B. Lippman for his book *The Media Lab: Inventing the Future at M.I.T.* [Brand 87].²⁷

Lippman defines interactivity as «mutual and simultaneous activity on the part of both participants, usually working toward some goal» [Ibid., p. 46]. Supplementary he nominates five related corollaries.

Interruptibility is required in contrast to a just alternating behavior. Each participant is allowed to interrupt at any time. That does not mean that the active part turns immediately, but the interrupt should at least be acknowledged in a reasonable short period thereafter. Many so-called interactive computer systems miss the point of interruptibility. They offer simply an alternating flow of control.

After interrupting and acknowledging the interrupt it is a matter of *granularity* when the focus of activity can change to the other participant. For human conversation it is most likely a couple of words or a phrase. A sentence is polite but not common. A paragraph is too long for an interactive conversation. Then the conversation mutates to a

²⁷ The Media Lab has succeeded the Architecture Machine Group in 1985. It is also led by Nicholas Negroponte.

lecture. An interactive computer system should aim for smaller grain. Response times have come down from days when computers were operated in batch mode to fractions of seconds for PCs. But slow access to external resources or time consuming calculations make it necessary to think about granularity today.

Graceful degradation is Lippman's third corollary. It means the ability to handle requests that cannot be answered for the moment. They can and should be suspended without bringing the conversation to halt.

The border between *limited look-ahead* and *non-determination* is blurred. A conversation is interactive if it is open to unexpected events. In case of interacting with computers this calls for flexibility. The computer system should not be rigid and force its terms of operation on the user.

Interactivity is often claimed as a property for computer systems. Given Lippman's understanding of interactivity, those systems should be considered at most as dialogical systems. The active role changes from the user, who is formulating a command, to the computer for executing it, back to the user for the next command, and so forth. An interactive conversation of this kind falls apart, if the response time becomes too long. The user starts to enter a command during a phase when the computer is not ready to accept it. The provision of graceful degradation is applicable in situations where the computer cannot calculate an appropriate result for the given command. Many of the error and alert messages provided by current systems leave the user at a dead end – unsure of how to proceed. They do not offer any explanation or help to achieve the desired result.

Real interactive systems have to be more flexible and less rigid with respect to user interaction.

3.3 Windows, Icons, Menus, and Pointing Device

These are the main components of typical graphical user interfaces like Apple Macintosh and Microsoft Windows. WIMP²⁸ interfaces often come along with the desktop metaphor. This section takes a closer look at the pieces one by one.

3.3.1 Windows

Windows have a natural affinity to paper. They are rectangular framed areas on screen that can be moved around like their physical counterparts. Windows do overlap like real paper does – nearly everywhere in current graphical user interfaces.

In contrast to paper, computer windows can be resized, and the content area can be much larger than the actual window frame is. Horizontal and vertical scrollbars are

²⁸ 'Windows, Icons, Menus, and Pointing Device' is abbreviated as WIMP. Opposed to that, Ben Shneiderman defines WIMP as 'Windows, Icons, Mouse, and Pull-Down Menus' [Shneiderman 98, p. 207]. But this is not as general as the definition used here.

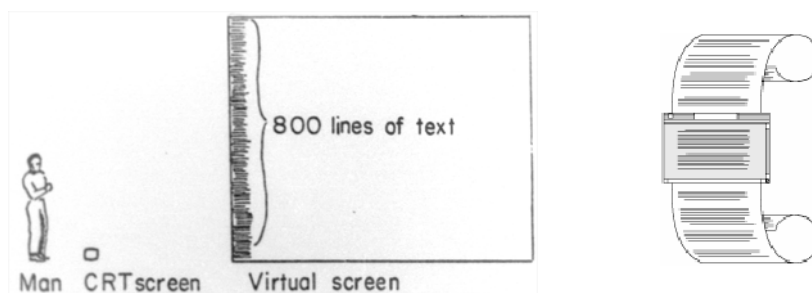


Fig. 3.14 Relationship between Window and Document. (a) Illustration by Alan Kay, 1969, (b) by Apple

deployed to move the content into the visible portion of the window. The metaphorical relation between windows and paper becomes weak with respect to the content displayed. A relation between window and document is more appropriate.

Especially if paper is represented inside of windows, the different angle of the relationships becomes obvious (cf. Fig. 3.13 on page 75). Doug Engelbart argues [Engelbart 88, p. 219]:

One way to look at how to use the computer to help work with documents would be the possibility of a conventional word-processor approach. It's a very straightforward way. The orientation is to simulate paper on a display, targeted solely to produce hard copy. An considerable advantage in many situations, but it is a very anemic example of what the [Augmenting Human Intellect] framework promises.

Media like paper and book have been with us for several hundred years. They have shaped our idea of how text has to be presented. It needs to be considered how the new medium computer should deal with our expectations.

In a document-centric environment windows can overlap each other in random order. This solves the dilemma of preemption between application programs, because the user can switch to different tasks with ease. Apple Macintosh introduced a process-based approach. That is, a process can handle several open document windows. If any of those is activated, all windows come front [Ludolph/Perkins 98, p. 19]. This behavior minimized the chance that other windows of interest remain partially visible. Microsoft Windows has a mode where a window opens for an application and the documents are displayed as sub-windows within the main application window. This has a similar effect on the user as the Macintosh model. Application programs dominate the user model. The user thinks in terms of programs instead of documents.

For some windows it is desirable to protect them against being obscured by other windows. A layer of tool windows has been introduced for this purpose. Tool windows hover on top of all other windows, and can therefore only be covered by other tool windows, but never by windows that belong to the document layer. Unfortunately tool

windows reduce the effective space left over for document windows. They have to be utilized with care in order not to increase the mess of a cluttered desktop model.

3.3.2 Icons

Icons are used in three different roles. They can represent objects. They can depict commands or actions. And they can be used as signs. As signs they communicate a message in a non-textual manner. The role of traffic signs is similar to those icons used for alert message boxes. Their purpose is to draw the user's attention to a special incident.

Icons that represent commands and actions can often be found in tool bars. They are operated with a single mouse click and change a mode or trigger a command. For instance the Web editor Adobe GoLive has a tool bar with command icons for text alignment and style (cf. Fig. 3.15 and Fig. 3.16). Image processing applications usually have tool bars with icons to change the current mode of editing, e.g. icons for PENCIL, PAINT BUCKET and ERASER tools.

Icons that symbolize objects are the most complex kind of icons. David C. Smith has adopted the term icon for interface design in the mid-1970s at Xerox PARC (cf. 3.1.6 Xerox Star (p. 65)). Icons that represent documents and folders bear a quality that makes it easy to manipulate abstract data structures. Alan Kay's slogan «Doing with Images makes Symbols» conveys the essence of icons (cf. Tab. 3.2 on page 80).

3.3.3 Menus

Menus are means to present a hierarchy of commands to the user. In graphical user interfaces the menu titles are typically lined up horizontally. A click to a title reveals the menu items for that menu. This kind of interaction led to the name pull-down menus. Vertical alignment is rarely used for menus bars. An exception can be found in NeXT-Step (cf. Fig. 2.12 on page 33).

The best position for the menu bar is a wonderful exercise in Fitts' Law. The graphical user interfaces of Apple Lisa and Macintosh put the menu bar at the top edge of the screen (Fig. 3.16). Opposed to that, Microsoft Windows puts the bar inside the document window, respectively inside the application window. Even if the application window is maximized on screen the menu bar is not at the top edge of the screen (cf. Fig. 3.15). The slightly different design leads to a huge difference in usability. The performance of the Macintosh approach exceeds Microsoft's.

The explanation to this phenomenon is Fitts' Law. Despite the fact that the menu titles have the same size on both platforms, objects on top of the screen have a virtually unbounded height. It does not matter how fast the user arrives at the menu bar. She cannot miss the correct vertical mouse position, because the movement is limited by

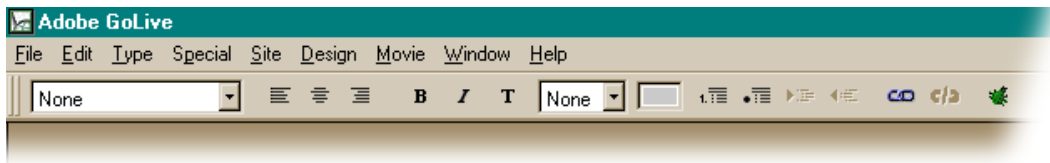


Fig. 3.15 Menu and tool bar layout of Adobe GoLive on Microsoft Windows

the top edge of the screen. No vertical fine tuning is necessary to position the mouse pointer over the desired menu title [Tognazzini 92, p. 201] (also *The Humane Interface – New Directions for Designing Interactive Systems* by Jef Raskin [Raskin 2000, p. 94]).

A different form of menus are pop-up menus. They were originally developed for the Smalltalk environment at Xerox PARC. One of the mouse buttons is reserved to open the menu wherever the click occurs. The menu items of pop-up menus can be tailored to the context given by the mouse position. For example an EMPTY TRASH command needs only to be offered if the pop-up menu has been opened with a click on the trash can. Context sensitive pop-up menus – or context menus for short – are part of the graphical user interfaces Microsoft Windows and Apple Macintosh.

If the number of commands becomes too high to fit into a straight menu structure hierarchical menus are a way out. Some menu items are used as menu titles themselves. They reveal a second level next to the first column of menu items.

