Scheme For Identifying and Describing Behavioral Innovations Embodied in Computer Programs

by

Sean Sang-Chul Pak

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Eng.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEP 17 1996

LIBRARIES

© Sean S. Pak, 1996. All Rights Reserved.

The author hereby grants to M.I.T. permission to reproduce and to distribute copies of this thesis document in whole or in part, and to grant others the right to do so.

Author .....
Department of Electrical Engineering and Computer Science
May 17, 1996

Certified by .....
Randall Davis
Supervisor

Accepted .....
F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Scheme For Identifying and Describing Behavioral Innovations Embodied in Computer Programs

by

Sean Sang-Chul Pak

Submitted to the
Department of Electrical Engineering and Computer Science

May 17, 1996

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science and Master of Engineering in Electrical Engineering and Computer Science

Abstract

To clarify some of the key issues underlying the current debate on anti-cloning protection of software, a systematic scheme for identifying behavioral innovations embodied in computer programs was developed. The proposed scheme consists of three stages: (1) behavioral analysis, (2) state of the art analysis, and (3) commercial significance analysis. The feasibility and usefulness of the proposed scheme were evaluated through an in-depth case study of two World Wide Web browsers: NCSA Mosaic and Netscape Navigator. Based on the Mosaic-Navigator case study, the following conclusions were advanced: (1) The proposed scheme provides a consistent and technically coherent framework for identifying innovations in program behavior without referring to internal design elements; (2) the proposed scheme represents an excellent starting point for developing a systematic method of distinguishing market-destructive clones from legitimate competitors and follow-on innovations; and (3) the proposed scheme contains a number of useful insights and mechanisms for describing computer programs.

Acknowledgments

It all started with a brief meeting in Professor Davis's office. Having learned of his interest in law and technology, I was hoping to discuss possible topics for my thesis. After listening to my ideas, Professor Davis smiled and handed me a small, blue book titled "A Manifesto Concerning the Legal Protection of Computer Programs." I did not know at the time, but that little book would serve as my window into the fascinating and byzantine world of intellectual property law.

After a long year and a half, my journey has finally come to an end. Through it all, I could always count on the following professors for much-needed support and guidance: Professor Randall Davis, my thesis advisor; Professor Jesus del Alamo, the VI-A advisor for Intel Corporation; and Professor James Chung, my academic advisor.

In addition, I must thank everyone at the Intel Legal department, especially Mohammad Aboobaker, Ted Jenkins, Joyce Dagley, Jim McKevitt, Charles Wymore, Michael Moradzadeh, Isabella Fu, Tim Roper, and Chuck Quarton. Thank you all for making my internship an educational and enjoyable experience. I hope that our paths will cross again some day.

Very special thanks also go to George Dallas of Intel Corporate College Recruiting for his unwavering dedication and enthusiasm. Through thick and thin, George was there for the VI-A interns, making sure that we had the best possible internships.

I am also deeply indebted to Professor Margaret Burnham of the MIT Political Science Department. As my pre-law advisor, Professor Burnham introduced me to the study of law. Without her counsel and support, this thesis would not have been possible in the first place.

Finally, I thank my parents, my sister, and my grandmother. Thank you all for believing in me. I could not have done it without your love and patience.

Sean Pak

Table of Contents

1 Introduction 11
1.1 Background 11
1.2 Problem Description 21
1.3 Purpose of the Study 22
1.4 Significance of the Study 23
1.5 Brief Overview of the Methodology 24
1.6 Assumptions and Limitations 25
2 Phase 1: Proposed and Existing Schemes 26
2.1 Defining Behavioral Innovation 26
2.2 Proposed Scheme for Identifying Innovations in Program Behavior 33
2.3 Existing Schemes for Describing Innovations in Program Behavior 44
2.4 Proposed Scheme for Describing Innovations in Program Behavior 55
3 Phase 2: Mosaic-Navigator Case Study 60
3.1 Background 60
3.2 NCSA Mosaic Web Browser 62
3.3 Netscape Navigator Web Browser 105
4 Conclusion 160
4.1 Discussion of the Results 160
4.2 Suggestions for Further Research 166
Appendix A Patent Search Results 169
A.1 Patents Related to Hypertext or Hypermedia Technology 169
A.2 Patents Related to Cryptography 170
References 177

List of Figures

Figure 2.1: Diagram of the Proposed Scheme 34
Figure 2.2: Simple Transform Bubble 38
Figure 2.3: Transform Bubble with Conjunctive Inputs and Disjunctive Outputs 38
Figure 3.1: Graph M: Overview of Mosaic's Functionality 65
Figure 3.2: Graph M.1: Access the Internet 68
Figure 3.3: Graph M.2: Process Document 72
Figure 3.4: Graph M.3: Execute Hypermedia Jump 76
Figure 3.5: Snapshot of Mosaic's Main Window 81
Figure 3.6: Snapshot of Mosaic's Interface When Accessing the Internet 83
Figure 3.7: Snapshot of Mosaic's Interface When Processing a Retrieved Document 85
Figure 3.8: Snapshot of Mosaic's Interface When Executing a Hypermedia Jump 87
Figure 3.9: Notations Used in the Graphs of Novelty 99
Figure 3.10: Graph M (Modified): Overview of Mosaic's Functionality 100
Figure 3.11: Graph M.1 (Modified): Access the Internet 101
Figure 3.12: Graph M.2 (Modified): Process Document 102
Figure 3.13: Graph M.3: Execute Hypermedia Jump 103
Figure 3.14: Graph N: Overview of Navigator's Functionality 108
Figure 3.15: Graph N.1: Access the Internet 111
Figure 3.16: Graph N.1.1: Establish Secure Connection 116
Figure 3.17: Graph N.2: Process Document 122
Figure 3.18: Graph N.3: Execute Hypermedia Jump 126
Figure 3.19: Graph N.4: Process On-line Form 129
Figure 3.20: Snapshot of Navigator's Main Window 134
Figure 3.21: Snapshot of Navigator's Interface When Accessing the Internet 135
Figure 3.22: Snapshot of Navigator's Interface When Establishing a Secure Connection 137
Figure 3.23: Snapshot of Navigator's Interface When Processing a Retrieved Document 138
Figure 3.24: Snapshot of Navigator's Interface When Executing a Hypermedia Jump 139
Figure 3.25: Snapshot of Navigator's Interface When Processing an On-line Form 141
Figure 3.26: Notations Used in the Graphs of Novelty 150
Figure 3.27: Graph N (Modified): Overview of Navigator's Functionality 151
Figure 3.28: Graph N.1 (Modified): Access the Internet 152
Figure 3.29: Graph N.1.1 (Modified): Establish Secure Connection 153
Figure 3.30: Graph N.2 (Modified): Process Document 154
Figure 3.31: Graph N.3 (Modified): Execute Hypermedia Jump 155
Figure 3.32: Graph N.4 (Modified): Process On-line Form 156

Chapter 1

Introduction

1.1 Background

1.1.1 State of the Software Industry

In just over a decade, the software industry has evolved into a multi-billion dollar industry. This phenomenal growth can be largely attributed to the high level of innovation within the industry: According to a recent survey, new software products are appearing on the market at a rate of more than two per day (Massachusetts Computer Software Council, Inc. 89). In addition, these products are representing an increasingly diverse range of applications--from word processing to home education to personal finance.

On the whole, the software industry has been able to achieve such a high level of innovation by encouraging strong competition within its applications markets (Samuelson et al. 2374). Except for major markets dominated by a few firms, most of the applications markets, with their low barriers to entry, have attracted a large number of companies, each striving to capture a greater market share by offering more features, better performance, or superior user interface than their competitors (Samuelson et al. 2377). With increasing competition, some markets have also experienced price wars (Samuelson et al. 2374). The overall result is that software products today are more useful and, in general, less expensive than their past counterparts. Thus, from the standpoint of consumer welfare, it can be argued that free competition, tempered by existing legal regimes, has been largely beneficial for the software industry.

Recent developments in the industry, however, suggest that this argument may no longer be valid. With highly visible companies like Lotus and Apple leading the way,

more and more software developers are asserting their intellectual property rights in an attempt to protect their investments in developing innovative products. Because software innovations can often be copied rapidly and inexpensively by a second comer, these developers are claiming that they can no longer recoup the high costs of research and development without stronger legal protection (Samuelson et al. 2339). Some of these developers are seeking to extend existing copyright protection, while others are attempting to obtain unprecedented patent protection for their innovations (Samuelson et al. 2311). Both of these approaches are highly controversial, resulting in turmoil within the software industry as well as within the legal community.

1.1.2 The Manifesto

In order to ground the software legal protection debate on a more solid and constructive footing, two technologists, Randall Davis and Mitchell Kapor, and two lawyers, Pamela Samuelson and J. H. Reichman, have recently published "A Manifesto Concerning the Legal Protection of Computer Programs." Based on four years of collaboration, the authors of the Manifesto present a normative analysis of the key issues underlying the current debate on legal protection of software. Motivated primarily by societal rather than individual or corporate interests, they seek to explore the nature of computer programs and that of the software industry in order to identify gaps in the existing protection for software products and to suggest the proper legal mechanisms for addressing those gaps.

In the first part of their analysis, Samuelson et al. advance the following set of observations on the nature of computer programs (2315-2316): First, the principal source of value in a computer program lies in its behavior, not in its text, where program behavior comprises functionality, performance, and user interface. In other words, consumers buy and use computer programs for the useful behavior produced during execution and not for

the literary value contained in the program text. Second, program behavior and text are, to a large extent, independent in the sense that a second comer can develop a functionally identical clone without accessing the original program's text. Third, computer programs are "machines whose medium of construction happens to be text" (Davis 111). According to this view, computer programs, like other machines, embody a considerable amount of technical know-how, much of which can be accessed through regular use or program decompilation. Finally, innovations embodied in computer programs are typically incremental and cumulative in nature. By the term "innovations," Samuelson et al. are referring to "products of skilled effort that advance the state of the art but do not meet the patent standard of nonobviousness" (2330). Generally speaking, innovations come in two forms: (1) combining existing elements in new ways to accomplish a task more efficiently; or (2) combining new elements with the old in order to accomplish a task in a different way (Samuelson et al. 2332). In their analysis, Samuelson et al. emphasize behavioral innovations--that is, advances in program functionality, performance, or user interface--as opposed to structural or design innovations.

Taking these special properties of computer programs into consideration, Samuelson et al. argue that innovations in the useful behavior of software products are vulnerable to rapid and inexpensive copying by a second comer, known in the industry as "cloning" (2337). This vulnerability is problematic for three reasons: First, because software products are easy to mass-produce, a second comer can copy the innovations embodied in a software product and quickly introduce a competing product into the market, thereby shortening the natural lead time for the innovator (Samuelson et al. 2338). Second, the costs involved in developing software innovations, as a rule, are much higher than the costs of copying (Samuelson et al. 2338). Consequently, a second comer can not only develop a clone in a relatively short amount of time, but can undercut the innovator's price as

well. Finally, the situation is especially problematic for the software industry because the primary source of value for a software product is its useful behavior and not its program text (Samuelson et al. 2318). For the average user, it may be difficult to distinguish an innovative software product from a clone offering a substantially identical combination of functionality, performance, and user interface--even if the program texts of the two products exhibit different structure, sequence, and organization (Samuelson et al. 2319). In fact, the program texts of software products are generally invisible to the user. Although the situation can be mitigated by factors such as exposure to brandnames (Samuelson et al. 2319), clones represent a significant threat to the developers of innovative software.

Unfortunately, according to Samuelson et al., the problems raised by the behavioral cloning of innovative software have not been adequately addressed by existing legal regimes--namely, trade secrets, copyrights, and patents (2342). In most cases, trade secrets cannot protect behavioral innovations, which can easily be accessed through regular use or program decompilation (Samuelson et al. 2343). Copyrights and patents are more promising, but they also give rise to conceptual and procedural difficulties when applied to innovations in program behavior (Samuelson et al. 2346-2347).

In theory, copyrights should not be used to protect utilitarian designs or processes; instead, they should be limited to the textual elements of a computer program. Consequently, copyrights cannot protect against a clone that imitates the program behavior of an innovative product without copying any of the source code (Samuelson et al. 2360-2361).

Patents, on the other hand, are intended to protect utilitarian works; accordingly, patents appear to be the logical choice for protecting innovative software against unauthorized cloning. In reality, however, patents should serve a limited role because most software innovations do not satisfy the inventive standard of non-obviousness (Nimmer 2-

5): Patents should only be issued for the few innovations that are not obvious to those skilled in the art.

Due to the mismatch between existing regimes and behavioral innovations, a large share of the software products currently on the market are vulnerable to rapid and inexpensive cloning by a second comer (Samuelson et al. 2364). Left unchecked, competition from clones can shorten the natural lead time of innovative products, making it difficult to justify the substantial investments needed for research and development (Samuelson et al. 2339). By and large, if innovation as a corporate policy becomes less profitable than imitation, consumers will suffer as fewer and fewer products embodying innovative ideas will be introduced into the marketplace.

1.1.3 Alternative Courses of Action

Having articulated the dangers of unchecked cloning, Samuelson et al. proceed by asking the following question: What can be done, at a societal level, to prevent market-destructive cloning of behavioral innovations and thereby preserve the high level of innovation within the software industry? They respond by presenting three policy options (2420-2421): (1) Do nothing; (2) make minimal changes within existing regimes to address the current lack of protection against behavioral cloning; or (3) adopt a more forward-looking approach and establish a new market-oriented regime, tailored to protecting behavioral innovations in software.

Based on their analysis, Samuelson et al. endorse option (3). Unlike the other two options, option (3) looks beyond the current lack of anti-cloning protection and seeks to provide a flexible legal framework that can accommodate both the present and future needs of the software industry (Samuelson et al. 2426).