Based on Fitts' Law, context menus would seem to be optimal. No mouse movement is necessary to open a context menu. But Bruce Tognazzini refers to experiments that show that this advantage is lost against a menu bar that is along the top edge of the screen. He argues, that the effect of the top edge on the first step of choosing the right menu cannot be beaten by context menus. After the context menu has opened no edge guides the mouse cursor to increase the virtual size of the menu items [Tognazzini 92, p. 203].

Considerations of consistency have another important impact on menu design. If the order of menus and menu items is consistent across the range of application programs the structure becomes predictable for the user. It can be remembered by means of spacial organisation. For example the second menu is always the Edit menu, that contains CUT, COPY and PASTE commands for the clipboard. Spacial consistency in user interface design is a precondition to reach a habitual interaction mode. That is, the user can act with confidence in the software environment and make full use of motor memory.

3.3.4 The Mouse and other Graphical Input Devices

Jef Raskin defines the graphical input device (GID) as «a mechanism of communicating information, such as a particular location or choice of object on a display, to a system»



Fig. 3.16 Menu and tool bar layout of Adobe GoLive on Apple Macintosh

[Raskin 2000, p. 34]. The mouse is just one possible graphical input device. It was invented by Doug Engelbart and Bill English in 1963. They did additional tests with the light pen, the joystick, and various other experimental devices [English/Engelbart/Berman 67]. They found the mouse to be the fastest device and decided to use it for NLS. Other graphical input devices are for example touch screens, graphic tablets, track balls, and track pads.

The article *A Morphological Analysis of the Design Space of Input Devices* [Card et al. 91] – by Stuart Card, Jock Mackinlay, and George Robertson – provides a taxonomy for input devices. Referring to the model, the mouse is a relative pointing device. The position of the pointer is controlled by movement rather than by absolute position. The mouse is also a linear input device, opposed to a rotary device like the joystick. The latter generates angle values along two axes, that can be used absolutely as position and relatively as velocity for the pointer on screen.

The second physical property that is employable for graphical input devices is force. An application can be found in some notebook PCs where space is a critical issue. An extra button is placed between the keys G, H, and B. Horizontal force to this button is transposed to movements of the pointer on screen. Another example for the application of force would be a graphic tablet with a pressure sensitive pen.

Summarizing, the design space for graphical input devices is made up of all possible combinations of absolute and relative, linear and rotary, and position and force.

Raskin continues to define the primary button of a graphical input device as the GID button. A click «is to position the GID and then to tap the GID button.» [Raskin 2000, p. 34]. Definitions for drag, and double click actions follow analogously.

The previous section on menus has shown that graphical input devices should have an tremendous influence on the design of graphical user interfaces. The mouse is a Fitts' Law device – it can be best modeled by a variation of Fitts' Law, as Stuart Card and Thomas Moran have shown in *User Technology: From Pointing to Pondering* [Card/Moran 86, p. 496]. The importance of the edges of the screen follows immediately. Other devices have different characteristic qualities that affect the usability of the entire system. For instance, all graphical input devices that are operated with a finger instead of the arm – i.e. track pads – are less sensitive for the Fitts' Law effect at the edges of the screen.

3.4 Provisions for the Future of the Desktop Model

Joseph Licklider's vision was the symbiosis between man and machine. Early systems – like Sketchpad and NLS – have reached a level of intimacy with the user that is still remarkable. But the effort to learn the correct handling of those systems was a stumbling block for users. The five-finger chording keyset is a typical example. Once you have learned how to handle the device the usage can become habitual. It can be operated with ease and feels natural. But the chording keyset is not approachable for a novice user. It is most likely to make many errors in the beginning, because the keys are not labeled like the keys of a standard keyboard. Furthermore, the user has permanently to keep in mind in which mode of the five cases the keyset currently operates. This is an extra load for the memory.

The mouse is not an intuitive input device either. But the user becomes familiar with it very soon.²⁹ Alan Kay has shown up the relation between the mouse and Jerome Bruner's notion of the enactive mentality. The mouse evolves to a seamless extension of the hand to interact with objects on the screen. The term direct manipulation has been coined in 1983 by Ben Shneiderman for this style of interacting with the computer (e.g. in *Designing the User Interface* [Shneiderman 98, p. 185]).

The targeted user plays a key role in interface design. Back when computer systems were used only by a few computer experts, not much attention was paid to well designed user interfaces. The user was required to adopt to the machine. Things changed with the development of the Xerox Star 8010 Information Systems. Now the intended user is an average office worker. The desktop metaphor was invented to cover the technical details of the computer in order to create an friendly and familiar working environment. The user's mental model is based on everyday objects like paper, documents, and folders. This world of office metaphors helps the user to gain an practical understanding of the system.

Despite the promising approaches of NLS, Xerox Star, and Apple Lisa we are facing today computer systems that are complex and awkward to use – for office workers, as well as for scientists. All current PCs fall short compared to the original vision of personal computing.

The next three sections will discuss the relation between documents, application programs, and the filing system. The difference between a file system and a filing system can be best clarified with the following analogy: What a file is to the file system, is a document to the filing system. Both, document and filing system, belong to the user model. File and file system are technical implementations to physically store the

²⁹ In *Intuitive equals Familiar* [Raskin 94b] Jef Raskin argues that 'intuitive' is often used inaccurately by non-HCI specialists. Very few systems are intuitive in the sense to be immediately usable without explanation and training. A better denotation of an intuitive interface would be the quality to use «readily transferred, existing skills» [Ibid.].

content of documents. The critical comments on the filing system will be followed by a recapitulation of Xerox Star's and Apple Lisa's document-centered design approach. This chapter will close with some reflections on metaphors and the chances of the medium computer.

3.4.1 Filing

The desktop metaphor allows to put documents on the desktop, and to file them in folders. Folders can also reside on the desktop, or they can be filed away in volumes. As the real screen estate is limited the desktop metaphor is extended to a hierarchical structure, i.e. folders can be stored inside of other folders, and so on. This approach worked quite well as long as the number of items was in the range of hundreds. But it does not scale to thousands or ten thousands of files. The hierarchy of folders was not invented for graphical user interfaces. It was adopted without reconsideration from the model of hierarchical directories of previous generations of computer systems. In a talk – given at the symposium *Engelbart's Unfinished Revolution* in December 1998 – Ted Nelson argues with sarcasm [Nelson 99b, p. 4],

we've got hierarchical directories which we accidentally invented like in 1947 – 'where are we going to put all this stuff?' 'well, lets make a file' – they named it files, right. So, the hierarchical assumption has passed on to us which assumes that there is no overlap between things we do. You work on one thing, then you finish that, put it away neatly and then you work on something else. {{laughter}}. right. There is no overlap, there is no interpenetration, projects are never redefined, we don't have to change our terminology once we've started...

The consequences of this decision penetrate the way how the desktop model works today. Users have problems to keeping track of their documents. They do not remember, for example, where they have saved the letter to a specific customer a couple of months ago. The only handle to a document is its file name and perhaps its position in the hierarchy of folders. Both criteria don't scale very well. Furthermore, they are also weak with respect to the user's real office experience.

File names are artificial and do not work the way one would expect. Jef Raskin describes the situation when file names are created in the following way [Raskin 2000, p. 118]:

File names are bothersome when you are about to save work, because you have to stop in the middle of your activity, which is trying to store your work away, and invent a file name. Creating names is an onerous task: You are required to invent, on the spot and in a few moments, a name that is unique, memorable, and within the naming conventions of the system you are using.

Names for folders and their position in the hierarchy are generated in similar fashion. What sounded reasonable in the moment of naming, is unintelligible when the item needs to be retrieved.

Means to identify files other than by name are underdeveloped in the present desktop model – although some promising approaches have been taken in the history of personal computing. Those include the employment of visual qualities. SDMS' Data-land is founded on the idea of spacial arrangement of items. The remnants of this approach lead to the persistent arrangement of icons in WIMP interfaces like the Lisa Desktop Manager and the Macintosh Finder (cf. top left window in Fig. 3.13 on page 75). Visual clues for documents are also rarely used. SDMS displayed facsimile – scaled down images of the original content. Opposed to that, Xerox' desktop model introduced icons – abstract graphical images with a well defined semantics. The role of thumbnails as a substitute for icons should be further investigated.

Other options to identify documents can be based on content and context. «The content of a text file is its own best name», says Jef Raskin [Ibid., p. 119]. Tools to easily perform full-text searches on a given set of files can be an efficient way to retrieve files. Some key words are often sufficient to reduce the resulting set of possible documents to a practicable number of files.

Context calls for new interface elements. Documents are related to each other in terms of tasks. For instance the creation of this present document involves several hundred files: e-mails, HTML pages, downloaded articles in PDF format, audio tracks, and archives to backup previous versions – and finally the layouted text documents themselves. Books, magazines, and video tapes – although not in digital form – belong to the project as well. Representing this network of information with a hierarchical folder structure is not sufficient. New concepts are needed to express such relations.

Some new objects for the desktop are proposed by Bruce Tognazzini in *Tog on Software Design* [Tognazzini 96, p. 196]. *Intelligent folders* should continuously compile the set of documents that match certain criteria. *Archives* look and behave like standard document icons. But they can additionally store the historical chain of versions for a document. *Piles* are informal clusters of documents. Tognazzini explains in *Scaling Information Access* [Tognazzini 98],

You start a pile by dropping one document icon on another. You may then continue to add icons to your heart's content. The resulting pile is easy to read: click on it and drag up and down to see thumbnails of each document instantly appear adjacent to the pile.

It should be easy to collect items that might be of interest for some vague goals. Finally, a *Scrapbook* is Tognazzini's answer to keep track of the flock of documents for a task related project – like the one mentioned above.

The key to develop new filing concepts, that are closer to the user, is the abstraction of the hierarchical file system. The technical layer should be concealed from the user's

perspective, as other technical aspects have been covered by the desktop metaphor before.

3.4.2 Document-Centered Design

Xerox Star and Apple Lisa are the two main systems that follow a document-centered design approach. Jeff Johnson explains the role of programs for the Star in *The Xerox Star: A Retrospective* [Johnson et al. 89, p. 11]:

The applications included in the system were those that office professionals would supposedly need: documents, business graphics, tables, personal databases, and electronic mail. The set was fixed, always loaded, and automatically associated with data files, eliminating the need to obtain, install, and start the right application for a given task or data file. Users could focus on their work, oblivious to concepts like software, operating systems, applications, and programs.

Despite the fact that Apple Lisa has an open architecture to load new application programs, the user experience is very similar to the Xerox Star system.

Limitations of the hardware in the early history of Macintosh computer lead to the dominant application-centric user model, that we are facing today (cf. 3.1.9 Apple Macintosh (p. 74)). Nevertheless, the document-centered design approach bears several advantages for the user. The most important is the fusion of document file and document window. It is simply not possible to have an unsaved document without an associated file. New documents are created with the help of stationery pads in the filing system, opposed to a NEW command inside of application programs. The file name is generated by the system and can become subordinate. This is coherent with the previous section on the filing system, if other means to identify documents are employed.

In a document-centered software environment, documents are not related exclusively to one application program. The document is the central element. Programs are used as tools to contribute pieces of content to the document. One tool might be a text editor, while another tool provides imaging functionality. The unrestricted composition of all those parts constitutes the document. The operating system has to define a software interface between the tools and the documents. Furthermore, it is necessary to agree on open document formats.

Such an environment would be more flexible than the application-centric model. The user is free to access tools wherever they are needed. Documents would belong to the user, and can be edited with several tools. Monolithic application programs and proprietary document formats restrain the possibilities of personal computing.

3.4.3 User Illusion

Graphical user interfaces have the power to create visual and interactive environments for abstract data spaces. The original term – before ‘physical-office metaphor’ or ‘desktop metaphor’ became common – was «user illusion». The term was coined at Xerox PARC in the 1970s [Tognazzini 96, p. 291]. Metaphorical concepts are useful for users to build an initial understanding of the computer system. On the flip side they narrow the perspective to functions that have no counterpart in the real world. For example, drag and drop of a document to a folder moves the object, whilst a drag and drop to a printer icon triggers a printing job. But does it also move the document icon into the printer? The system’s behavior is not guarded by the metaphor anymore. Users are also confused if items are rearranged by the computer. They expect persistence of spacial properties. Therefore a concept like Bruce Tognazzini’s intelligent folders tests the limit of the desktop metaphor.

Other interaction techniques use no metaphors at all. Point & shoot, for instance, is a technique to create hyperlinks in Adobe GoLive. A line is pulled out to connect link marker and link target with each other. It is plain abstract graphics and has no metaphorical correspondence with the real world [Müller-Prove 99].

User illusion is a more sweeping idea. The user does not have to keep in mind whether an aspect of an object is in accord with the metaphor or not. Alan Kay says, that the term has «clear connotations to the stage, theatrics, and magic» [Kay 90, p. 199]. But it should be an understandable kind of magic. The plain representation of paper on computer screens falls short compared to the possibilities of the medium. Kay argues, «it is the *magical* part that is all important and that must be most strongly attended to in the user interface design» [Ibid.]. Ted Nelson emphasizes this idea. In *The Future of Information* he suggests to consider software design as a new form of movie [Nelson 97a, p. 16]

Movies are systems of events on a screen that affect the heart and mind of the viewer. Software—even office software— is a system of events on a screen that affect the heart and mind of the participant, and interact with the participant—who is no longer a mere viewer.

That means that software is exactly what movies are, and more. Software is not just a branch, but the *generalization* of movies, not metaphorically but literally.

Theater, movie, and magic also point to consistency and aesthetic integrity in human interface design. The value of consistency is a gain in familiarity and predictability of software environments.

According to the *Macintosh Human Interface Guidelines*, aesthetic integrity means that «information is well organized and consistent with principles of visual design» [Apple 95, p. 11]. A graceful appearance is desirable for graphical user interfaces.

4 Beyond the Desktop

Today, many personal computers are connected to the Internet. Communication services like electronic mail are in everyday use, and the World Wide Web definitely has characteristics of a new medium.

Tim Berners-Lee and Robert Cailliau used the NeXT application framework to develop the first program to read and edit HTML pages in WYSIWYG mode [Gillies/Cailliau 2000, p. 190]. NCSA Mosaic by Marc Andreessen was not able to edit HTML pages, but it was available for X-Windows, Microsoft Windows and Apple Macintosh. This led to a worldwide distribution of this Web browser program. Berners-Lee's World-WideWeb/Nexus and Mosaic follow similar principles. The programs utilize graphical user interface elements of the WIMP environment. Web pages correspond to windows. But in the case of Mosaic, the current page is replaced by the targeted page of an activated hyperlink. The shifted meaning of 'browsing' depicts this behavior.

None of these efforts put any work into the user experience, to try to integrate the Web as a new dimension into the desktop model. Browsers are ordinary application programs. This is true for early browsers as well as for current versions of Netscape and Microsoft Internet Explorer. But inside the browser windows a new kind of graphical user interface unfolds.

The first section of this chapter will show how the WIMP desktop model and the Web interface fall apart.

4.1 Web GUI meets Desktop GUI

The Web interface and the typical desktop interface are both graphical user interfaces. But this is nearly all that these two have in common. Some important aspects of WIMP have been discussed in section 3.3 Windows, Icons, Menus, and Pointing Device (p. 81). Analyzing the Web interface on the basis of WIMP shows that the Web should not be considered as such.

Browsers use windows to display Web pages. The relation between window and Web page is in accord with the expectations of the user. The window displays a document, even if it is a remote one. The user can resize the windows, which causes the content to be recomposed. No assumption is made on the screen size of the client's machine; therefore the maximum size of the window is unknown and the Web page aims to fit any window size. A click to a hyperlink loads the next page into the browser window. The history function keeps track of the path how the pages were reached. The path is a list

of URLs. Control elements are provided to navigate forward and backward in history. If the user closes the window this information is deleted.

Icons are not used in Web interfaces. To be more specific, icons that symbolize objects with a well defined semantics are not utilized, because direct manipulation is not applicable in Web context. Response times would be too long, and the technical side still suffers under the original statelessness of the HTTP protocol. Nevertheless, icons as signs and icons as labels for command buttons and image links are frequently used.

Menus are not part of Web interfaces. A Web page currently has neither control over the standard menu bar, nor of any context menu. These menus always offer standard commands of the browser interface. Unfortunately, the current Web site cannot add custom items to the menus. A couple of examples can be found in section 2.3.4 Browser (p. 45). Some Web sites misuse popup controls for navigation purposes. Popup control items are intended to make a choice, for instance a country should be selected out of a list of possible countries. Selecting commands with popup controls is in conflict with the common meaning of this item in desktop application programs.

With respect to the pointing device, hardly any difference between the desktop model and the Web can be seen. It should just be mentioned that a Fitts' Law effect on a screen edge cannot be achieved for Web interfaces (cf. 3.3.3 Menus (p. 83)). Browser windows are not near to any edge of the screen.

Limited use of icons and no use of menus disqualify Web interfaces from being WIMP interfaces.

The Web interface also introduces its own kind of interaction mode. For example, the dominant way to use the graphical input device is to click on a hyperlink. One click is sufficient to trigger the link. In the desktop world a single click usually selects the item. Only a double click triggers the item to open. Text selection is another field of inconsistency. A piece of text usually can be selected by a click-drag action. The same action initiated on a textual hyperlink starts a drag action of the URL.

Forgiveness is a quality of well designed user interfaces. The Macintosh interface guidelines urge the application developers to implement UNDO functionality wherever possible. The guideline reads [Apple 95, p. 10]:

People need to feel that they can try things without damaging the system; create safety nets for people so that they feel comfortable learning and using your product.

It is too easy to damage valuable user data with a Web browser. Stability of the product is one factor, but a simple action like closing a browser window irrevocably clears the history for that window.

This passage can only give some examples to demonstrate that even the interaction modes of Web interfaces and the conventional desktop model are in conflict with each other. The misuse of familiar control elements, like popup controls, results in a mixed up user experience. The user has always to keep in mind whether she is in Web mode or desktop mode. The user is confused. Errors are more likely under such conditions.

4.2 Provisions for the Future

What can be done to bridge the gap between the desktop environment and the Web? The sections 2.3 Provisions for the Future of the World Wide Web (p. 44) and 3.4 Provisions for the Future of the Desktop Model (p. 86) show remarkable similarities. The structural problems of the Web seem to have direct counterparts in the field of desktop interfaces, and vice versa. The main three areas are: reliable and efficient identification of documents, the demand for new concepts for groups of documents, and the reign of technical solutions.

Documents on a local desktop and Web pages have in principle the same structure. They are files that contain content that matters to the user. Of course, file formats are arbitrary. Different application programs use proprietary formats. This challenge for the application-centric model has been tackled for the Web. HTML is standard for Web pages. The format is platform independent and can be edited with any text editor, although WYSIWYG editing is more comfortable for the user. Data of arbitrary type can be encoded in XML structures. The Extensible Markup Language (XML) is a subset of the Standard Generalized Markup Language (SGML). Applications of XML are powerful enough to express any data structure in an open and flexible manner. XML files share platform independence with HTML files.