Following the publication of the Manifesto, the Patent and Trademark Office (PTO) has initiated a number of reforms in order to address its deficiencies in handling software-related applications. The most visible of these reforms is a set of proposed guidelines for examining computer-implemented inventions (Hickman 3). These guidelines were developed in response to recent court rulings that have unequivocally established the patentability of computer-implemented inventions (Hickman 3).

Some critics within the industry predict that these new guidelines in conjunction with further reforms within the PTO will lead to increased patent protection for software innovations, thereby remedying the current state of underprotection (Hickman 3). Others, however, maintain that existing legal regimes, including patents, are fundamentally ill-suited for protecting behavioral innovations in software (Samuelson et al. 2364). Instead of relying on reforms within the PTO, they contend that a new market-oriented regime should be adopted in order to avoid market-destructive consequences (Samuelson et al. 2365).

Assuming that the PTO will implement the proposed guidelines, policymakers have at their disposal two out of the three options presented in the Manifesto: (1) Continue reforms within the PTO, and work within existing regimes to provide adequate protection against behavioral cloning; or (2) implement a new market-oriented regime, tailored to protecting behavioral innovations in software.

1.1.3.1 Option 1: Continue Reforms in the Patent Regime

Initially, both the PTO and the courts were very hostile to the concept of software patents. Throughout the 1960s and the 1970s, the PTO rejected most patent applications drawn to software-related inventions based on subject matter grounds (Hickman 3). The first major change came in 1981, when the Supreme Court recognized the general patentability of software-related inventions in its landmark ruling of *Diamond v. Diehr* (Nimmer

2-21, 2-22). Despite this ruling, the PTO continued to reject most of the incoming applications for software-related inventions and refused to hire examiners with specific training in software (Hickman 3). For the next decade, the PTO and the courts maintained divergent and often conflicting views on software patents (Hickman 4). The matter was finally resolved in 1994, when the Federal Circuit of Appeals, upon reviewing *In re Allapat,* rendered an en banc decision that was very favorable to software patents (Hickman 4). At this point, the PTO had no choice but to comply: Shortly following the *Allapat* ruling, the PTO issued a set of proposed guidelines, which instruct patent examiners to treat software-related inventions on the same ground as other types of inventions (Hickman 4).

The proposed guidelines establish the following procedure for reviewing applications drawn to computer-implemented inventions (Hickman Appendix II: 1-5):

1) Determine what the applicant is claiming by reviewing the written description and the claims.
   a) A computer whose actions are directed by a computer program is a statutory "machine."
   b) A computer-readable memory that can be used to direct a computer is a statutory "article of manufacture."
   c) A series of specific steps to be performed on a computer is a statutory "process."
   d) The following subject matter, independent of any physical element, is non-statutory: compilation of data, machine-readable storage encoded with data representing creative or artistic expression, a data structure, or a process involving manipulation of abstract ideas or concepts.

2) Analyze each claim to determine if it complies with 35 U.S.C. Section 112, second paragraph, and with 35 U.S.C. Section 112, first paragraph.
   a) If the claims fail to accurately define the invention, they should be rejected under Section 112, second paragraph.
   b) If the claims fail to reflect features of the invention necessary to impart the specific utility, they should be rejected under Section 112, first paragraph.
   c) If the claims are defined using "means plus function" language but the elements claimed are not adequately disclosed in the specification, they should be rejected under Section 112, second paragraph.
   d) If the claims are defined using computer program code, they should be rejected under Section 112, second paragraph.

3) Determine if the claimed invention is novel and nonobvious under 35 U.S.C. Section 102 and 35 U.S.C. Section 103.

If implemented, the above guidelines would remove most of the statutory objections against software-related patents. Consequently, patent applications drawn to software-related inventions would be examined as if they were drawn to any other type of invention (Hickman 3). Upon filing, these applications would be examined according to the patent standards of novelty, usefulness, and nonobviousness. As with other types of applications, the toughest hurdle for most software-related applications would then be the standard of nonobviousness: The applicant must demonstrate that the claimed invention is nonobvious to those skilled in the art in order to receive patent protection.

Many in the software industry believe that the proposed guidelines represent a promising first step in remedying the current lack of anti-cloning protection. They contend that these guidelines, in conjunction with further reforms within the PTO, would lead to the emergence of patents as the major form of protection for software innovations. The twenty years of protection associated with patents would effectively deter most second-comers from cloning innovative products in market-destructive ways. Although some innovative products would remain unprotected, those receiving patent protection would provide sufficient incentives to maintain the high level of innovation within the software industry. In summary, proponents of option (1) believe that existing legal regimes are capable of accommodating all types of technology including software. What the industry needs is continued reforms not a new regime.

1.1.3.2 Option 2: Implement a Market-Oriented Regime

Although continued reforms within the PTO may solve the immediate problem of behavioral cloning, some of the critics, including Samuelson et al., contend that such re-

forms will inevitably lead to undesirable consequences (2346). According to these critics, the adoption of the proposed guidelines and the increasing influx of patent applications would thrust the software industry into a state of overprotection (Samuelson et al. 2361). In fact, the PTO has already issued a number of patents for software-related "inventions" that many believe do not satisfy the inventive standards of novelty and nonobviousness (Aharonian). On the other hand, if patent protection is withheld from most software products for failing to satisfy the inventive standards, the software industry is likely to remain in a state of underprotection (Samuelson 2364).

Generally speaking, these cycles of under- and overprotection of software innovations are highly undesirable from the standpoint of consumer welfare and are likely to worsen as the software industry matures (Samuelson et al. 2365). Instead of attempting to force-fit existing regimes onto software, Samuelson et al. call for a new legal regime, tailored to protecting software innovations (2365). Taking into account the special properties of computer programs and the unique characteristics of the software industry, the proposed regime departs from the paradigm of exclusive property rights used in the patent and copyright regimes (Samuelson et al. 2370). Instead, it adopts a market-oriented approach, rooted in "the principles of market economics and market preservation" (Samuelson et al. 2365).

In proposing a market-oriented regime, Samuelson et al. are seeking to prevent market failures arising from the under- or overprotection of software innovations, where market failure implies a less than optimal level of innovation (2365). As described in the preceding sections, underprotection reduces the level of innovation within a market, as unfair appropriations by second comers prevent innovators from recouping investment costs (Samuelson et al. 2339). Similarly, overprotection reduces the level of innovation, as un-

fair levels of protection for original innovators prevent competing developers from improving upon existing innovations (Samuelson et al. 2357).

In order to end these damaging cycles of under- and overprotection while adhering to the principles of market economics and market preservation, Samuelson et al. recommend the adoption of a sui generis framework for protecting software innovations (2312). The framework would consist of two legal mechanisms (Samuelson et al. 2417-2418): (1) a short anti-cloning blocking period for innovative software products, available to their developers from the date of first commercial distribution; and (2) an opportunity to register innovative program compilations. Although this particular framework is not without flaws, they consider it as the most promising for implementing a market-oriented form of legal protection for software innovations (2413).

Under this framework, identical or near-identical clones of an innovative software product would be barred from entering the market during an initial blocking period. This period would be long enough to give developers an opportunity to develop a niche in the market, but not so long as to stifle the incremental advance of technology or to prevent the establishment of new de facto standards (Samuelson et al. 2413).

Following this initial blocking period, the proposed framework would allow developers to register innovative compilations of their products. According to Samuelson et al., the registration system may be implemented in one of two ways: (1) as a way of extending the duration of the blocking period; or (2) as a way of receiving compensation through an automatic royalty-bearing license, available on standard terms after the expiration of the initial blocking period (2417). The automatic licensing scheme, if adopted, would eliminate the traditionally high transaction costs of licensing and "provide incentives to agree rather than litigate" (Samuelson 2411). In either case, the registration system would provide additional incentives for software developers to invest in the development of innova-

tive products, without undermining the long term public interest in free competition. In addition, it would encourage innovators to disclose the internal design elements of their innovative products, thereby reducing duplication of effort and further advancing consumer welfare (Samuelson et al. 2426).

It is important to note that the proposed regime would not replace but supplement protection afforded by existing legal regimes (Samuelson et al. 2406). Copyrights would still be available for protecting the textual elements of computer programs, such as source and object codes, from wholesale copying; and patents would continue to protect software innovations, behavioral and design, that qualify as inventions according to patent standards.

In summary, proponents of option (2) argue that existing legal regimes cannot protect behavioral innovations in software without incurring undesirable consequences. What the industry needs is a paradigm shift, a new way of looking at the problem. The market-oriented regime proposed by Samuelson et al. offers just that. By departing from the traditional paradigm of exclusive property rights, the proposed regime seeks to remedy the current lack of anti-cloning protection while avoiding the damaging cycles of under- and overprotection.

1.2 Problem Description

As illustrated in the preceding sections, the lack of anti-cloning protection is an important and controversial topic within the larger debate on the legal protection of software. To date, two policy options have emerged from this debate: (1) Maintain the status quo and continue reforms in the patent regime; or (2) implement a new market-oriented regime, tailored to protecting innovations in program behavior.

Citing recent reforms within the Patent and Trademark Office (PTO), proponents of option (1) argue that the patent regime, augmented by copyrights and trade secrets, can

provide adequate protection against behavioral cloning: What the software industry needs is not a new regime, but continued reforms within existing regimes. On the other hand, proponents of option (2) contend that the status quo will inevitably lead to market-destructive consequences due to the fundamental mismatch between existing regimes and behavioral innovations in software--a mismatch that cannot be addressed through procedural reforms within the PTO. In light of this mismatch, they argue that a new market-oriented regime is necessary for achieving and sustaining an optimal level of innovation within the software industry.

The problem being investigated in this study is the following: It is difficult to determine whether the patent regime or a new market-oriented regime should be used to protect behavioral innovations in software, without an accurate understanding of what is to be protected. Merely labelling a program as "innovative" says nothing about the type of legal protection it may or may not require. Unfortunately, the analysis to date has not focused on the task of identifying innovative elements of program behavior in a structured and technically coherent manner.

1.3 Purpose of the Study

The purpose of this study was three-fold: (1) to develop and evaluate a systematic scheme for identifying behavioral innovations embodied in computer programs; (2) to analyze how behavioral innovations would be described under existing regimes; and (3) to suggest a workable scheme for describing behavioral innovations under a market-oriented regime.

In meeting the first objective, I considered the following design principles:

1) A model scheme should focus on the behavioral elements of a computer program as opposed to its internal design elements.

2) A model scheme should provide a consistent framework that can accommodate different types of computer programs.

3) A model scheme should support a technically coherent view of innovations in program behavior.

The above principles represent a model scheme and thus may not be fully realizable in any one implementation. Accordingly, I sought to strike a balance among competing or conflicting principles and to develop a workable scheme that would satisfy as many principles as possible.

1.4 Significance of the Study

In the course of this study, my primary objective was to develop and evaluate a systematic scheme for identifying protectable innovations in program behavior. My secondary objectives were to analyze the descriptive schemes used in existing regimes and to suggest a workable scheme for describing behavioral innovations under a market-oriented regime. Through these objectives, I was aiming to clarify some of the key issues underlying the current debate on anti-cloning protection of software. Although a comprehensive analysis of the debate is beyond the scope of this study, I believe that its findings are significant, in that they add to the groundwork necessary for assessing the actual merits and drawbacks of various policy options.

This study can also be viewed as a natural extension of the work presented in the Manifesto. In its conclusion, Samuelson et al. recommend further study on the registration of innovative software products (2431). They note that developing a registration system for software may prove to be difficult for a number of reasons (Samuelson et al. 2417). By contributing workable schemes for identifying and describing innovations in program be-

havior, this study represents an important first step in developing an effective and efficient registration system for innovative software products.

1.5 Brief Overview of the Methodology

To meet the stated purpose, this study was organized into two phases. During the first phase, I began by exploring the nature of computer programs and that of technical innovation in order to construct a model definition of the phrase "behavioral innovations embodied in computer programs." Based on this definition, I then developed a systematic scheme for identifying innovations in program behavior, to be used under existing and proposed legal regimes. Next, I surveyed the descriptive schemes used in existing regimes and assessed their usefulness in describing behavioral innovations. Finally, I concluded the first phase of the study by suggesting a workable scheme for describing behavioral innovations under a market-oriented regime.

The second phase involved an extensive case study of two software products in the rapidly growing field of World Wide Web applications. The World-Wide Web is a popular networking scheme for organizing and accessing data on the Internet. The two software products under consideration were the Mosaic browser from the National Center for Supercomputing Applications and the Netscape Navigator browser from Netscape Communications Corporation. Through the case study, I sought to evaluate the feasibility and usefulness of the proposed scheme for identifying behavioral innovations in software. I based the evaluation primarily on the goals and principles listed in the Purpose section of this document. For further guidance, I relied on the principles of market economics and market preservation, as presented in the Manifesto.

1.6 Assumptions and Limitations

In many respects, this study was analogous to an engineering exercise: I was aiming to develop and evaluate workable solutions to problems in design. Throughout this study, I had to consider both design goals as well as design constraints. Unlike other engineering exercises, however, this study utilized qualitative rather than quantitative methods. Accordingly, it is difficult to prove, in a formal sense, that the schemes proposed in this study will yield optimal or even desirable results once implemented. On the other hand, I believe that the value of this study, as with other studies of its kind, lies not only in the final results but in the process of arriving at those results. In fact, the questions asked during this study may prove to be just as instructive as the answers offered. Accordingly, the schemes proposed in this study should be viewed as prototype frameworks rather than final implementations.

Furthermore, this study utilized a case study of the Mosaic and Navigator Web browsers. I used the case study to evaluate the feasibility and usefulness of the proposed scheme for identifying behavioral innovations in software. It is important to note, however, that all case studies by definition involve a limited number of samples. Accordingly, care must be taken when generalizing the findings of any single case study. In light of these considerations, more test cases should be analyzed to validate the findings of this study. Possible test cases are discussed in the Suggestions for Further Research section.

Chapter 2

Phase 1: Proposed and Existing Schemes

2.1 Defining Behavioral Innovation

The phrase "behavioral innovations embodied in computer programs" comprises two distinct notions. The first notion is that of program behavior. Computer programs, unlike other textual works, "behave"; that is, they solicit input, perform computations, display results, and so on (Samuelson et al. 2316). In fact, computer programs are created and used in order to produce some useful behavior, giving rise to the view of computer programs as machines (Samuelson et al. 2317). The other distinct notion is that of technical innovation. In the Manifesto, Samuelson et al. define innovation as the act of producing "products of skilled effort that advance the state of the art but do not meet the patent standard of nonobviousness" (2330). This definition implies that an innovative product should at least satisfy the patent standards of usefulness and novelty to merit protection. In other words, an innovative machine or process must serve a useful function and must not be already known, used, or available to others.

Although the above definitions of program behavior and technical innovation are adequate for the purposes of general discussion, a more substantive understanding of these notions is needed in order to develop a workable scheme for identifying innovations in program behavior. Accordingly, the first stages of the study involved a survey of the existing literature on computer programs and technical innovation. Drawing from a wide range of articles and texts, the survey yielded a number of important observations and insights. Based on the results of the survey, a model definition of the phrase "behavioral innovations embodied in computer programs" was constructed. The objective was to arrive at a

definition that would distinguish protectable advances in technology from nonprotectable changes or modifications. The results are presented in this section.

2.1.1 Dimensions of Program Behavior

The concept of program behavior consists of three distinct dimensions: functionality, performance, and user interface. The dimension of functionality refers to the purpose of a computer program. Like most machines, computer programs are designed to serve a useful purpose, often called a "function" (Samuelson et al. 2320). For example, word processing programs, such as Microsoft Word and FrameMaker, are designed to provide users with an enhanced method of authoring and editing documents. As alternatives to the traditional methods of handwriting and typing, these programs give users the illusion that they are working with an extraordinary type of paper (Samuelson et al. 2324). Unlike ordinary paper, this "virtual paper" allows users to recover automatically from deletions and even check for spelling errors (Samuelson et al. 2324).

The second dimension of performance refers to the notion of efficiency. In the case of computer programs, efficiency is primarily measured by the time it takes to perform a given function or set of functions (Yourdon and Constantine 10). Consider two programs that are performing identical functions on identical systems under identical conditions. In this case, the program that completes its function faster than the other is considered more efficient. Other factors of efficiency include the amount of memory and disk space required to perform a given function or set of functions (Yourdon and Constantine 10).

Finally, the third dimension of user interface refers to the means through which a computer program and a human user communicate with one another (Nimmer 1-61). At a minimum, the user interface of a computer program includes the formatting of inputs and outputs as well as the organization of commands. Most modern interfaces, however, com-

prise much more: They include the layout of screen displays, the wording of menu entries, and the design of program icons. In the future, the user interface dimension will expand in scope due to advances in technology. For instance, user interfaces of the future will most likely utilize a greater range of media, including three-dimensional displays and enhanced audio capabilities.

At this point, it is important to note that the latter dimensions of performance and user interface necessarily depend on the dimension of functionality, whereas the dimension of functionality does not necessarily depend on the other two dimensions. In other words, it is nonsensical to consider how well a machine performs or how it interacts with the user without considering its function. On the other hand, it makes perfect sense to consider the function of a machine without considering its performance or user interface. For instance, both a bicycle and a car serve the identical function of transportation. Yet, they differ greatly in their performance and in the way they interact with the user.

2.1.2 Definition of Technical Innovation

During the last two decades, numerous studies and discussions have focused on the concept of technical innovation. With few exceptions, the existing literature can be classified into five distinct categories based on subject area:

1) Promotion of innovation (Haustein and Maier)

2) Measurement of innovation (Clark and Staunton)

3) Regulation of innovation (Hill)

4) Evaluation of innovation (Udell, O'Neill, and Baker)

5) Forecasting of innovation (Henry)

Interestingly, most of these studies have approached the concept of innovation from a

macroscopic perspective, examining the patterns of innovation within and across corporate and national boundaries. Taken as a whole, these studies advance a number of important observations regarding the role of technical innovation in today's global economy.

2.1.2.1 Definition Used in Innovation: A Challenge to the Engineer

Of all the studies and discussions surveyed, Innovation: A Challenge to the Engineer by Knut Holt presents the most straightforward and applicable analysis of technical innovation. Holt organizes his analysis around the following four questions (2):

1) What is innovation?

2) What are the basic stages of the innovation process?

3) What is the result of the innovation process?

4) What are the different types of innovation?

In the first part of Innovation, Holt discusses the confusion created by the different ways in which people have used the term "innovation." He notes that much of the confusion stems from the fact that some of the definitions refer to the process of innovation, whereas others refer to the result of this process (4). For the purpose of his analysis, Holt interprets the concept of innovation in a broad sense, covering both the innovation process and its result (3).

According to Holt, the process of innovation "encompasses the use of knowledge for the generation and practical application of new and viable ideas" (3). Based on this definition, Holt identifies four distinct stages in the innovation process: discovery, invention, application, and diffusion (4).

As the first stage, discovery is the act of bringing forth previously unknown information through pure or exploratory research (Holt 4). In comparison, invention is the act of

applying newly acquired information to solve a particular problem or improve existing technology (6). Often, discovery and invention are completely unrelated (Holt 4). In other cases, they are connected indirectly with considerable time-lag; still, in some cases, they are linked directly, as illustrated by the discovery and invention of the transistor technology (Holt 5). Throughout these discussions, Holt uses the term "invention" in a broad sense; he is not referring to the rigorous notion of invention as used in the patent regime. As the third stage of the innovation process, application involves converting an invention into an actual product that is introduced into the marketplace or taken into practical use (Holt 8). Finally, diffusion refers to the widespread dissemination and adoption of newly discovered technology; this stage may take many years to complete (Holt 9).

The result of the innovation process, according to Holt, is the innovation itself (Holt 12). In other words, the term "innovation" refers to a product or process embodying the successful application of newly acquired knowledge.

It is important to note the distinction between invention and innovation, as understood by Holt. An invention--even if it is technologically sound--will not always come into practical use (Holt 3); conversely, an idea does not have to be sold nor used in order to qualify as an invention. In contrast, an innovation by definition must be commercially viable (Holt 3); in other words, an idea must lead to the introduction of an actual product in order to qualify as an innovation. Therefore, successful commercialization is a precondition for innovation, but not for invention.

In his analysis, Holt identifies three distinct classes of innovation: original innovations, adopted innovations, and radical improvements (Holt 13). Original innovations by definition reflect a genuine contribution of a pioneering nature, representing the greatest degree of change and novelty (Holt 13). They can either be basic innovations, representing a major advance of the primary function, or incremental innovations, representing an im-

provement of existing technology (Holt 13). In contrast, adopted innovations represent the lowest degree of change and novelty (Holt 17). They can either be pure adoptions with no change at all or adapted innovations with minor changes to accommodate special needs (Holt 18). Finally, radical improvements involve significant modifications of existing products or processes, representing an advance in the state of the art (Holt 18). In these cases, the improvements exhibit the characteristics of incremental innovations (Holt 18-19).

2.1.2.2 Proposed Definition of Technical Innovation

Based on Holt's analysis, it is clear that basic and incremental innovations as well as radical improvements of existing technology promote consumer welfare and thus merit legal protection. On the other hand, adopted innovations and mere clones fail to represent legitimate advances in the state of the art and thus lie beyond the scope of legal protection. Taking into consideration the various types of protectable innovations and improvements, the following definition of "technical innovation" was constructed:

> "an advance in the state of the art that produces a novel and useful result of commercial significance"

In the above definition, the term "advance" denotes a change or departure from prior art, and the term "result" denotes a discernible effect. Similarly, the term "commercial significance" refers to a direct connection to increased sales or increased use of a product embodying the claimed innovation. Under this interpretation, the first product embodying the claimed innovation need not be a commercial success. In fact, many commercially important innovations in software first appeared in products that were themselves commercially unsuccessful (Samuelson et al. 2410). What is required is that the claimed innovation contributes to the commercial success of any one product embodying it.

2.1.3 Proposed Definition of "Behavioral Innovation Embodied in Computer Programs"

Building on the preceding analysis, the following definition of the phrase "behavioral innovation embodied in computer programs" was constructed:

> "an advance in the state of the art that produces a novel and useful result of commercial significance in the functionality, performance, or user interface of a computer program"

The proposed definition offers a number of advantages. First, it supports a program-by-program analysis as opposed to a macroscopic evaluation. As previously noted, many of the definitions used in earlier studies sought to characterize the level of innovation within and across companies and countries, making them unsuitable for use in an intellectual property regime. In contrast, the proposed definition allows one to determine whether a given program possesses innovative behavior and thus merits legal protection.

Second, the proposed definition utilizes the existing patent standards of novelty and usefulness. By incorporating existing standards, it facilitates acceptance by the legal community. In addition, it enables the legal community to leverage existing precedents and guidelines from the patent regime.

Finally, the proposed definition focuses on the valuable aspects of technology by employing the standard of commercial significance. This definition, unlike the definition of invention used in the patent regime, explicitly specifies commercial significance as a necessary condition. By imposing a requirement of commercial significance, the proposed definition excludes trivial or inconsequential changes in technology from legal protection while supporting anti-cloning measures.

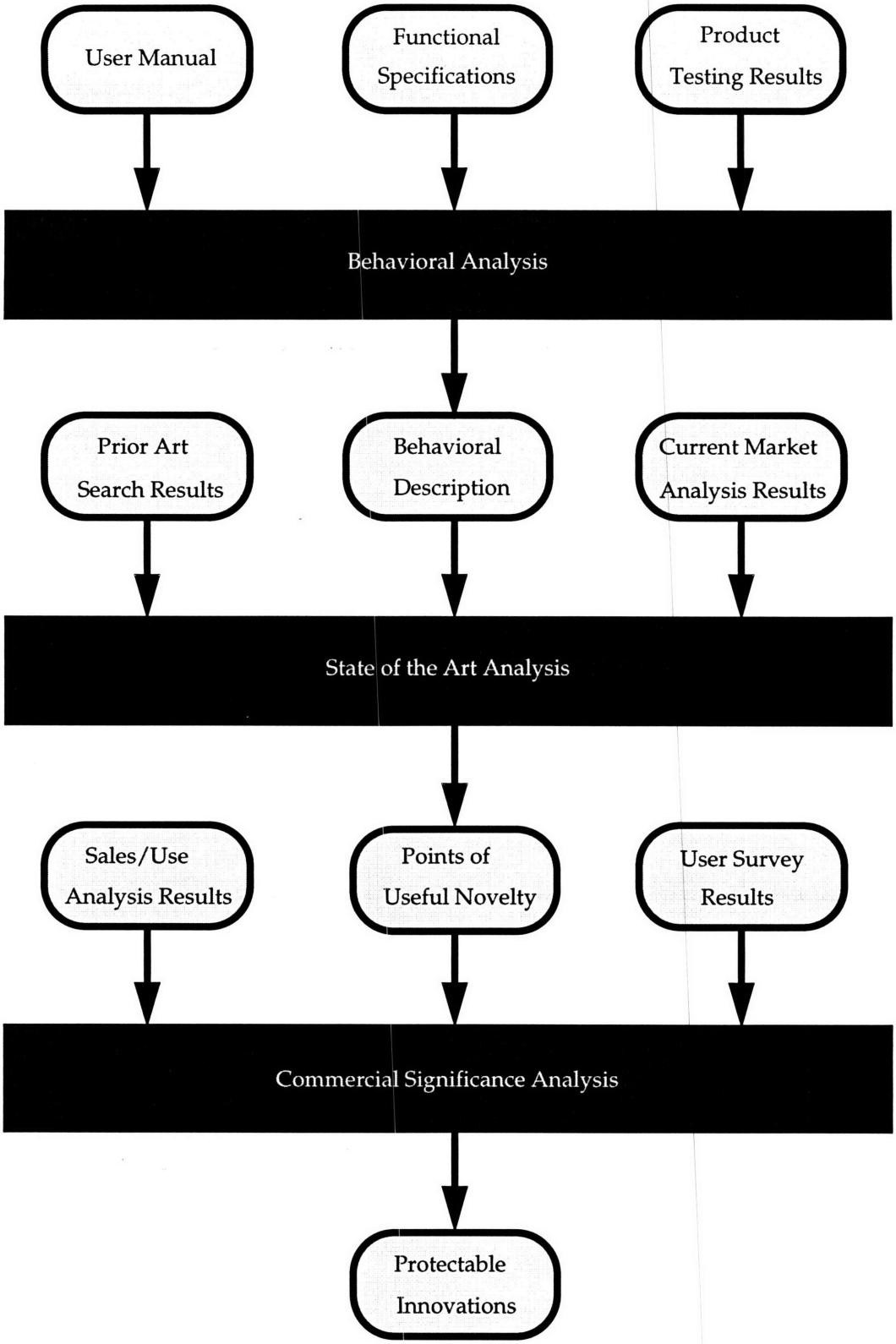
It is important to note that a second comer is likely to clone behavioral elements that are considered commercially significant; otherwise, there is no real incentive for the second comer to develop a clone. Thus, if the commercially significant elements of new prod-

ucts were to be protected, software developers would still be encouraged to invest in the research and development of innovative products.

2.2 Proposed Scheme for Identifying Innovations in Program Behavior

Based on the proposed definition presented in the preceding section, a three-stage scheme for identifying behavioral innovations embodied in computer programs was developed. The scheme consists of the following stages: behavioral analysis, state of the art analysis, and commercial significance analysis.