The robust identification of document files is unsolved for both domains. They suffer from weak concepts to identify entities of content. Hyperlinks between Web pages can break because the URL of the target page might change. Retrieval of documents fails, because the rationale behind the file name turns out to be totally unintelligible. The documents are lost in the hierarchy of folders.

New methods are needed to reliably resolve documents – on local file servers as well as on remote Web servers. Tim Berners-Lee calls it the concept of **location independence** and explains [Berners-Lee 99, p. 159],

the appearance of the information and the tools one uses to access it should be independent of where the information is stored [...]. Whether they are hypertext pages or folders, both valid genres of information management, they should look and feel the same wherever they physically happen to be. Filenames should disappear; they should become merely another form of URI. Then people should cease to be aware of URIs,

seeing only hypertext links. The technology should be transparent, so we interact with it intuitively.

It is not desirable that everything looks like a Web site – especially not under the current usability flaws of the Web. The point here is that a layer has to be installed that is responsible for the reliable access to any document. It serves as an abstraction of documents from their storage. The user interface follows thereafter. Tools that scale from the personal desktop environment to the world-wide information space give access to a seamless field of data. Consistency in user experience between the extremes is the most important factor.

The second correspondence is the lack of flexible concepts to group documents. Once we have the layer of abstraction between files and documents, once we have a technical difference between directories and folders, new powerful concepts can be implemented for different kinds of relations between objects. In *Cleaning up the User Interface* [Berners-Lee 97] Tim Berners-Lee proposes a new conception for folders. They should evolve into a new type of hypertext document. Placing a document into a folder would just establish a special kind of relationship between the document and the folder-document. That means that a folder would be represented by an XML data file. The membership of a document to a folder can then expressed as a hyperlink from the folder's XML data to the document.

The design and application of those new folders is independent from the technical implementation. Section 2.3.2 Groups of Nodes (p. 45) recapitulates the purpose of groups for hypertext nodes. Sequences and clusters represent guided tours and, for instance, Web sites as a whole. The corresponding section in the chapter on graphical user interfaces 3.4.1 Filing (p. 87) outlines some concepts for extended folder functionalities. Aggregation of hypertext nodes and grouping of conventional desktop documents should become the same.

Limitations on the technical side gave preference to the application-centered design model. The document-centered approach, although beneficial for the user, did not succeed. Applications for the Internet face a similar situation, because they have to follow the protocol first. Therefore Tim Berners-Lee argues for **protocol independence** [Berners-Lee 99, p. 160],

The next step would be protocol independence. Right now, every time I write something with a computer, I have to choose whether to open the “electronic mail” application or the “net news” application or the “Web editor” application. The mail, news, and Web systems use different protocols between computers, and effectively, I am being asked to select which protocol to use. The computer should figure this out by itself.

The user should have full sovereignty on her documents. She should be free to do anything she likes, whether it is editing, high quality printing, sending the document

to a friend, publishing it on the Web, or filing it at several places at the same time. It should be easy to put the document in context. Robust hyperlinks to and from any part of the document should be possible. Also vague relations like Tognazzini's piles or spatial proximity should be supported by new grouping concepts. It does not matter, where the resources actually reside. Web pages and local documents can be intermixed without any restrictions. Surfing the Web and accessing files on a local hard drive should have the same user experience. Larry Tesler's idea of modelessness has to be applied to the two dogmatic competing modes Web and Desktop.

5 Synopsis

The original vision of Vannevar Bush, Joseph Licklider, Ted Nelson, Doug Engelbart, and Alan Kay is a computer that supports the user in cognitive processes – a personal dynamic medium for creative thought. In variation of Isaac Asimov’s first law of robotics,³⁰ Jeff Raskin writes forty years after Licklider’s article on *Man-Computer Symbiosis* the first law of interface design. It reads [Raskin 2000, p. 34]:

Any system shall not harm your content
or, through inaction, allow your content to come to harm.

The necessity to formulate such a law demonstrates how far away we are from the original visions.

The present thesis has shown that the foundations of hypertext are already rooted in the 1960s. Doug Engelbart, Andries van Dam, and Ted Nelson, implemented the first systems that are capable of hyperlinking. The 1970s were relatively quiet. But with the advent of personal computers in the 1980s, a lot of new hypertext programs were developed. They utilized WIMP concepts on the Apple Macintosh, on powerful workstations, and later on Microsoft Windows for IBM-PCs.

The foundations for the present form of graphical user interfaces were in the 1960s and 1970s. This thesis recalls the invention of the mouse by Doug Engelbart and Bill English, overlapping windows and popup menus by Alan Kay, and icons by David Canfield Smith. The parts were assembled in a consistent way for the Xerox Star in the second half of the 1970s. Extensive user studies, task analysis, and applied cognitive psychology produced evidence for the development of the desktop metaphor. These efforts can be rated as the first emergence of user-centered design. The industry needed a few attempts until the Apple Macintosh defined the standard for graphical user interfaces as we know them today. Since the late 1980s, the pace of innovation on the field of user interfaces for PCs came to a halt. The section on filing systems has shown that the dominating operating systems with graphical user interfaces do not offer sufficient methods for the user to cope with thousands of objects (cf. 3.4.1 Filing (p. 87)). The concept of direct manipulation and the paradigm of the desktop metaphor do not scale to the vast amount of items we are managing today. They were right in the beginning – today they are inadequate. Innovation is necessary to regain control of our files.

The situation became even more complicated with the tremendous success of the World Wide Web. The user interface for the Web, that got momentum in the early

³⁰ Isaac Asimov’s first law of robotics is «A robot shall not harm a human, or, through inaction, allow a human to come to harm.» The law is taken from Asimov’s science fiction novel *I Robot* (Bantam Books, New York, 1977).

1990s, unfortunately lacks a profound approach of user-centered design. No one felt responsible to start an attempt comparable with that of Xerox PARC's research for the Star computer or Apple's research for Lisa and Macintosh. As a result the Web interface was never integrated into the desktop environment.

A solution can be found by reflecting upon the core values of hypertext and graphical user interfaces. The following considerations are in accord with the present thesis.

The fundamental idea of hypertext is the relation between different texts. The common means to express such structures are hyperlinks. In *As We Should Have Thought* [Nürnberg/Leggett/Schneider 97] – another pun on the title of Vannevar Bush's article *As We May Think* – Peter Nürnberg, John Leggett, and Erich Schneider criticize linking for two reasons [Ibid., p. 1]:

Firstly, linking implies a certain kind of structural paradigm, one in which the user [...] links information together for purpose of navigation. [...]

Secondly, linking implies the primacy of data, not structure.

Navigation is important, but it should not be the only purpose of hypertextual structures. Especially section 3.4 Provisions for the Future of the Desktop Model (p. 86) has shown the need to express various kinds of relations in a flexible way. Neither hierarchical nor navigational structure should be imposed on the data.

The second argument opens the discussion to put structure, opposed to data, into the center of computing. At least, structure should not be considered as a second-order attribute to data.

Graphical user interfaces have the potential to convey such structures. Humans are able to understand abstract concepts, that go beyond the desktop metaphor. With Ted Nelson's words [Nelson 97a, p. 21]: «The fundamental information problem is to keep track of ideas, and represent them, accurately.»

APPENDIX

Acronyms

ACM	Association for Computing Machinery
AFIPS	American Federation of Information Processing Societies
API	Application Programming Interface
ARC	Augmentation Research Center at SRI
ARPA	Advanced Research Project Agency
ASCII	American Standard for Coded Information Interchange
A/UX	Apple UNIX
BBN	Bolt, Beranek & Newman Inc. – the company that built the IMPs
CAD	Computer Aided Design
CERN	Conceil Européen pour la Recherche Nucléaire (European Organization for Nuclear Research)
CEO	Chief Executive Officer
CGI	Common Gateway Interface
CHI	ACM Conference on Computer Human Interaction
CMU	Carnegie Mellon University, Pittsburgh, PA
CRT	Cathode Ray Tube
CSCW	Computer Supported Cooperative Work
CSL	Computer Science Laboratory at Xerox PARC
CSS	Cascading Style Sheets
CTO	Cybernetics Technology Office, division at DARPA
DARPA	Defence Advanced Research Project Agency
DDE	Dynamic Data Exchange service on Windows
DoD	Department of Defence
DOS	Microsoft Disk Operating System
DPI	Dots per Inch
DTD	Document Type Definition
DTP	Desktop Publishing
DVP	Harmony Document Viewer Protocol
EARS	Ethernet-Alto-RCG-SLOT
ECHT	European Conference on Hypertext
ECICS	European Conference on Integrated Interactive Computing Systems
FJCC	AFIPS Fall Joint Computer Conference
FLEX	Flexible Extendable
FRESS	File Retrieval and Editing System
FTP	File Transfer Protocol
GID	Graphical Input Device
GREP	Get Regular Expression
GUI	Graphical User Interface
HCI	Human Computer Interaction
HCIL	Human Computer Interaction Laboratory at the University of Maryland
HES	Hypertext Editing System
HTF	Hyper-G's Hypertext Format
HTML	Hypertext Markup Language

HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IEEE	Institute of Electronic and Electrical Engineers
IICM	Institute for Information Processing and Computer Supported New Media at Graz University of Technology
IMP	Interface Message Processor – the router of the ARPANet
INTERCHI	International ACM Conference on Computer Human Interaction
IPTO	Information Processing Techniques Office, division at ARPA
IRIS	Institute for Research in Information and Scholarship at Brown University
LAN	Local Area Network
LASER	Light Amplification by Stimulated Emission of Radiation
LISP	List Processing Language
LRG	Learning Research Group at Xerox PARC
MIT	Massachusetts Institute of Technology
MPEG	Moving Picture Encryption Group
NASA	National Aeronautics and Space Administration
NCSA	National Center for Supercomputing Applications
NLS	nN-Line System
NNTP	Network News Transfer Protocol
OHS	Open Hypermedia System
OOP	Object-Oriented Programming
OS	Operating System
OWL	Office Workstations Ltd.
PARC	Xerox Palo Alto Research Center
PC	Personal Computer
PDF	Adobe Portable Document Format
RADC	US Air Force's Rome Air Development Center
RAND	Research Associates for National Defence
RCG	Research Character Generator
SAGE	Semi-Automatic Ground Environment
SDD	Systems Development Department at Xerox
SDMS	Spatial Data Management System
SDS	Scientific Data Systems
SGML	Standard Generalized Markup Language
SID	Statement Identifier in NLS
SILK	Speech, Image, and Language understanding, all driven by Knowledge bases
SJCC	AFIPS Spring Joint Computer Conference
SLOT	Scanning Laser Output Terminal
SRI	Stanford Research Institute
TCP/IP	Transfer Control Protocol/Internet Protocol
TIES	The Electronic Encyclopedia System (as in Hyperties)
UID	Globally Unique Identifier (Dexter Model)
UIST	ACM Conference on User Interface Software and Technology
URI	Universal Resource Identifier
URL	Uniform Resource Locator
UUCP	Unix to Unix CoPy
W ₃ C	World Wide Web Consortium

WAIS	Wide Area Information Servers
WebDAV	Web-Based Distributed Authoring & Versioning
WIMP	Windows, Icons, Menus and Pointing Device
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XSL	Extensible Style Language

Software

Amaya	Charles McCathieNevile / et al. ³¹ ; W ₃ C, 1986 until today
Bravo	Charles Simonyi / Tom Maloy: Xerox PARC, 1974-1976
Bryce	Eric Wenger / Kai Krause / Phil Clevenger: MetaCreations Corp., 1994-1999
Concordia	Janet H. Walker: Symbolics Inc., 1985
Document Examiner	dto.
Finder	Bruce Horn / Steve Capps: Apple Computer, Inc., 1982 until today
FrameMaker	Frame Technology Corp., 1986-1995; Adobe Systems Inc., 1995 until today
FRESS	Andries van Dam: Brown University, 1968-mid-70s
GoLive	GoLive Systems, 1996-1998; Adobe Systems Inc., 1999 until today
Guide	Peter Brown: University of Kent at Canterbury, 1982 (marketed by OYce Workstations Ltd. since 1986)
HES	Andries van Dam / Ted Nelson: Brown University, 1968
HyperCard, HyperTalk	Bill Atkinson / Dan Winkler: Apple Computer, Inc., 1987-mid-90s
Hyper-G/HyperWave, Harmony	Hermann Maurer / Frank Kappe / Keith Andrews: IICM at Graz University of Technology and University of Auckland in New Zealand, early 1989-1996
Hyperties	Ben Shneiderman / Dan Ostroff: HCIL at University of Maryland, 1983 (marketed by Cognetics Corp. since 1987)
Intermedia	Nicole Yankelovich / Norman K. Meyrowitz / Paul Kahn / Bernard J. Haan / Victor A. Riley / James H. Coombs: IRIS at Brown University, 1985-1990
Internet Explorer	Microsoft Corp., 1995 until today
Lisa Desktop Manager	Dan Smith Keller / Frank E. Ludolph / Bill Atkinson: Apple Computer, Inc., 1981-1983
MacPaint	Bill Atkinson: Apple Computer, Inc., 1984-
Microcosm	Wendy Hall: University of Southampton, 1990 until today
Mosaic	Marc Andreessen / Eric Bina: NCSA, 1993

³¹ This footnote is dedicated to the Unknown Software Engineer. It requires a lot more bright and talented people to develop and test software than can be listed here.

MS-DOS	Microsoft Corp., 1981 until 1990s
Navigator/ Communicator	Marc Andreessen: Netscape Communications Corp., 1994 until today
NLS/Augment	Doug Engelbart / Bill English: Stanford Research Institute, Menlo Park, CA, 1960s-1977; Tymshare Corp. 1977-1984; McDonnell Douglas Corp. 1984-
NoteCards	Randall H. Trigg / Thomas P. Moran / Frank G. Halasz : Xerox PARC, 1985
PageMaker	Paul Brainard: Aldus Corp., 1986-1994, Adobe Systems Inc., 1994-2000
Photoshop	Thomas Knoll / Marc Hamburg: Adobe Systems Inc., 1989 until today
SDMS	Nicholas Negroponte / Richard A. Bolt: MIT, 1971-1977
Sketchpad	Ivan E. Sutherland, 1963
Sketchpad III	Timothy Johnson, 1963
Smalltalk	Alan Curtis Kay / Daniel H. H. Ingalls: Xerox PARC, 1972-1980
Squeak	Daniel H. H. Ingalls / Alan Curtis Kay: Apple Computer, Inc., 1996-98; Walt Disney Imagineering, 1998 until today; Learning Research Institute (LRI), 2001. http://www.squeak.org (Nov 2001) ³²
Star	David Canfield Smith / Charles Irby: Xerox PARC, 1976 until early 1980s
Storyspace	Mark Bernstein / Michael Joyce: University of North Carolina; Eastgate Systems, 1990
Windows	Microsoft Corp., 1985 until today (Windows 3.0 since 1990)
WAIS	Brewster Kahle: Thinking Machines, 1989
Word	Charles Simonyi: Microsoft Corp., early 1980s until today
WorldWideWeb/Nexus	Tim Berners-Lee / Robert Cailliau: CERN, Geneva, 1990
Xanadu	Ted Nelson: 1960 until today

³² URLs may be uniform rather than being universal and persistent. This is why I provide the date of last successful visiting the page.

Credits to Figures

2.1	Memex Desk	Nyce, James / Kahn, Paul (eds): <i>From Memex to Hypertext: Vannevar Bush and the Mind's Machine</i> . Academic Press, Boston, MA, 1991: p. 110. Reprinted from Alfred D. Crimi, LIFE Magazine 19(11), 1945	6
2.2	Proposal for the World Wide Web	Berners-Lee, Tim: <i>Information Management: A Proposal</i> . CERN, Geneva, 1989. http://www.w3.org/History/1989/proposal.html (Mar 2001): p. 1	11
2.3	Memex Desktop	Adelman, Ian / Kahn, Paul: Illustration for <i>As We May Think</i> . In: Interactions 3(2, Mar.) p. 42, 1996 – edited by the author	13
2.4	Parallel Textface™ for Xanadu	Nelson, Theodor Holm: <i>Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use</i> . In: ACM Computing Surveys 31(4es) Article No. 33, 1999. http://www.sfc.keio.ac.jp/~ted/XUsurvey/xuDation.html (Jul 2001): Fig. 3. http://www.sfc.keio.ac.jp/~ted/XUsurvey/PTF2CLO8.JPG (Jul 2001). (Reprinted from [Nelson 72])	15
2.5	Interactive Webster's Dictionary	Kay, Alan Curtis: <i>The Reactive Engine</i> . PhD., 1969: University of Utah: p. 158. http://www.mprove.de/diplom/gui/kay69.html (Nov 2001)	20
2.6	NoteCards	Halasz, Frank G.: <i>Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems</i> . In: Communications of the ACM 31(7) p. 836-852, 1988: Fig. 1, p. 837 and Fig. 2, p. 838 – montage	22
2.7	Compound Hypertext	Nelson, Theodor Holm: <i>Literary Machines</i> . 93.1. Mindful Press, Sausalito, CA, 1981. http://www.sfc.keio.ac.jp/~ted/TN/PUBS/LM/LMpage.html (Nov 2001): p. 1/15-1/16	24
2.8	Symbolics Document Examiner	provided by Ralf Möller, University of Hamburg	25
2.9	Guide's Cursors	the author, based on [Nielsen 90, Fig. 5.4, p. 92]	26
2.10	Guide	mockup by the author, based on [Nielsen 90, Fig. 5.3, p. 91]	27
2.11	Storyspace	the author	29
2.12	WorldWideWeb/Nexus	Berners-Lee, Tim / Cailliau, Robert / Luotonen, Ari / Nielsen, Henrik Frystyk / Secret, Arthur: <i>The World-Wide Web</i> . In: Communications of the ACM 37(8) p. 76-82, 1994: http://www.w3.org/History/1994/WWW/Journals/CACM/screensnap2_24c.tiff (Mar 2001) – edited by the author	33
2.13	Hyper-G Architecture	Andrews, Keith / Kappe, Frank / Maurer, Hermann: <i>The Hyper-G Network Information System</i> . In: Journal on Universal	35