Figure 2.1: Diagram of the Proposed Scheme



2.2.1 Behavioral Analysis

The objective of the behavioral analysis stage is to analyze a given computer program along the dimensions of functionality, user interface, and performance--in short, to capture its behavioral characteristics. The primary inputs for this stage are user manuals, functional specifications, and program testing results. User manuals, also known as "program manuals," are publications written by software developers and distributed to users. They usually contain detailed descriptions of program features as well as instructions on how to interface with the program--all written in non-technical prose and illustrated with helpful diagrams and figures. Readily accessible, these manuals serve as an excellent starting point for understanding the program behavior of a computer program (Samuelson et al. 2316).

Whereas user manuals are essentially marketing documents, functional specifications, on the other hand, are written by engineers for other engineers (Yourdon and Constantine 5). Usually written in technical prose, these documents describe in precise terms the inputs to be supplied by the user, the outputs to be delivered to the user, and the algorithms to be used (Yourdon and Constantine 5). They may also include a number of constraints concerning the user interface, performance, and target user base. The overall objective is to capture the user's conception of a computer program in precise and technical terms (Yourdon and Constantine 5). Accordingly, the functional specification for a computer program should not specify any decisions related to the design or the implementation of the program, such as the programming language to be used (Yourdon and Constantine 5-6).

A third source of information on program behavior is program testing, also known as "black box" testing. Program testing involves installing and using a computer program in order to acquire an empirical, hands-on understanding of its program behavior (Samuelson et al. 2317). In most cases, program testing is the only reliable way of gauging the perfor-

mance of a computer program. It is also useful in cases where user manuals are not sufficiently descriptive or where functional specifications are not accessible.

The final output of the behavioral analysis stage is a comprehensive behavioral description of the computer program under consideration. The description should be organized into three distinct sections according to the three dimensions of program behavior: functionality, user interface, and performance.

2.2.1.1 Functionality

As noted earlier, the functionality of a computer program is by definition its purpose for existence: Most computer programs are designed to serve a useful purpose (Samuelson et al. 2320). Unlike other machines, however, most computer programs are designed to process or manipulate information rather than physical objects. As information machines, computer programs generally have known inputs, known outputs, and known transforms--where transforms represent transformations of data from one form to another (Yourdon and Constantine 43).

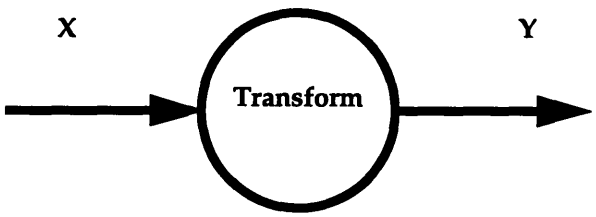
For most users, these three dimensions constitute their entire understanding of a computer program's functionality (Yourdon and Constantine 5). Accordingly, most users perceive a computer program as a collection of "black boxes," where "black boxes" are systems which can be fully exploited without knowing what is inside them (Yourdon and Constantine 22-25). Some programs are rather simple and may consist of only one transform and a limited set of inputs and outputs; others may be very complex, with a plurality of transforms and an unlimited number of inputs and outputs. Most software products on the market today are quite complex and offer a wide range of transforms organized into a hierarchy of features.

For illustration, consider a word processing application. Based on the black box approach, the user can perceive the application as a collection of transforms that have been organized into the following classes of features: authoring, viewing, editing, and file processing. At any given point in time, the input is a virtual document, residing in system memory or permanent storage; and the output is a modified version of the same document.

Imagine that the user has invoked a viewing feature of the application in order to change the font size of the current document. Upon receiving this command, the word processor takes the current document as input and outputs a modified document with an updated font size. In this case, the user is not required to know anything about the internal design of the word processor--it could be written using C or Pascal or even Fortran. From the user's perspective, the word processor is simply a collection of black boxes, one of which happens to contain a transform for changing the font size of a document.

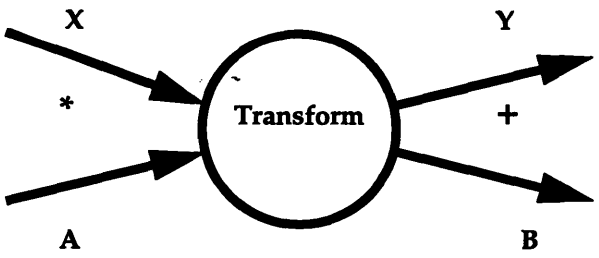
Based on the black box approach to analyzing computer programs, Edward Yourdon and Larry Constantine has developed a design technique called "data-flow graphs" (43-46). Since its introduction in the late 1970s, data-flow graphs have emerged as one of the basic tools of software developers. A data-flow graph by definition deemphasizes the procedural aspects of computer programs and instead focuses on the relationship between the input and the output data (Yourdon and Constantine 43). Accordingly, a data-flow graph typically consists of three types of elements: (1) small bubbles representing transforms; (2) a set of labelled arrows representing input data; and (3) another set of labelled arrows representing output data. Figure 2.2 illustrates a simple transform with a single input stream and a single output stream (Yourdon and Constantine 44).

Figure 2.2: Simple Transform Bubble



In a data-flow graph, a transform may require data from more than one input stream to produce its outputs. If two adjacent data streams are both required for a transform, an asterick (*) should be placed between them; the asterick is formally referred to as a "conjunction operator." If only one of the two adjacent streams is required, a ring-sum operator (+) should be used to denote an either-or situation; the ring-sum operator is formally referred to as a "disjunction operator." Figure 2.3 illustrates a transform with conjunctive input streams and disjunctive output streams (Yourdon and Constantine 44).

Figure 2.3: Transform Bubble with Conjunctive Inputs and Disjunctive Outputs



Software developers typically use data-flow graphs when following a structural design strategy called "transform analysis" (Yourdon and Constantine 188). As a form of top-down strategy, transform analysis seeks to produce initial designs that exhibit high levels

of modularity (Yourdon and Constantine 187). Generally speaking, transform analysis comprises the following four major steps (Yourdon and Constantine 188):

1) Restating the functional specifications as a data flow graph

2) Identifying the afferent (i.e. input) and efferent (i.e. output) data elements

3) First-level factoring

4) Second-level factoring of afferent, efferent, and transform branches

In addition to the transform analysis strategy outlined above, data-flow graphs can also be used to support the proposed scheme for identifying behavioral innovations in software. Data flow graphs are ideal because they are required to reflect only the external aspects of a computer program. Using these graphs, it is possible to analyze a computer program as a collection of black boxes, yielding a comprehensive and complete understanding of its program behavior. In addition, data flow graphs have been tested extensively by the engineering community in developing software products and by the legal community in drafting patent applications.

2.2.1.2 User Interface

Generally speaking, most computer programs require some level of interaction with the user. The means through which such interaction takes place is called a "user interface" (Nimmer 1-61). In today's highly competitive market, the user interface is a very important component of a software product. A well-constructed interface will attract users and facilitate use, whereas a poorly-designed interface may significantly limit the commercial potential of a given product.

A straightforward way to capture the user interface of a computer program is to take snapshots of the screen display (Nimmer 1-61). Snapshots are particularly effective be-

cause most computer programs relay information to the user through a monitor. When properly taken, these snapshots can contain all the information necessary to identify the command hierarchy, the formatting of inputs and outputs, and the screen layout of a computer program. In fact, the Copyright Office currently permits software developers to register the user interface of their products by depositing copies of screen displays along with identifying portions of the program code (Nimmer 1-119). These copies can be fixed in a wide range of media including printouts, photographs, drawings, and videotape reproductions (Nimmer 1-119).

2.2.1.3 Performance

Although the user interface is an important factor, many users regard performance as the single most important dimension of program behavior. For these users, a computer program with a poorly designed interface is bothersome, whereas a slow program is simply not acceptable. As a case in point, the text-based shell from Quarterdeck has enjoyed considerable success in recent years, despite competition from graphics-oriented interfaces like the one used in the Windows operating system. Why? Because the text-based shell offers superior performance, both in terms of speed and memory requirements.

The most accurate way to capture the performance of a given computer program is through program testing. Despite the claims of software developers, actual performance may vary significantly, depending on the configuration of the host system and other external factors. Accordingly, the dimension of performance should be evaluated by comparing two competing programs, which are executing identical functions under identical conditions.

Although performance is definitely an integral dimension of program behavior, it becomes an issue for most users only when a significant gap exists between competing pro-

grams. That is, most users are not likely to consider a marginal difference in performance when deciding between two programs.

2.2.2 State of the Art Analysis

As the second stage of the proposed scheme, the state of the art analysis seeks to filter the novel elements of program behavior from existing technology. This type of filtering is necessary because novelty is a precondition of innovation. In other words, an innovation must advance the state of the art: Software products that merely implement existing technology with trivial or minimal changes cannot be considered innovative.

The state of the art analysis requires two types of input: prior art search and current market assessment. The search for prior art is the act of finding existing or disclosed technologies that are substantively related to the useful behavior of a given computer program. The search should cover as many resources as possible including patent databases, copyright databases, software repositories, and existing literature.

In addition to a standard prior art search, the state of the art analysis should also involve a comprehensive assessment of the current market. This type of assessment is crucial because the software industry has historically evolved without relying heavily on legal protection (Nimmer 2-9). As a result, the bulk of software-related innovations have not been documented in public databases (Samuelson et al. 2326). In reality, the patent database covers only a small fraction of all the behavioral innovations embodied in existing software (Nimmer 2-9), and the copyright database only contains, at best, information on the user interfaces of computer programs.

Although some private databases have been developed to address the inadequacies of public databases (Aharonian), it is still difficult to develop an accurate understanding of the existing state of the art prior to the introduction of a computer program. The most reli-

able way to develop such an understanding is to survey related software products currently on the market. Descriptions of current products can usually be found in trade journals, sales guides, and company catalogs.

After assessing the existing state of the art, the next step is to identify the novel elements of useful behavior embodied in the given program. Throughout this process, it is important to consider that new elements are generally novel combinations of existing elements (Nimmer 2-47). Accordingly, a new element of program behavior is likely to be a novel combination of already known functionality, user interface, or performance. For instance, a computer program that combines word processing features with data encryption may be considered novel, even though word processing and data encryption are not novel in of themselves. Similarly, a computer program that combines a unique three-dimensional user interface with word processing features may be considered novel, even though it may introduce nothing new in terms of functionality. Finally, a computer program that provides an order of magnitude increase in performance over existing programs may be considered novel, even though it may introduce nothing new in terms of functionality or user interface.

The final output of the state of the art analysis stage is a descriptive document listing all the points of novelty embodied in the given program. The points of novelty should involve one or more of the three dimensions of program behavior (i.e. functionality, user interface, and performance) and should be represented using the behavioral descriptions generated earlier. For instance, functional novelty can be represented by highlighting novel combinations of transform bubbles in a data-flow graph.

As a final reminder, it is important to note that a novel element may still infringe earlier claims of innovation by fully incorporating a claimed innovation. For example, a computer program that combines word processing features with data encryption may infringe

innovation claims on data encryption, even though the combination itself may be novel. Such programs are known in the industry as "follow-on innovations" (Samuelson et al. 2422).

2.2.3 Commercial Significance

As the final stage of the proposed scheme for identifying behavioral innovations in software, commercial significance analysis aims to separate valuable points of novelty from insignificant changes in technology. Based on the view of innovation as the commercial embodiment of novel technology, this stage of the proposed scheme introduces a requirement of commercial significance in addition to the more familiar standards of novelty and usefulness. The concept of commercial significance denotes a direct or causal relationship between a point of novelty and increased sales or use of any one program embodying it. For instance, in the *Lotus Development Corp. v. Borland International, Inc.* case, the menu hierarchy of Lotus 1-2-3 was cited as a commercially significant element of program behavior because it was a key factor in the purchasing decisions of many users: Users would purchase Borland's Quattro instead of other third-party spreadsheets because it provided menu compatibility with Lotus 1-2-3.

This requirement of commercial significance serves a crucial role in the proposed scheme by allowing the market, namely the users, to determine which advances in technology are valuable and thus merit legal protection. Without this requirement, it would be very difficult to assign value or significance to a particular point of novelty (Holt 13). The other realistic option is to consider every point of novelty as protectable innovation. This type of broad characterization would likely lead to a state of overprotection and result in unnecessary waste of resources.

Inputs to the commercial significance analysis are sales or use reports, market share reviews, and user surveys. Of the three, user surveys are probably the most useful in determining whether a particular point of novelty is directly related to increased sales or use of a program.

The final output of the proposed scheme is a comprehensive listing of protectable innovations in useful behavior embodied in the given program. The listing is intended for identification purposes only and may not be suitable for use as a descriptive document in a legal regime.

2.3 Existing Schemes for Describing Innovations in Program Behavior

Over the years, the legal community has generally described computer programs as textual works, with emphasis on the structure, sequence, and organization of program texts (Samuelson et al. 2357). This emphasis on the textual aspects of computer programs can be attributed to the historical emergence of copyright law as the primary form of protection for computer programs. In recent years, however, the legal community has come to view computer programs in a different light, due to the increasing demand for patent protection (Samuelson et al. 2362). This section presents a critical survey of the descriptive schemes used in the existing legal regimes.

Currently, computer program are protected under three distinct intellectual property regimes--copyrights, patents, and trade secrets. Trade secrets aside, each of the above regimes requires software developers to disclose certain aspects of their products. This requirement of public disclosure stems from the belief that free exchange of ideas promotes creative expression and technical innovation (Nimmer 2-6). Despite sharing a common emphasis on public disclosure, the copyright and patent regimes nevertheless support two very different schemes for describing computer programs.

In many respects, the two schemes represent opposite ends of the spectrum. The copyright regime covers virtually all computer programs and employs a light registration scheme, in which software developers are required to describe their products by depositing "identifying portions" with the Copyright Office (Nimmer 1-118). In contrast, the patent regime looks beyond the textual elements of computer programs and focuses instead on the nontextual elements, such as program behavior and design. Furthermore, the patent regime employs a thorough application scheme, in which software developers are required to provide detailed descriptions of their products.