- Computer Science (J.UCS) 1(4), 1995.
<ftp://ftp.unibw-muenchen.de/pub/comp/infosys/Hyper-G/papers/dms94.ps.gz> (Jun 2001): Fig. 2 – legend moved
- 2.14 Hyper-G Harmony 37
 Andrews, Keith / Kappe, Frank / Maurer, Hermann: *The Hyper-G Network Information System*. In: Journal on Universal Computer Science (J.UCS) 1(4), 1995.
<ftp://ftp.unibw-muenchen.de/pub/comp/infosys/Hyper-G/papers/dms94.ps.gz> (Jun 2001): Fig. 4
- 2.15 Dexter Hypertext Reference Model – Storage Layer 42
 Halasz, Frank G./ Schwartz, Mayer: *The Dexter Hypertext Reference Model*. In: Communications of the ACM 37(2) p. 30-39, 1994: Fig. 4, p. 35
- 3.1 Sketchpad Console 55
 Schwarz, Hans-Peter (ed.): *Medien – Kunst – Geschichte*. p. 61, ZKM, Zentrum für Kunst- und Medientechnologie Karlsruhe; Prestel, München, 1997 – section
- 3.2 NLS Workstation 57
 Douglas C. Engelbart: *Study for the Development of Human Intellect Augmentation Techniques*. Final Report under Contract NAS1-5904, SRI Project 5890 for Nasa Langley Research Center. Stanford Research Institute, Menlo Park, CA, 1968.
 User's Work-Station Console
<http://www.histech.rwth-aachen.de/www/quellen/engelbart/study68index.html> (Sep 2001),
<http://www.histech.rwth-aachen.de/www/quellen/engelbart/Workstation.jpg> (Sep 2001)
- 3.3 First Mouse 58
<http://sloan.stanford.edu/mousesite/> (Mar 2001)
- 3.4 Five-Finger Chording Keyset 58
<http://sloan.stanford.edu/mousesite/chordkeyboard.jpg> (Mar 2001)
- 3.5 Flex Machine 60
 Kay, Alan Curtis: *The Reactive Engine*. PhD., 1969: University of Utah. <http://www.mprove.de/diplom/gui/kay69.html> (Nov 2001). Reprinted in [Friedewald 99, Fig. 72, p. 253]
- 3.6 Dynabook 61
 a) Kay, Alan Curtis: *A Personal Computer for Children of All Ages*. In: Proceedings of the ACM National Conference, 1972. Reprinted in [Friedewald 99, Fig. 73, p. 255]
 b) Kay, Alan Curtis: *Personal Computing*. In: Meeting on 20 Years of Computing Science. Istituto di Elaborazione della Informazione, Pisa, Italy, 1975: p. 2
- 3.7 Xerox Alto and Smalltalk 63
 a) Johnson, Jeff / Roberts, Teresa L. / Verplank, William L. / Smith, David Canfield / Irby, Charles / Beard, Marian / Mackey, Kevin: *The Xerox Star: A Retrospective*. In: Computer 22(9) p. 11-29, 1989: Fig. 9, p. 22
 b) Kay, Alan Curtis: *The Early History of Smalltalk*. In: Bergin, Thomas J. / Gibson, Richard G. (eds): *History of Programming Languages II*. p. 511. Addison-Wesley, Reading, MA, 1996: Fig. 11.53, p. 554

3.8	Textediting in Smalltalk	64
	Tesler, Lawrence G.: <i>The Smalltalk Environment</i> . In: Byte 6(8) p. 90-147, 1981: Photo 5, p. 106	
3.9	Xerox Star	67
	Smith, David Canfield / Irby, Charles / Kimball, Ralph / Verplank, William L. / Harslem, Eric: <i>Designing the Star User Interface</i> . In: Degano, Pierpaolo / Sandewall, Erik (eds): <i>Integrated Interactive Computing Systems. ECICS '82</i> . Stresa, Italy. p. 297-313, 1982. Reprinted from Byte 7(4), 1982: p. 303	
3.10	Spacial Data Managment System – Media Room	68
	Brand, Stewart: <i>The Media Lab: Inventing the Future at M.I.T.</i> Viking-Penguin, New York, 1987: color insert, p. 9 – section	
3.11	Spacial Data Managment System – Dataland	69
	Bolt, Richard A.: <i>The Human Interface: Where People and Computers Meet</i> . Lifelong Learning Publications, Belmont, CA, 1984: Fig. 2-4 (color insert, p. i)	
3.12	Apple Lisa	73
	Bolt, Richard A.: <i>The Human Interface: Where People and Computers Meet</i> . Lifelong Learning Publications, Belmont, CA, 1984: Fig. 2-19, p. 27	
3.13	Apple Macintosh – Finder and Adobe FrameMaker	75
	the author	
3.14	Relationship between Window and Document	82
	a) Kay, Alan Curtis: <i>The Reactive Engine</i> . PhD., 1969: University of Utah: p. 129. http://www.mprove.de/diplom/gui/kay69.html (Nov 2001)	
	b) Apple Computer, Inc.: <i>Macintosh Human Interface Guidelines</i> . Addison-Wesley, Reading, MA, 1995: Fig. 5-27, p. 158. http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html (Nov 2001)	
3.15	GoLive menu and tool bar on Windows	84
	the author	
3.16	GoLive menu and tool bar on Macintosh	85
	the author	

References

- [Andrews/Kappe/
Maurer 95] Andrews, Keith / Kappe, Frank / Maurer, Hermann: *The Hyper-G Network Information System*. In: Journal on Universal Computer Science (J.UCS) 1(4), 1995.
<ftp://ftp.unibw-muenchen.de/pub/comp/infosys/Hyper-G/papers/dms94.ps.gz> (Jun 2001)
- [Apple 95] Apple Computer, Inc.: *Macintosh Human Interface Guidelines*. Addison-Wesley, Reading, MA, 1995. <http://developer.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html> (Nov 2001)
- [Baecker et al. 95] Baecker, Ronald M. / Grudin, Jonathan / Buxton, William A. S. / Greenberg, Saul (eds): *Readings in Human-Computer Interaction: Toward the Year 2000*. 2nd ed. Morgan Kaufman, San Francisco, CA, 1995
- [Bardini 2000] Bardini, Thierry: *Bootstrapping – Douglas Engelbart, Coevolution, and the Origins of Personal Computing*. Stanford University Press, Stanford, CA, 2000
- [Berners-Lee 89] Berners-Lee, Tim: *Information Management: A Proposal*. CERN, Geneva, 1989. <http://www.w3.org/History/1989/proposal.html> (Mar 2001)
- [Berners-Lee 93] Berners-Lee, Tim: *The WorldWideWeb browser*. CERN, Geneva, 1993. <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html> (Mar 2001)
- [Berners-Lee 97] Berners-Lee, Tim: *Cleaning up the User Interface*. W₃C, 1997. <http://www.w3.org/DesignIssues/UI.html> (Mar 2001)
- [Berners-Lee 99] Berners-Lee, Tim: *Weaving the Web*. HarperCollins, New York, 1999
- [Berners-Lee/
Cailliau 90] Berners-Lee, Tim / Cailliau, Robert: *WorldWideWeb: Proposal for a Hypertext Project*. CERN, Geneva, 1990.
<http://www.w3.org/Proposal.html> (Mar 2001)
- [Berners-Lee/
Cailliau et al. 94] Berners-Lee, Tim / Cailliau, Robert / Luotonen, Ari / Nielsen, Henrik Frystyk / Secret, Arthur: *The World-Wide Web*. In: Communications of the ACM 37(8) p. 76-82, 1994
- [Bolt 84] Bolt, Richard A.: *The Human Interface: Where People and Computers Meet*. Lifelong Learning Publications, Belmont, CA, 1984
- [Bolter et al. 96] Bolter, Jay David / Joyce, Michael / Smith, John B. / Bernstein, Mark: *Getting Started with Storyspace for Macintosh 1.5*. Eastgate Systems, Watertown, MA, 1996. <http://www.eastgate.com> (Mar 2001)
- [Brand 87] Brand, Stewart: *The Media Lab: Inventing the Future at M.I.T.* Viking-Penguin, New York, 1987
- [Bush 45] Bush, Vannevar: *As We May Think*. In: Interactions 3(2, Mar.) p. 35-46, 1996. Reprinted from The Atlantic Monthly 176 (July 1945)
- [Cailliau/Ashman 99] Cailliau, Robert / Ashman, Helen: *Hypertext in the Web - a History*. In: ACM Computing Surveys 31(4es) Article No. 35, 1999
- [Card/Moran 86] Card, Stuart K. / Moran, Thomas P.: *User Technology: From Pointing to Pondering*. 1986. In: [Goldberg 88, p. 493-521]