2.3.1 Descriptive Scheme Used in the Copyright Regime

Copyright laws, historically, were intended to protect and thereby promote the creative expressions of authors and artists (Nimmer 1-3). Accordingly, the registration process used in the copyright regime is designed to address literary and artistic expressions rather than technical concepts.

In contrast to the thorough application scheme of the patent regime, copyright registration involves a straightforward, three-step process: (1) filing a completed application form; (2) paying a registration fee; and (3) depositing copies of the work to be registered with the Copyright Office and the Library of Congress. Upon filing, the registration application is subjected to an examination by the Copyright Office and the Registrar of Copyrights. The examination process is limited in scope and focuses primarily on matters involving adequacy of notice and authorship of the work (Nimmer 1-116). If all of the statutory conditions are satisfied, the Copyright Office issues a registration certificate, which serves as prima facie proof of the copyright's validity.

Essentially, the application form and the deposited copies of the registered work constitute the descriptive scheme for the copyright regime. The application form is a short

document listing the title of the work to be registered, its authorship information, and its publication history. In conjunction with the application form, copies of the work must be deposited with the Copyright Office for examination purposes and with the Library of Congress for record purposes. Generally, the deposited copies must be complete, disclosing the entire work.

In the case of computer programs, however, this requirement of complete disclosure conflicts with the desires of many software developers, who wish to retain secrecy concerning the internal design of their products (Nimmer 1-118). According to these developers, secrecy is necessary in order to prevent competitors from making use of proprietary algorithms and other design elements that may lie beyond the scope of legal protection.

In recognition of this conflict, the Copyright Office has recently amended its policies, requiring software developers to deposit only "identifying portions" of their work. The Copyright Office defines "identifying portions" as the first and last twenty-five pages of the source code or the entire source code if it is less than fifty pages long.

As noted earlier, the Copyright Office also permits software developers to register the user interface of their products by depositing copies of screen displays along with identifying portions of the program code. Copies can be fixed in a wide range of media including printouts, photographs, drawings, and videotape reproductions.

2.3.2 Descriptive Scheme Used in the Patent Regime

Whereas the copyright regime is intended to protect literary and artistic expressions, the patent regime is intended to promote advances in science and technology (Nimmer 2-2). In support of this goal, the patent regime provides a set of exclusive property rights to the developers of inventive devices or processes. As of June 8, 1995, the property rights granted under a patent empower the inventor to exclude others from using, making, or

selling the invention for a period of twenty years from the original date of filing (U.S. Patent and Trademark Office). These rights, unlike their copyright counterparts, are not granted to all innovations in science and technology--only those representing significant advances in their respective fields are considered for patent protection (Nimmer 2-5).

Under existing patent laws, significant advances worthy of protection are distinguished from the incremental progress of science and technology based on the standards of novelty, usefulness, and nonobviousness (Nimmer 2-6). Derived from Section 102 of the Patent Act (i.e. 35 U.S.C.), the novel standard requires the claimed invention to be new and not already known or used by others. The usefulness standard is just as straightforward, requiring the claimed invention to be a working device or process that serves a useful function. The third standard of nonobviousness is the most elusive of the three and is often the most significant in the examination of a patent application (Nimmer 2-7). This standard is rooted in the principle that patent protection should be reserved for inventions representing a significant departure from the incremental and predictable advance of science or technology: The claimed invention must not be obvious to those skilled in the art.

The above standards, often called the "inventive standards," form the basis of the patent regime, giving rise to its rigorous application and examination processes. These processes, collectively known as "patent prosecution," are significantly more complex than their copyright counterparts. In most cases, patent prosecution is a costly and time-consuming process, requiring thousands of dollars in legal fees and lasting over two years in duration.

The first step in obtaining a patent is the preparation of a patent application. Unlike the short application form used in copyright registration, a patent application is a detailed document listing all the technical and legal information required for the examination process.

Preparing a patent application usually requires the services of a patent attorney and may last several months.

After completion, the application is submitted to the U.S. Patent and Trademark Office (PTO) for examination. The PTO is responsible for receiving, processing, and storing all submitted patent applications. Upon receiving an application, the PTO assigns a patent examiner to review the application in order to determine whether the claimed invention satisfies the inventive standards of novelty, usefulness, and nonobviousness. Typically, the patent examiner focuses on the last standard, as she attempts to evaluate the nonobviousness of the claimed invention by comparing it to prior art.

Based on this preliminary examination, the PTO rejects a significant number of patent applications and returns an even greater number to the applicants for further clarifications and possible revisions (Nimmer 2-5). This process is repeated until the PTO reaches its final decision; at this point, it either rejects the application for failing to satisfy all of the statutory requirements or issues a patent for the claimed invention. An issued patent, similar to a registration certificate under a copyright regime, gives rise to a presumption that the PTO has properly processed the application and has decided to grant patent protection to the claimed invention (Nimmer 2-5).

Essentially, the patent application along with possible clarifications and revisions serves as the descriptive scheme for the patent regime. When drafting a patent application, the applicant must satisfy the following objectives: (1) to disclose publicly all relevant aspects of the claimed invention (Nimmer 2-10); (2) to provide background information on prior art (Nimmer 2-9); and (3) to define the terms and scope of the legal protection being sought (Nimmer 2-7). These objectives are based on the principle that a patent should be viewed as a fair exchange between the people and the inventor: The people offer the inventor exclusive rights to an invention; in exchange, the inventor offers the people a sig-

nificant advance in science or technology along with a full description of the invention (Calderwood).

In meeting the first two objectives, an applicant must include a "specification" section in the patent application. This section is basically a technical document that enables others skilled in the art to make and use the claimed invention without undue effort and experimentation (Calderwood). In addition, the specification section must disclose the best mode contemplated by the applicant for practicing the claimed invention (Calderwood). Failure to meet either requirement may result in the invalidation of the patent.

The specification section generally comprises the following seven subsections (Calderwood): (1) Field of the Invention; (2) Background of the Prior Art; (3) Summary of the Invention; (4) Brief Description of the Drawings; (5) Detailed Description of the Preferred Embodiment; (6) Drawings; and (7) Abstract. Although a standard format is not required by the PTO, most patent applications include all of the above subsections or their equivalents.

In the "Field of the Invention" subsection, the applicant identifies the technical areas pertinent to the claimed invention. This subsection is included solely for informational purposes and is not intended to restrict the scope of the patent protection. That is, a patent drawn to a particular field of science or technology would nevertheless cover the use of the same invention in a different field.

Under the "Background of the Prior Art" subsection, the applicant recites and evaluates the existing state of science or technology prior to the claimed invention. This subsection serves a dual purpose: (1) to develop a motivation for the claimed invention by discussing the problems and disadvantages of the prior art; and (2) to facilitate the examination process by providing background information necessary for evaluating the novelty

and nonobviousness of the claimed invention. Intentional failure to disclose all relevant prior art may result in the invalidation of the patent.

The actual description of the claimed invention begins with the "Summary of the Invention" subsection. Under this subsection, the applicant presents a high-level summary of the key functionality and structure underlying the claimed invention. By disclosing both functionality and structure, the applicant must enable others skilled in the art to make and use the invention.

With the summary serving as a roadmap, the applicant proceeds to describe the claimed invention under the "Detailed Description of the Preferred Embodiment" section. The preferred embodiment is typically a working machine or device that embodies the claimed invention and illustrates the best mode contemplated by the applicant. In describing the preferred embodiment, the applicant is not required to use a particular style or terminology; she is only required to present sufficient detail to allow others skilled in the art to make and use the claimed invention in the best mode contemplated. The applicant is also allowed to include figures and diagrams to illustrate the claimed invention or prior art; she can present them under the "Drawings" subsection and describe them under the "Brief Description of the Drawings" subsection.

Finally, under the "Abstract" subsection, the applicant summarizes the entire specification section, focusing on the most novel aspects of the claimed invention. In addition, she may highlight the key aspects of the "claims" section.

Whereas the specification section is a technical document, the claims section is essentially a legal document, which defines the boundaries of the property rights to be granted if and when the patent is issued (Calderwood). For the most part, patent claims are written in technical language with legal overtones: Many of the terms commonly used in drafting patent claims, like "means for" and "comprising," have been litigated in court and have

precise meaning under case law. In addition, the claims often use technical terms and concepts without explanation, given that these terms and concepts are adequately defined or described in the specification section (Calderwood). Consequently, the claims section is not intended to be a stand-alone document but to be read in conjunction with the rest of the patent application.

The organization of the claims are rather straightforward. There are two types of claims: independent and dependent (Calderwood). To infringe an independent claim, a device or a process must include every element recited in the claim or its equivalent (Calderwood). Similarly, to infringe a dependent claim, a device or process must include every element or its equivalent recited in the claim and every element or its equivalent of every related claim, tracing back to an independent claim (Calderwood).

In most cases, the claims are structured and written according to the category under which the invention is being claimed. For functional inventions, the patent regime recognizes four distinct categories (Nimmer 2-8): processes, machines, manufactures, and compositions of matter. These categories are not mutually exclusive: An invention may be claimed under two or more of the above categories.

For instance, consider a patent application drawn to an invention for automatically resolving address conflicts among adapters connected to a single bus. In this case, the applicant could file two independent sets of claims. She could structure the first set of claims as process patents, beginning with the phrase "A method for resolving address conflicts among adapters in a computer system, comprising the steps of." In contrast, she could structure a second set of claims as device patents, beginning with the phrase "An apparatus for resolving address conflicts among adapters in computer system, comprising." Although both sets of claims relate to the same invention, they differ in their scopes of

protection and standards of infringement. This particular invention is actually covered under the U.S. Patent 5,386,515 "Automatic Input/Output Address Conflict Resolution."

2.3.3 Analysis of the Descriptive Schemes Used in Existing Regimes

Neither of the descriptive schemes presented in the preceding sections is well-suited to describing behavioral innovations embodied in computer programs. For the most part, both schemes have evolved without taking into consideration the special properties of behavioral innovations in software (Nimmer 2-9).

When it comes behavioral innovations, the descriptive scheme used in the copyright regime poses serious conceptual and procedural difficulties. First, it does not support the notion of technical innovation and the related standards of novelty and usefulness. Instead, the entire scheme is based solely on the limited standard of originality (Samuelson et al. 2353). Under this standard, a copyrightable work is not required to be unique or exceptional; it is only required to be an original work of creative expression. Accordingly, the copyright scheme does not possess the necessary vocabulary for distinguishing innovative products from clones.

Second, the copyright scheme fails to define clearly the scope of legal protection when it comes to nonliteral copying. In fact, Section 102(b) of the Copyright Act explicitly states: "In no case does copyright protection extend to any...procedure, process, system, [or] method of operation...regardless of the form in which it is described, explained, illustrated, or embodied in such work" (Samuelson et al. 2351). The above language in conjunction with its statutory history suggests that nonliteral elements of computer programs, including their program behavior, lie beyond the scope of copyright protection. Nevertheless, more and more developers are seeking to protect the useful behavior of their products by claiming copyright infringement. As a result, there are serious concerns and doubts re-

garding the proper scope of copyright protection for computer programs (Samuelson et al. 2354).

Finally, on a related point, the copyright scheme also fails to identify the dimensions of protection when it comes to nonliteral copying. Throughout the copyright registration process, software developers are allowed to deposit only the source code and screen displays of their products. Consequently, they have no opportunity to describe the nonliteral elements embodied in their products, such as the useful behavior produced during program execution. At best, they can describe the user interface by depositing copies of screen displays. Thus, even if copyrights were to protect nonliteral elements of computer programs, it would be difficult to describe these elements under the existing scheme.

Whereas the copyright scheme discloses insufficient information about the behavior and design of computer programs, the descriptive scheme used in the patent regime gives rise to the opposite problem: It requires software developers to disclose more information than is necessary for identifying and describing behavioral innovations embodied in their products. Generally, a patent applicant must provide accurate and sufficient detail to enable others skilled in the art to make and use the claimed invention without undue effort or experimentation. This "enablement" requirement benefits the larger society, as free exchange of inventive ideas will likely inspire and assist others in further advancing the state of the art (Nimmer 2-4).

There are, however, trade-offs to this requirement--trade-offs that render the patent scheme inappropriate for describing behavioral innovations in software. First, because the applicant is required to describe in detail both the functionality of the claimed invention and the means for accomplishing that functionality, the resulting patent application can be a rather lengthy document (Samuelson et al. 2345). It usually contains illustrative diagrams as well as precise descriptions of various functional units or process-related steps.

Drafting such an application can lead to costly delays that may not be acceptable, given the rapid rate of change within the software industry (Samuelson et al. 2346).

Second, the enablement requirement conflicts directly with trade secrecy protection: A software developer generally cannot receive both patent and trade secrecy protection for the same product: Nimmer writes: "Under patent system disclosure requirements, if the secret program or process is integral to the invention and unique, it must be disclosed adequately to allow replication by one skilled in the art" (Nimmer 2-13). This conflict is problematic for many developers who consider trade secrecy as the most effective and, in some cases, the only form of legal protection for the design know-how embodied in their products (Nimmer 2-10). These developers face a difficult dilemma: (1) They can either seek protection for the innovative elements of program behavior and, in the process, relinquish trade secrecy protection for the valuable elements of program design; or (2) they can forgo legal protection for the behavioral innovations, risking unauthorized cloning by a second-comer.

Finally, the descriptive scheme used in the patent regime does not allow an applicant to rely solely on high-level descriptions in disclosing the claimed invention (Nimmer 2-12). Generally speaking, the applicant must provide sufficient detail to allow others skilled in the art to generate a functional program (Nimmer 2-12). As a consequence, high-level flowcharts and verbal descriptions may not suffice. These types of descriptions, however, are ideal for identifying and describing behavioral innovations embodied in computer programs. They allow software developers to disclose their innovations quickly and concisely, enabling others skilled in the art to understand the innovations with ease.

In summary, neither of the existing descriptive schemes is especially well-suited for describing behavioral innovations in program behavior: The scheme used in the copyright

regime discloses too little information, whereas the scheme used in the patent regime discloses too much.

2.4 Proposed Scheme for Describing Innovations in Program Behavior

Based on the preceding analysis, a four-part descriptive scheme, to be used under a market-oriented regime, was developed. The objective was to develop a workable scheme that would retain the strengths of existing schemes while overcoming their most significant drawbacks. The resulting scheme is a hybrid of the descriptive schemes used in the copyright and patent regimes. The results are presented in this section.

2.4.1 Description of the Proposed Scheme

Similar to the descriptive scheme used in the patent regime, the proposed scheme consists of an "innovation application." This application is to be filed by software developers when registering their innovative products.

In drafting an innovation application, the applicant must satisfy the following objectives: (1) to provide background information on the prior state of the art; (2) to enable others to understand the claimed innovation; (3) to establish the commercial significance of the claimed innovation; and (4) to delineate the terms and scope of the legal protection being sought. These objectives are consistent with the goals of a market-oriented regime, which are to protect innovative software products from market-destructive clones and to maintain an optimal level of innovation within the industry.

Corresponding to the objectives listed above, an applicant is required to include four distinct sections in the innovation application. The required sections are as follows: (1) Prior State of the Art; (2) Description of the Innovation; (3) Commercial Significance of the Innovation; and (4) Claims.

In the "Prior State of the Art" section, the applicant summarizes the state of the art prior to the introduction of the claimed innovation; the term "the state of the art" refers to the existing state of science or technology related to the claimed innovation, including any issued patents and products currently on the market. This section is required in order to support the novelty standard: An innovative machine or process must not be already known, used, or available to others. In preparation, the applicant should conduct a thorough search, covering as many resources as possible--including patent databases, copyright databases, software repositories, and existing literature. Furthermore, the applicant should assess the current state of the market by reviewing trade journals, sales guides, and company catalogs.

Whereas the "Prior State of the Art" section can be viewed as a historical document, the "Description of the Innovation" section is essentially a technical document. Under this section, the applicant presents a detailed and technical description of the claimed innovation in program behavior. Unlike the patent regime, however, the applicant is only required to disclose in detail the external characteristics of the claimed innovation: She is allowed to describe the internal design elements in purely high-level terms. The goal is to enable others to understand the claimed innovation without precluding trade secrecy protection.

The "Description of the Innovation" section generally comprises the following subsections: (1) Summary of the Behavioral Innovation; (2) Brief Description of the Drawings; (3) Drawings; (4) Detailed Description of the Behavioral Innovation; (5) Brief Description of Related Design Elements; and (6) Abstract. These subsections are similar in content and format to their counterparts in a patent application. The only difference is that the design elements related to the claimed innovation need not be disclosed in detail. As noted

earlier, the applicant is only required to describe in detail what the claimed innovation does, not how.

The third section required in an innovation application is that of "Commercial Significance of the Innovation." In this section, the applicant offers evidence in support of the assertion that the claimed innovation directly contributed to the increased sales or use of a computer program embodying it; the evidence can be in the form of sales reports, market reviews, or user surveys. Usually, the applicant considers the commercial success of the first program embodying the claimed innovation. In cases where such a program was commercially unsuccessful, the applicant may refer to follow-on programs or describe the commercial potential of the claimed innovation. In any case, the applicant would only receive protection if the claimed innovation indeed contributed to the increased use or sales of a program embodying it.

Finally, under the "Claims" section, the applicant defines the terms and scope of the legal protection being sought. Written in technical language with legal overtones, the claims should describe the behavioral innovation in precise terms. The structure and format of the claims should be similar to those of patent claims. The only difference is that innovation claims should be written in purely behavioral terms; they should not refer to any design elements.

2.4.2 Analysis of the Proposed Scheme

When it comes to describing behavioral innovations in software, the proposed scheme overcomes most of the difficulties present in the existing schemes. First, it does not require full disclosure of the embodying program. Under the proposed scheme, the applicant is only required to describe the related design elements in purely high-level terms. Accordingly, this scheme, unlike the one used in the patent regime, is fully compatible with

trade secrecy protection. At the same time, the proposed scheme does enable others to understand the claimed innovation. This "understanding" requirement is necessary to support both anti-cloning and licensing measures.

Second, the proposed scheme clearly defines the scope and dimensions of protection in the case of computer programs. Through the innovation application, software developers are given the opportunity to describe the nonliteral elements embodied in their products and to argue in favor of granting them legal protection. This opportunity does not exist with the descriptive scheme used in the copyright regime.

Finally, the proposed scheme is based on the principles of market economics and market preservation, rather than on the notion of exclusive property. As a consequence, it can be tailored to maintain an optimal level of innovation within the software industry. None of the existing schemes is as flexible.

In addition to overcoming the drawbacks of existing schemes, the proposed scheme retains their major strengths. First, the proposed scheme is similar in format and terminology to the descriptive scheme used in the patent regime. As a result, it should be familiar to the legal community, facilitating its acceptance. Second, the proposed scheme lowers the burden of proof for applicants and reduces their requirements of disclosure. Consequently, it is compatible with a fast application and examination process, like the one used in the copyright regime.

On the other hand, there are some drawbacks to the proposed scheme. First, it places heavy emphasis on analyzing the prior state of the art. In practice, this type of analysis may be difficult to conduct, given the rapid pace of change and lack of adequate documentation. Second, the proposed scheme does not enable others skilled in the art to make and use the claimed innovation; it only enables others to understand the innovation. As a re-

sult, in licensing cases, the parties involved must exchange additional information on program design.

Chapter 3

Phase 2: Mosaic-Navigator Case Study

3.1 Background

According to Anthony Rutkowski, executive director of the Internet Society, a "digital canon" was fired on November 12, 1993 (Schatz and Hardin 895). On that day, the National Center for Supercomputing Applications (NCSA) introduced final versions of its Mosaic Web browser for three of the most popular operating system platforms: Apple Macintosh, X Window, and Microsoft Windows. Reaching across multiple platforms and across the globe, Mosaic has attracted millions of new users from all sectors of society and, in the process, has transformed the Internet from a scientific experiment in distributed computing to an international phenomenon (Schatz and Hardin 900). In addition, it has spawned a number of commercial browsers for the World Wide Web, creating one of today's most exciting and fastest growing software applications markets.

The most successful of the commercial Web browsers is Netscape Navigator--called "Navigator," for short--from Netscape Communications Corporation. According to recent estimates, Navigator has captured over 80 percent of the market share for Web browsers (Netscape Communications, Corp., "Navigator Business Update").

The main objective of the Mosaic-Navigator case study was to evaluate the feasibility and usefulness of the proposed scheme for identifying behavioral innovations in software. Often noted for its "innovative" nature, the Web browser market, represented by Mosaic and Navigator, was selected for a number of reasons. First, the Web browser market is a relatively "young" market. Mosaic was first introduced in April of 1993 (NCSA, "Introduction to NCSA Mosaic for X"), and Navigator was only introduced in December of

1994 (Netscape Communications, Corp., "Netscape Press Release 12/15/94"). Consequently, the early stages of the market are well-documented in readily accessible articles and texts. Generally speaking, access to relevant documentation is a key requirement for a successful case study: Without an accurate account of the pertinent facts and historical developments, it becomes difficult, if not impossible, to conduct a detailed study.

Second, with the emergence of the Internet as an important national topic, the Web browser market has received a great deal of publicity and attention. As a result, an unusually large number of articles and texts have been written on the topic of Web browsers. As noted earlier, access to relevant documentation is an important consideration in selecting a case study.

Finally, the Web browser market is viewed as a highly valuable sector of the software industry and an increasingly significant component of the national economy. Since its introduction, Mosaic has been recognized throughout the industry and the academia as an award-winning product. The awards given to NCSA for its Mosaic browser include: Industry Week's "Technology of the Year Award," Information Week's "Ten Most Important Products of 1994," and Fortune magazine's "Products of the Year for 1994" (NCSA, "Mosaic Awards"). Not to be outdone, Navigator has also received a number of accolades including four Excellence in Software Awards--"Best Business Software Program," "Best New Use of a Computer," "Best Internet/On-line Product or Service," and "Best Consumer Software Program"-- from the Software Publishers Association in 1995 (Netscape Communications Corp., "Netscape Press Release 3/5/96"). Given its growing importance, the Web browser market represents an ideal testing ground for the debate on anti-cloning protection of software, giving rise to some interesting and instructive questions: What is innovative, if anything, about Mosaic? And, if Mosaic is to be considered innovative, is

Navigator a mere clone of Mosaic? Or, are there aspects of Navigator that are innovative in their own right?

As with other studies of its kind, the Mosaic-Navigator case study involves a number of assumptions and limitations. First, the case study does not take into account the fact that Mosaic was developed by a non-profit organization, namely the NCSA, and was funded in part by public funds. Second, the case study is based on the following versions of each program: Version 1.0 of Mosaic for X Window, introduced in April of 1993; and Version 1.0 of Navigator for X Window, released in December of 1994. As to be expected, subsequent versions have since been released and are now available to the general public. To streamline the analysis, however, this study considers these two particular versions as being representative of their respective programs. Finally, the case study is offered as a first attempt at evaluating the feasibility and usefulness of the proposed scheme. Clearly, more test cases are needed to validate the findings of this study.

3.2 NCSA Mosaic Web Browser

According to NCSA, the technical definition of Mosaic is "a distributed hypermedia system designed for information discovery and retrieval over the Internet" ("NCSA Mosaic for the X Window System User Guide"). In more straightforward terms, Mosaic is a software application that enables users to access and display a screen of information located anywhere on the Internet and stored in a variety of formats.

Designed to provide a graphical rather than a purely textual interface to the Internet, Mosaic can be classified as a "World Wide Web browser." The World Wide Web--or the "Web," for short--is a popular networking scheme for organizing and accessing data on the Internet. Like all Web browsers, Mosaic supports the Hypertext Markup Language (HTML) for specifying hypermedia links and the Hypertext Transfer Protocol (HTTP) for

executing hypermedia jumps (Vaughan-Nichols et al. 45-49). Essentially, the HTML language is the thread with which the World Wide Web is spun, and the HTTP protocol is the means for traversing the various links of the Web.

In addition, Mosaic provides access to non-Web information on the Internet by supporting existing tools like Gopher, WAIS, and FTP. Appropriately, some observers have characterized Mosaic as "a single point of entry to the many Internet information sources" (Duval and Main 269).

As an application, Mosaic has enjoyed great popularity because it enables users to view the Internet as a global space of information rather than a distributed network of computers (Schatz and Hardin 895). Prior to the advent of Mosaic, network users had to be intimately familiar with the numerous addressing schemes, protocols, and data formats underlying the Internet in order to locate and access a desired piece of information (Ernst 17). In contrast, Mosaic allows users to ignore the physical structures of computers and to concentrate instead on the logical structures of interconnected information (Schatz and Hardin 895). With Mosaic, network experts and beginners alike can browse the thousands of different types of documents proliferating on the Internet through a common, easy-to-use interface (Ernst 19).

3.2.1 Behavioral Analysis of Mosaic

In Inside the World Wide Web, Steven Vaughan-Nichols et al. note: "[Mosaic] took off like wildfire, and the Web's popularity exploded with it. Why? Because, although Mosaic didn't really bring anything new in functionality to Web interfaces, it brought together the best ideas of existing interfaces and gave them an attractive, easy-to-use interface" (23).

The above comments suggest that a behavioral analysis of Mosaic should take into consideration not only the individual units of functionality but the nature of their combina-

tion. Furthermore, the analysis should closely examine the relationship between Mosaic's functionality and the functionality of earlier Internet applications. Finally, the analysis should consider in detail the various aspects of Mosaic's user interface.

With the above guidelines in place, the program behavior of Mosaic was analyzed along the dimensions of functionality, user interface, and performance. The analysis was based entirely on the proposed scheme for identifying behavioral innovations in software. The results are presented in this subsection.