- [Card et al. 83] Card, Stuart K. / Moran, Thomas P. / Newell, Allen: *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983
- [Card et al. 91] Card, Stuart K. / Mackinlay, Jock D. / Robertson, George G.: *A Morphological Analysis of the Design Space of Input Devices*. In: ACM Transactions on Information Systems 9(2, Apr) p. 99-122, 1991
- [Ceruzzi 98] Ceruzzi, Paul E.: *A History of Modern Computing*. The MIT Press, Cambridge, MA, 1998
- [Conklin 87] Conklin, Jeff: *Hypertext: An Introduction and Survey*. In: Computer 20(9) p. 17-41, 1987
- [Cooper 99] Cooper, Alan: *The Inmates Are Running the Asylum*. Macmillan Computer Publishing, Indianapolis, 1999
- [Davis et al. 92] Davis, Hugh / Hall, Wendy / Heath, Ian / Hill, Gary / Wilkins, Rob: *Towards an Integrated Information Environment with Open Hypermedia Applications*. In: ECHT '92, Milan, Italy. p. 181-190, 1992
- [Engelbart 62] Engelbart, Douglas C.: *Augmenting Human Intellect: A Conceptual Framework*. Summary Report AFOSR-3223 under Contract AF 49(638)-1024, SRI Project 3578 for Air Force Office of Scientific Research. SRI, Menlo Park, CA, 1962. <http://www.histech.rwth-aachen.de/www/quellen/engelbart/ahi62index.html> (Sep 2001)
- [Engelbart 63] Engelbart, Douglas C.: *A Conceptual Framework for the Augmentation of Man's Intellect*. In: Howerton, Paul W. / Weeks, D.C. (ed): *Vistas in Information Handling*, Vol. 1. p. 1-29. Spartan Books, Washington D.C., 1963. Reprinted in [Greif 88]
- [Engelbart 75] Engelbart, Douglas C.: *NLS Teleconferencing Features: The Journal and Shared-Screen Telephoning*. Stanford Research Institute, Menlo Park, CA, 1975. <http://www.bootstrap.org/augment-33076.htm> (Sep 2001)
- [Engelbart 84] Engelbart, Douglas C.: *Authorship Provisions in Augment*. In: IEEE Computer Conference, 1984. Reprinted in [Greif 88]. <http://www.bootstrap.org/oad-2250.htm> (Sep 2001)
- [Engelbart 88] Engelbart, Douglas C.: *The Augmented Knowledge Workshop*. In: [Goldberg 88, p. 185-236]. <http://www.bootstrap.org/augment-101931.htm> (Sep 2001)
- [Engelbart/English 68] Engelbart, Douglas C. / English, William K.: *A Research Center for Augmenting Human Intellect*. In: FJCC '68, San Francisco. p. 395-410. Thompson Books, 1968. Reprinted in [Greif 88]. <http://www.histech.rwth-aachen.de/www/quellen/engelbart/ResearchCenter1968.html> (Sep 2001)
- [English/Engelbart/Berman 67] English, William K. / Engelbart, Douglas C. / Berman, M.: *Display Selection Techniques for Text Manipulation*. In: IEEE Transactions on Human-Factors in Electronics 8(1) p. 5-15, 1967. <http://www.histech.rwth-aachen.de/www/quellen/engelbart/Display1967.html> (Sep 2001)
- [Fenn/Maurer 94] Fenn, Barry / Maurer, Hermann: *Harmony... on an Expanding Net*. In: Interactions 1(4, Oct.) p. 26-38, 1994
- [Frenkel 94] Frenkel, Karen A.: *A Conversation with Alan Kay*. In: Interactions 1(2, Apr.) p. 13-22, 1994

- [Friedewald 99] Friedewald, Michael: *Der Computer als Werkzeug und Medium*. GNT-Verlag, Berlin, 1999
- [Gillies/Cailliau 2000] Gillies, James / Cailliau, Robert: *How the Web was Born*. Oxford University Press, Oxford, 2000
- [Gloor 97] Gloor, Peter: *Elements of Hypermedia Design*. Birkhauser, Boston, 1997. <http://www.birkhauser.com/hypermedia/> (Jul 2001)
- [Goldberg 88] Goldberg, Adele (ed): *A History of Personal Workstations*. Addison-Wesley, Reading, MA, 1988
- [Greif 88] Greif, Irene (ed): *Computer-Supported Cooperative Work: A Book of Readings*. Morgan Kaufman, San Mateo, CA, 1988
- [Haan et al. 92] Haan, Bernard J. / Kahn, Paul / Riley, Victor A. / Coombs, James H. / Meyrowitz, Norman K.: *IRIS Hypermedia Services*. In: *Communications of the ACM* 35(1) p. 36, 1992
- [Halasz 87] Halasz, Frank G.: *Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems*. In: *Hypertext '87*, Chapel Hill, NC. p. 345-365, 1987
- [Halasz 88] Halasz, Frank G.: *Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems*. In: *Communications of the ACM* 31(7) p. 836-852, 1988: revised edition of [Halasz 87]
- [Halasz/Schwartz 94] Halasz, Frank G./ Schwartz, Mayer: *The Dexter Hypertext Reference Model*. In: *Communications of the ACM* 37(2) p. 30-39, 1994
- [Hegland 2000] Hegland, Frode: *Doug Engelbart Audio Glossary*. (Audio) Liquid Information Organization, 2000.
<http://www.liquid.org/glossary/index.html> (Nov 2001)
- [Hiltzig 99] Hiltzig, Michael: *Dealers of Lightning – Xerox PARC and the Dawn of the Computer Age*. Harper Collins, New York, 1999
- [Horowitz 84] Horowitz, Ellis: *Fundamentals of Programming Languages*. 2nd ed. Springer, Berlin, 1984
- [Johnson et al. 89] Johnson, Jeff / Roberts, Teresa L. / Verplank, William L. / Smith, David Canfield / Irby, Charles / Beard, Marian / Mackey, Kevin: *The Xerox Star: A Retrospective*. In: *Computer* 22(9) p. 11-29, 1989
- [Joyce 91] Joyce, Michael: *Storyspace as a hypertext system for writers and readers of varying ability*. In: *Hypertext '91*, San Antonio, TX. p. 381-387, 1991
- [Joyce 92] Joyce, Michael: *Afternoon, a story*. 3rd ed. Eastgate Press, Cambridge, MA, 1992. <http://www.eastgate.com> (Mar 2001)
- [Kay 68] Kay, Alan Curtis: *FLEX – A flexible extendable language*. MSc., 1968: University of Utah.
<http://www.mprove.de/diplom/gui/kay68.html> (Nov 2001)
- [Kay 69] Kay, Alan Curtis: *The Reactive Engine*. PhD., 1969: University of Utah. <http://www.mprove.de/diplom/gui/kay69.html> (Nov 2001)
- [Kay 72a] Kay, Alan Curtis: *A Personal Computer for Children of All Ages*. In: *Proceedings of the ACM National Conference*, 1972
- [Kay 72b] Kay, Alan Curtis: *A Dynamic Medium for Creative Thought*. In: *Proceedings of the 1972 Minnesota NCTE Seminars on Research in English Education*. 1972
- [Kay 75] Kay, Alan Curtis: *Personal Computing*. In: *Meeting on 20 Years of Computing Science*. Instituto di Elaborazione della Informazione, Pisa, Italy, 1975

- [Kay 87] Kay, Alan Curtis: *Doing With Images Makes Symbols: Communicating with Computers*. Apple Computer, Inc., Cupertino, CA, 1987. (Video, 97')
- [Kay 90] Kay, Alan Curtis: *User Interface: A Personal View*. In: Laurel, Brenda (ed): *The Art of Human-Computer Interface Design*. p. 191-207. Addison-Wesley, Reading, MA, 1990
- [Kay 96] Kay, Alan Curtis: *The Early History of Smalltalk*. In: Bergin, Thomas J. / Gibson, Richard G. (eds): *History of Programming Languages II*. p. 511. Addison-Wesley, Reading, MA, 1996
- [Kay/Goldberg 77] Kay, Alan Curtis / Goldberg, Adele: *Personal Dynamic Media*. In: *Computer* 10(3) p. 31-41, 1977. Reprinted in [Goldberg 88, p. 254-263]
- [Kay/
Müller-Prove 2001] Kay, Alan Curtis / Müller-Prove, Matthias: *personal communication at 4/5/2001*. <http://www.mprove.de/diplom/mail/kay.html> (Nov 2001)
- [Klaphaak 96] Klaphaak (Jr.), David: *Events in the Life of Vannevar Bush*. Brown University, 1996. <http://www.cs.brown.edu/research/graphics/html/info/timeline.html> (Nov 2001)
- [Lampson 72] Lampson, Butler: *Why Alto*. XEROX Inter-Office Memorandum 1972. Reprinted in [Friedewald 99, p. 422-425]
- [Licklider 60] Licklider, Joseph C. R.: *Man-Computer Symbiosis*. In: *IRE Transactions on Human Factors in Electronics* (March) p. 4-11, 1960. Reprinted in [Goldberg 88, p. 131-140]. <http://www.histech.rwth-aachen.de/www/quellen/SRC61-Licklider.pdf> (Aug 2001)
- [Linzmayr 99] Linzmayer, Owen W.: *Apple Confidential: The Real Story of Apple Computer, Inc.* No Starch Press, San Francisco, CA, 1999
- [Lowe/Hall 99] Lowe, David / Hall, Wendy: *Hypermedia & the Web: An Engineering Approach*. John Wiley & Sons, Chichester, 1999
- [Ludolph/Perkins 98] Ludolph, Frank / Perkins, Roderick: *The Lisa User Interface*. In: *CHI '98*, Los Angeles, CA, (Summary). p. 18-19, 1998
- [Maurer et al. 98] Maurer, Hermann / Scherbakov, Nick / Halim, Zahran / Razak, Zaidah: *From Databases to Hypermedia*. Springer, Berlin, 1998
- [McLuhan 64] McLuhan, Herbert Marshall: *Understanding Media: The Extensions of Man*. MIT Press ed. The MIT Press, Cambridge, MA, 1997
- [Müller-Prove 99] Müller-Prove, Matthias: *Adobe GoLive's Point & Shoot: an interface technique for creating hyperlinks*, 1999. <http://www.mprove.de/script/99/pointshoot/> (Nov 2001)
- [Myers 98] Myers, Brad A.: *A brief history of human-computer interaction technology*. In: *Interactions* 5(2, Mar.) p. 44-54, 1998
- [Naiman 84] Naiman, Arthur: *Introduction to the Lisa*. Addison-Wesley, Reading, MA, 1984
- [Nelson 72] Nelson, Theodor Holm: *As We Will Think*. In: Nyce, James / Kahn, Paul (eds): *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*. Boston, MA p. 245. Academic Press, 1991
- [Nelson 74] Nelson, Theodor Holm: *Computer Lib / Dream Machines*. Chicago, 1974
- [Nelson 93] Nelson, Theodor Holm: *Literary Machines*. 93.1. self published 1993. <http://www.sfc.keio.ac.jp/~ted/TN/PUBS/LM/LMpage.html> (Nov 2001): revised edition of Mindful Press, Sausalito, CA, 1981

- [Nelson 97a] Nelson, Theodor Holm: *The Future of Information*. ASCII Corporation, Tokyo, 1997. <http://www.sfc.keio.ac.jp/~ted/INFUTscans/INFUTscans.html> (Nov 2001)
- [Nelson 97b] Nelson, Theodor Holm: *Embedded Markup Considered Harmful*. In: XML: Principles, Tools, and Techniques (World Wide Web Journal) 2(4, Fall), 1997. <http://www.xml.com/pub/a/w3j/s3.nelson.html> (Sep 2001)
- [Nelson 98a] Nelson, Theodor Holm: *Parallel Visualization: Transpointing Windows*. Ted Nelson's home page, 1998. <http://www.sfc.keio.ac.jp/~ted/TN/PARALUNE/paraviz.html> (Jul 2001)
- [Nelson 98b] Nelson, Theodor Holm: *Examples of Parallel Documents*. Ted Nelson's home page, 1998. <http://www.sfc.keio.ac.jp/~ted/TN/PARALUNE/parexamples.html> (Jul 2001)
- [Nelson 99a] Nelson, Theodor Holm: *Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use*. In: ACM Computing Surveys 31(4es) Article No. 33, 1999. <http://www.sfc.keio.ac.jp/~ted/XUsurvey/xuDation.html> (Jul 2001)
- [Nelson 99b] Nelson, Theodor Holm: *The Unfinished Revolution and Xanadu*. In: ACM Computing Surveys 31(4es) Article No. 37, 1999
- [Newman/Mott 82] Newman, William M. / Mott, Timothy: *Officetalk-Zero: An Experimental Integrated Office System*. In: Degano, Pierpaolo / Sandewall, Erik (eds): *Integrated Interactive Computing Systems*. ECICS '82. Stresa, Italy. p. 315-331, 1982
- [Nielsen 90] Nielsen, Jakob: *Hypertext and Hypermedia*. Academic Press, Boston, MA, 1990
- [Nielsen 95] Nielsen, Jakob: *Multimedia and Hypertext - The Internet and Beyond*. Academic Press, Boston, MA, 1995
- [Nürnberg/Leggett/Schneider 97] Nürnberg, Peter J. / Leggett, John J. / Schneider, Erich R.: *As We Should Have Thought*. In: Hypertext '97, Southampton, UK, 1997
- [Nyce/Kahn 91] Nyce, James / Kahn, Paul (eds): *From Memex to Hypertext: Vannevar Bush and the Mind's Machine*. Academic Press, Boston, MA, 1991
- [Papert 80] Papert, Seymour: *Mindstorms: Children, Computers, and Powerful Ideas*. The Harvester Press Limited, Brighton, Sussex, 1980
- [Pearl 89] Pearl, Amy: *Sun's Link Service: A Protocol for Open Linking*. In: Hypertext '89, Pittsburgh, PA. p. 186-193, 1989
- [Perkins et al. 97] Perkins, Roderick / Smith Keller, Dan / Ludolph, Frank: *Inventing the Lisa user interface*. In: Interactions 4(1) p. 40-53, 1997
- [Raskin 94a] Raskin, Jef: *Holes in History*. In: Interactions 1(3, Jul.), 1994
- [Raskin 94b] Raskin, Jef: *Intuitive equals Familiar*. In: Communications of the ACM 37(9) p. 17, 1994. <http://www.asktog.com/papers/raskinintuit.html> (Nov 2001)
- [Raskin 2000] Raskin, Jef: *The Humane Interface – New Directions for Designing Interactive Systems*. ACM Press, New York, 2000
- [Schreiber 90] Schreiber, Jörg: *Macintosh Atlas*. 2nd ed. Verein Mensch am Computer (MAC) e.V., Duisburg, 1990
- [Segaller 98] Segaller, Stephen: *Nerds 2.0.1 – A Brief History of the Internet*. TV Books, New York, 1998

- [Shneiderman 98] Shneiderman, Ben: *Designing the User Interface*. 3rd ed. Addison-Wesley, Reading, MA, 1998
- [Shneiderman/
Kearsley 89] Shneiderman, Ben / Kearsley, Greg: *Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information*. Addison-Wesley, Reading, MA, 1989
- [Smith 93] Smith, David Canfield: *Pygmalion: An Executable Electronic Blackboard*. In: Cypher, Allen (ed): *Watch What I Do: Programming by Demonstration*. p. 19-48. The MIT Press, Cambridge, MA, 1993. <http://www.acypher.com/wwid/Chapters/01Pygmalion.html> (Oct 2001)
- [Smith et al. 82] Smith, David Canfield / Irby, Charles / Kimball, Ralph / Verplank, William L. / Harslem, Eric: *Designing the Star User Interface*. In: Degano, Pierpaolo / Sandewall, Erik (eds): *Integrated Interactive Computing Systems. ECICS '82*. Stresa, Italy. p. 297-313, 1982. Reprinted from *Byte* 7(4), 1982
- [Sutherland 63a] Sutherland, Ivan E.: *Sketchpad – A Man-Machine Graphical Communication System*. MIT, Lincoln Lab., 1963: Report #296, Reissued 1965
- [Sutherland 63b] Sutherland, Ivan E.: *Sketchpad – A Man-Machine Graphical Communication System*. In: *SJCC '63*. p. 329-346, 1963
- [Taft 79] Taft, Edward A.: *Alto User's Handbook*. September 1979. Xerox Parc, Palo Alto, CA, 1979. <http://www.spies.com/~aek/alto/> (Oct 2001)
- [Tesler 81] Tesler, Lawrence G.: *The Smalltalk Environment*. In: *Byte* 6(8) p. 90-147, 1981
- [Tognazzini 92] Tognazzini, Bruce: *Tog on Interface*. Addison-Wesley, Reading, MA, 1992
- [Tognazzini 96] Tognazzini, Bruce: *Tog on Software Design*. Addison-Wesley, Reading, MA, 1996
- [Tognazzini 98] Tognazzini, Bruce: *Scaling Information Access*. In: *AskTog* (8), 1998. <http://www.asktog.com/columns/008scaledinfo.html> (Nov 2001)
- [Tognazzini 99] Tognazzini, Bruce: *Reader Mail: On pull-down menus*. In: *AskTog* (5), 1999. <http://www.asktog.com/readerMail/1999-05ReaderMail.html#Anchor-On-35882> (Oct 2001)
- [van Dam 87] van Dam, Andries: *Hypertext '87 Keynote Address*. In: *Communications of the ACM* 31(7) p. 887-895, 1988
- [van Dam 97] van Dam, Andries: *Post-WIMP User Interfaces*. In: *Communications of the ACM* 40(2), 1997
- [van Dam 2001] van Dam, Andries: *Post-WIMP User Interfaces: the Human Connection*. In: Earnshaw, Rae / Guedj, Richard / van Dam, Andries / Vince, John (eds): *Frontiers of Human-Centered Computing, Online Communities and Virtual Environments*. Springer, Berlin, 2001
- [Wadlow 81] Wadlow, Thomas A.: *The Xerox Alto Computer*. In: *Byte* 6(9) p. 58-68, 1981
- [Wagner 2000] Wagner, Kirsten: *Informations- und Wissensorganisation anhand räumlicher Ordnungsmodelle: Das Spatial Data-Management System der Architecture Machine Group als Fallbeispiel*. In: *Wolkenkuckucksheim: Internationale Zeitschrift für Theorie und Wissenschaft der Architektur* 5(1, Feb.), 2000. <http://www.theo.tu-cottbus.de/Wolke/X-positionen/Wagner/wagner.html> (Aug 2001)

- [Walker 87] Walker, Janet H.: *Document Examiner: Delivery Interface for Hypertext Documents*. In: Hypertext '87, Chapel Hill, NC. p. 307-323, 1987
- [Weinreich et al. 2001] Weinreich, Harald / Obendorf, Hartmut / Lamersdorf, Winfried: *The Look of the Link – Concepts for the User Interface of Extended Hyperlinks*. In: Hypertext '01, Aarhus, Denmark, 2001. <http://www.obendorf.de/studium/projekte/lookoflink-ht01.pdf> (Feb 2001)
- [Yankelovich et al. 85] Yankelovich, Nicole / Meyrowitz, Norman K. / van Dam, Andries: *Reading and Writing the Electronic Book*. In: Computer 18(10), 1985
- [Yankelovich et al. 88] Yankelovich, Nicole / Haan, Bernard J. / Meyrowitz, Norman K. / Drucker, Steven M.: *Intermedia: The Concept and the Construction of a Seamless Information Environment*. In: Computer 21(1) p. 81-96, 1988
- [Zakon 2001] Zakon, Robert H.: *Hobbes' Internet Timeline v5.4*, 2001. <http://www.zakon.org/robert/internet/timeline/> (Sep 2001)