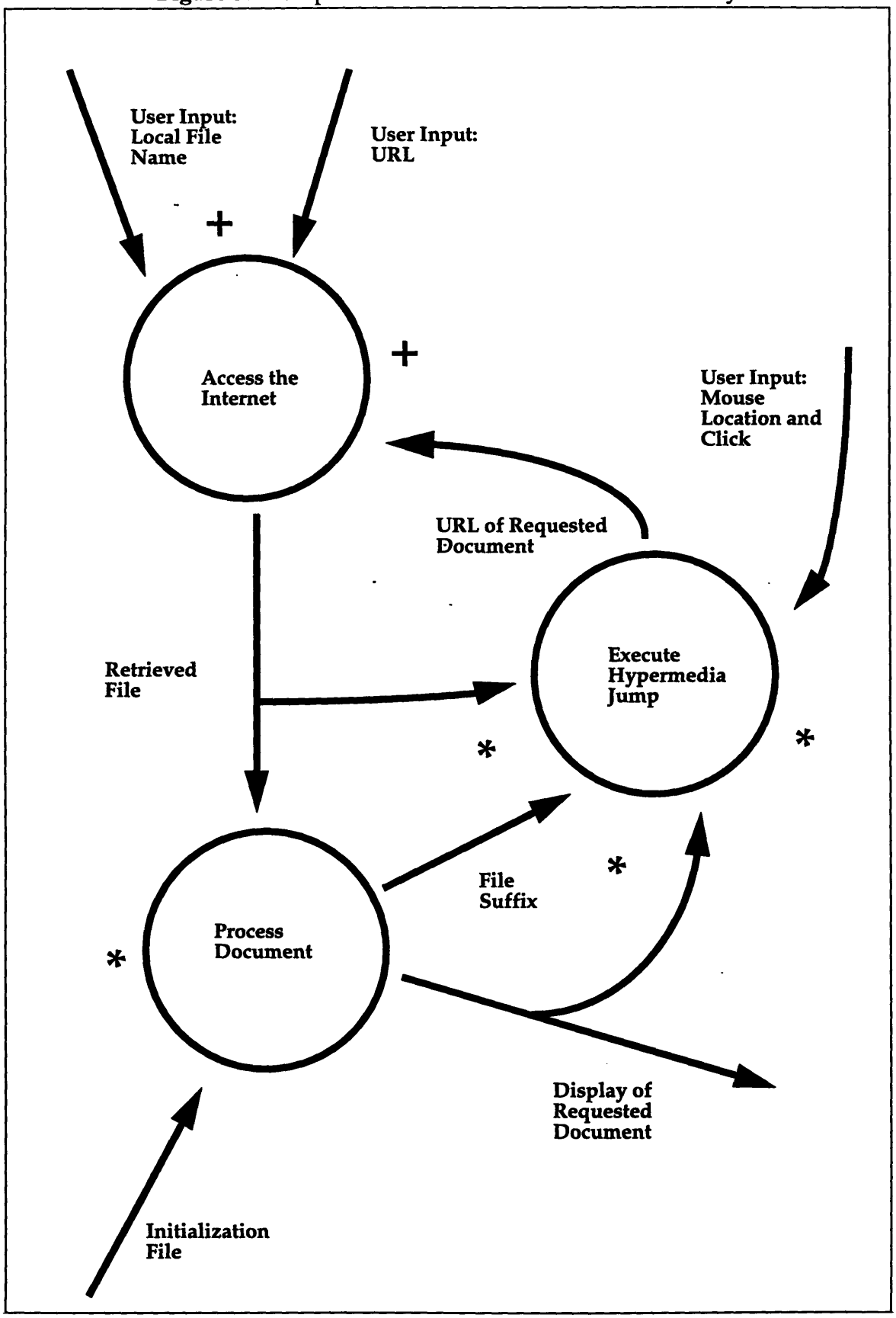
The following documents and texts were consulted in analyzing the program behavior of Mosaic: Inside the World Wide Web, The Mosaic Roadmap, "NCSA Mosaic and the World Wide Web: Global Hypermedia Protocols of the Internet," "Exploring the Internet with Mosaic," and various documents downloaded from the NCSA Web site.

Unfortunately, a working copy of Mosaic Version 1.0 for X Window could not be obtained for program testing; NCSA no longer distributes older versions of Mosaic. Instead, a copy of NCSA Mosaic Version 2.6 for X Window was installed and tested in order to acquire a hands-on understanding of Mosaic's program behavior. To reiterate, Version 1.0 of Mosaic for X Window, not Version 2.6, was the focus of this case study; however, some phases of the study relied on the program testing of Version 2.6 for guidance.

3.2.1.1 Functionality

Analyzing Mosaic as a collection of black boxes, a set of data-flow graphs describing Mosaic's functionality was constructed. There are four data-flow graphs in all--one high-level graph labelled "Graph M" and three low-level graphs labelled "Graph M.1," "Graph M.2," and "Graph M.3." Graph M captures the overall functionality of Mosaic and serves as a roadmap for the low-level graphs.

Figure 3.1: Graph M: Overview of Mosaic's Functionality



Graph M consists of three transform bubbles, each representing a major unit of functionality embodied in Mosaic:

1) "Access the Internet" represents Mosaic's ability to locate and retrieve information stored anywhere on the Internet through HTTP connections as well as through existing Internet tools like FTP and Gopher.

   The Internet is generally defined as the body of information that can be accessed through the Transmission Control Protocol/Internet Protocol (TCP/IP). First introduced in the early 1970s, the TCP/IP protocol is a flexible scheme for interconnecting packet-switching networks composed of different types of computers (Vaughan-Nichols 34).

2) "Process Document" represents Mosaic's ability to recognize and display a wide range of information including text, graphics, sound, and video.

3) "Execute Hypermedia Jump" represents Mosaic's ability to execute a hypermedia jump across the Internet through the single click of a mouse.

The inputs represented in Graph M are as follows:

1) "User Input: Local File Name" represents the user-specified name of a file stored on the client machine.

2) "User Input: URL" represents the user-specified Uniform Resource Locator of information stored anywhere on the Internet. URLs provide a universal and absolute addressing scheme for organizing the Web: A unique URL is associated with each object of information on the Web.

   Each URL is composed of a prefix followed by an encoded set of parameters. The prefix indicates the network protocol to be used (e.g. HTTP, Gopher, Telnet, or FTP) and specifies how the encoded set of parameters should be interpreted. For instance, consider the following URL: http://www.example.com/Public/Documents/Help.html. The prefix "http" indicates that the HTTP protocol is to be used in setting up a network connection. It also indicates that the rest of the URL string contains the Internet address of a server machine and the name of an object stored on that machine. The address of the server, in this case, is "www.example.com," and the name of the object is "Help.html" located in the "/Public/Documents" subdirectory.

3) "User Input: Mouse Location and Click" represents the user-specified location and click of a mouse.

4) "Initialization File" represents a file specifying the configuration of Mosaic's document processing features.

The intermediate outputs represented in Graph M are as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

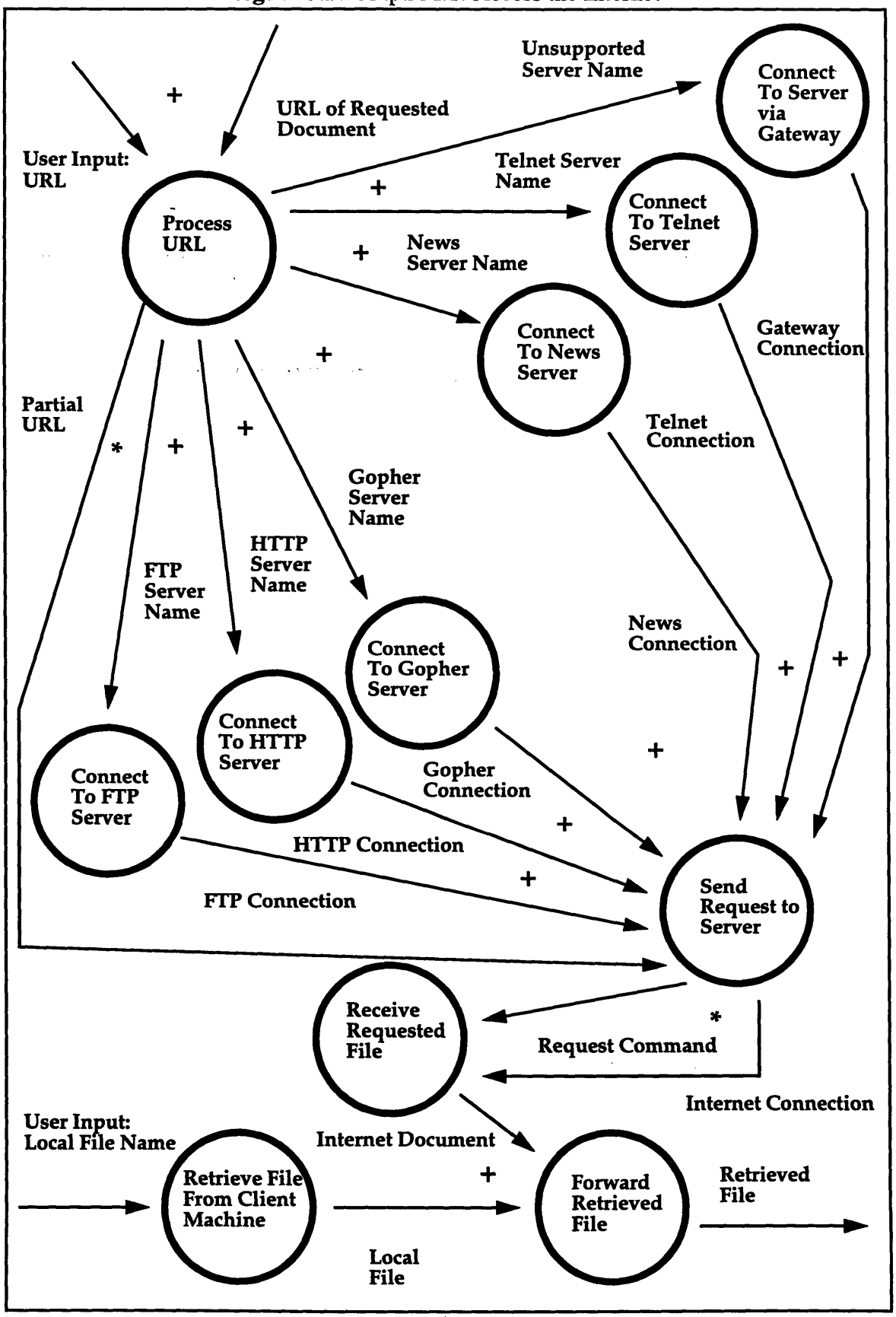
2) "File Suffix" represents the last element of a file name, indicating the file type. For instance, consider the file name "Help.html." The suffix in this case is "html," which indicates that the file is a hypermedia document stored in HTML format.

3) "URL of Requested Document" represents the Uniform Resource Locator of information stored anywhere on the Internet.

The final output represented in Graph M is as follows:

1) "Display of Requested Document" represents a screen display of the requested document, in the case of text and image files, or a playback of the requested document, in the case of audio and video files.

Figure 3.2: Graph M.1: Access the Internet



The primary function of Mosaic is to enable users to access and retrieve information over the Internet. The technical definition of the Internet is "a Transmission Control Protocol/Internet Protocol (TCP/IP)-bound network of networks and their common accessible resources" (Vaughan-Nichols 32). In more straightforward terms, the Internet is an umbrella network of smaller computer networks that are interconnected in order to facilitate the sharing and transfer of information. Today, the Internet is a global phenomenon consisting of more than two million host computers and boasting over twenty million users (Schatz and Hardin 901).

Representing the function of accessing the Internet, Graph M.1 consists of the following transform bubbles:

1) "Process URL" represents Mosaic's ability to recognize different types of network protocols based on the prefix of a URL.

2) "Connect to Telnet Server via Viewer" represents Mosaic's ability to support Telnet connections by invoking an external application. Telnet is a network service that enables users to login on other computers through a TCP/IP connection.

3) "Connect to News Server" represents Mosaic's ability to support Network News Transfer Protocol (NTTP)-based connections natively. These connections are generally used to obtain news from Usenet newsgroups.

   A Usenet newsgroup is a collection of messages on a single topic posted by Internet users. For instance, the newsgroup "comp.infosystems.www.users" contains messages discussing various aspects of Web-related applications and systems.

4) "Connect to Gopher Server" represents Mosaic's ability to support Gopher-based connections natively. Gopher is a distributed client-server database system that enables users to locate files and services on the Internet through an interactive menu interface.

5) "Connect to HTTP Server" represents Mosaic's ability to support HTTP-based connections natively. As noted earlier, HTTP connections are the primary means for retrieving hypermedia documents through the Web.

6) "Connect to FTP Server" represents Mosaic's ability to support FTP-based connections natively. As one of the oldest and most popular Internet services, File Transfer Protocol (FTP) enables users to download files from remote systems located anywhere on the Internet.

7) "Connect to Server via Gateway" represents Mosaic's ability to support various types of Internet connections by connecting to a gateway server. Two of the protocols supported in this way are WAIS and Archie. Wide-Area Information Server (WAIS) is a full-text search and retrieval service developed by a team of researchers at Thinking Machines. Archie is a client-server database program for finding and indexing files throughout the Internet.

8) "Send Request to Server" represents Mosaic's ability to request the transfer of a file located on a remote server.

9) "Receive Requested File" represents Mosaic's ability to retrieve a requested file using one of the Internet protocols or services discussed above.

10) "Retrieve File From Client Machine" represents Mosaic's ability to retrieve a requested file located on the client machine through a local file service.

11) "Forward Retrieved File" represents Mosaic's ability to forward a retrieved file to one of its internal document processing utilities or to an external viewer application.

The inputs represented in Graph M.1 are as follows:

1) "User Input: Local File Name" represents a user-specified name of a file stored on the client machine.

2) "User Input: URL" represents a user-specified Uniform Resource Locator of information stored anywhere on the Internet.

3) "URL of Requested Document" represents the Uniform Resource Locator of information stored anywhere on the Internet.

The intermediate outputs represented in Graph M.1 are as follows:

1) "Telnet Server Name" represents the Internet address of a Telnet server.

2) "Telnet Connection" represents a connection to a Telnet server.

3) "News Server Name" represents the Internet address of a Usenet News server.

4) "News Connection" represents a connection to a Usenet News server.

5) "Gopher Server Name" represents the Internet address of a Gopher server.

6) "HTTP Server Name" represents a connection to a Gopher server.

7) "FTP Server Name" represents the Internet address of an FTP server.

8) "FTP Connection" represents a connection to an FTP server.

9) "Unsupported Server Name" represents the Internet address of an unsupported server.

10) "Gateway Connection" represents a connection to an unsupported server via a gateway.

11) "Partial URL" represents the remainder of the URL string following the prefix and the server name. For instance, consider the URL string "http://www.example.com/Public/Documents/Help.html." In this case, the partial URL is the string "/Public/Documents/Help.html."

12) "Request Command" represents a command requesting a remote server to send a document to the client machine. For instance, in the case of an HTTP connection, the request command consists of the word GET, a space, and the partial URL of the document to be retrieved--terminated by the carriage return and line feed characters.

13) "Internet Connection" represents an Internet connection using one of the application protocols supported by Mosaic (e.g. HTTP).

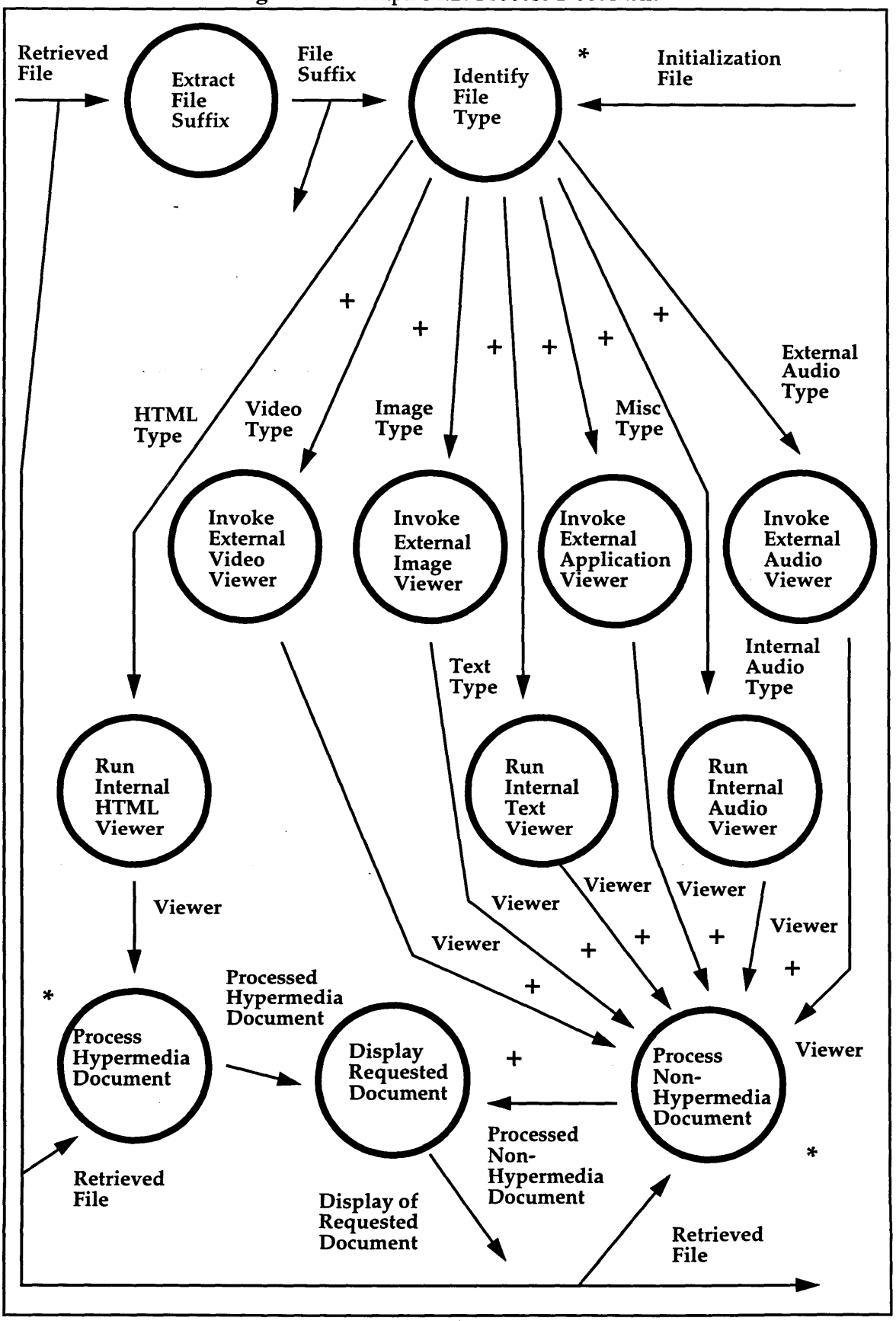
14) "Internet Document" represents a document retrieved from a server located somewhere on the Internet.

15) "Local File" represents a document retrieved from the client machine through a local file service.

The final output represented in Graph M.1 is as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

Figure 3.3: Graph M.2: Process Document



A secondary function of Mosaic is to process and display information retrieved from the Internet. Unlike other Internet tools such as FTP, Mosaic incorporates a sophisticated document processing capability, making it possible to "navigate" or "browse" the Internet. Mosaic provides native support for certain types of documents including HTML text, GIF images, and AU audio files. For documents that are not supported natively, Mosaic can be configured to invoke external viewers. For instance, if a link points to a video file in MPEG format, Mosaic can pass the video data to an external viewer that can recognize MPEG files. The viewer can then display the video data in a separate window next to the original Mosaic window.

Representing the function of processing a retrieved document, Graph M.2 consists of the following transform bubbles:

1) "Extract File Suffix" represents Mosaic's ability to extract the suffix corresponding to a retrieved file. Generally, a file name consists of two or more elements separated by periods. The last element is considered the suffix of a file.

2) "Identify File Type" represents Mosaic's ability to recognize various types of files based on their suffixes.

3) "Invoke External Video Viewer" represents Mosaic's ability to process video files by invoking an external viewer. A popular video viewer is MPEGPLAY.

4) "Invoke External Image Viewer" represents Mosaic's ability to process image files by invoking an external viewer. One of the more popular image viewers is XV, which enables users to view both GIF and JPEG images.

5) "Invoke External Audio Viewer" represents Mosaic's ability to process audio files by invoking an external viewer. A popular audio viewer is ShowAudio, which enables users to play back audio files in various formats.

6) "Invoke External Application Viewer" represents Mosaic's ability to process special types of files by invoking external viewers. Some of the more popular application-specific viewers include Ghostview for displaying Postscript documents and Adobe's Acrobat Reader for displaying PDF documents.

7) "Run Internal HTML Viewer" represents Mosaic's ability to process HTML documents natively.

8) "Process Hypermedia Document" represents Mosaic's ability to recognize and process hypermedia documents in HTML format.

9) "Run Internal Text Viewer" represents Mosaic's ability to process text files natively.

10) "Run Internal Audio Viewer" represents Mosaic's ability to process audio files in AU format natively.

11) "Process Non-Hypermedia Document" represents Mosaic's ability to process a wide range of non-hypermedia documents by invoking either an internal or external viewer.

12) "Display Requested Document" represents Mosaic's ability to display the requested document using a wide range of peripheral devices--including a monitor in the case of text, image, or video files; and a speaker in the case of audio files.

The inputs represented in Graph M.2 are as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

2) "Initialization File" represents a file specifying the configuration of Mosaic's document processing features.

The intermediate outputs represented in Graph M.2 are as follows:

1) "File Suffix" represents the last element of a file name, indicating the file type.

2) "HTML Type" represents a hypermedia document type in HTML format, indicated by the file suffix ".html."

3) "Video Type" represents a video document type. One of the more common video types is MPEG, represented by suffixes ".mpeg" and "mpg."

4) "Image Type" represents an image document type. Some of the more common image types include JPEG, represented by suffixes ".jpeg", ".jpe", and ".jpg"; GIF, represented by the suffix ".gif"; and TIFF, represented by the suffixes ".tiff" and ".tif."

5) "Text Type" represents a text document type. Text documents are usually indicated by the suffix ".txt."

6) "External Audio Type" represents an audio document type that is not supported natively by Mosaic. Some of the more common audio types include WAV, represented by the suffixes ".wave", ".wav", and ".WAV"; and MIDI, represented by the suffix ".mid."

7) "Internal Audio Type" represents an audio document type supported natively by Mosaic. Mosaic provides native support for AU audio files, indicated by the suffix ".au."

8) "Misc Type" represents miscellaneous document types that can be supported by invoking external viewers. Some of the more common types include Postscript, represented by the suffix ".ps," and PDF, represented by the suffix ".pdf."

9) "Viewer" represents one of the various external viewers supported by Mosaic.

10) "Processed Hypermedia Document" represents the processed version of a hypermedia document that is ready to be displayed.

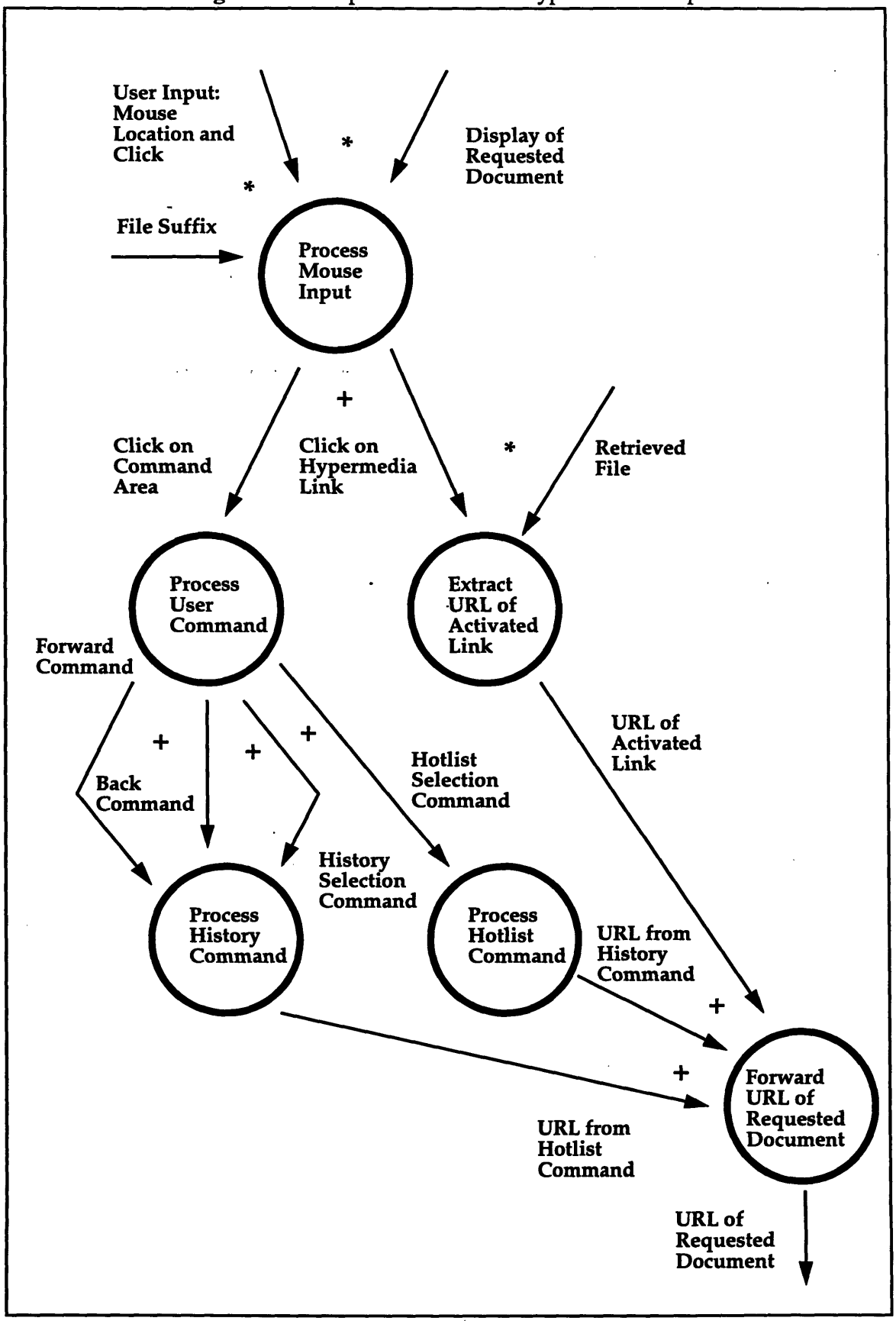
11) "Processed Non-Hypermedia Document" represents the processed version of a non-hypermedia document that is ready to be displayed.

The final outputs represented in Graph M.2 are as follows:

1) "File Suffix" represents the last element of a file name, indicating the file type.

2) "Display of Requested Document" represents a screen display of the requested document, in the case of text and image files, or a playback of the requested document, in the case of audio and video files. If an external viewer was invoked, the requested document is displayed in a separate window next to the original Mosaic window.

Figure 3.4: Graph M.3: Execute Hypermedia Jump



The main distinguishing feature of Mosaic and other Web browsers is the use of hypermedia links to locate and transfer information over the Internet. In fact, the World Wide Web is simply the subset of all the information on the Internet that can be accessed using hypermedia links (Merrin 2). Through Mosaic, users can traverse the Web by simply clicking on highlighted words or "hot spots" within the currently displayed document. Upon detecting such a click, Mosaic proceeds to retrieve the document indicated by the corresponding hypermedia link. The requested document can be located on the client machine itself or on a remote server thousands of miles away. To the user, there is no difference, except in the time required to retrieve it. Using the same point-and-click method, Mosaic users can also retrieve images, audio clips, or even full-motion video.

Representing the function of executing a hypermedia jump, Graph M.3 consists of the following transform bubbles:

1) "Process Mouse Input" represents Mosaic's ability to translate the location and click of a mouse into recognizable commands.

2) "Process User Command" represents Mosaic's ability to recognize and process mouse clicks on screen areas corresponding to user commands. For instance, the user can issue a "Forward" command by clicking on the right-arrow icon located at the top of the screen display.

3) "Extract URL of Activated Link" represents Mosaic's ability to extract the URL associated with an activated hypermedia link by referring to the retrieved HTML file.

4) "Process History Command" represents Mosaic's ability to keep track of the documents displayed during a single session. The user can view this list at any time and select an entry in order to retrieve a previous document. The user can also traverse this list in a backward or forward manner by clicking on the corresponding icons.

5) "Process Hotlist Command" represents Mosaic's ability to maintain a personalized list of favorite Web sites and documents. This list is usually stored on a hard disk.

6) "Forward URL of Requested Document" represents Mosaic's ability to forward the URL of the next document to be retrieved.

The inputs represented in Graph M.3 are as follows:

1) "User Input: Mouse Location and Click" represents the user-specified location and click of a mouse.

2) "Display of Requested Document" represents a screen display of the requested document, in the case of text and image files, or a playback of the requested document, in the case of audio and video files.

3) "File Suffix" represents the last element of a file name, indicating the file type.

4) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

The intermediate outputs represented in Graph M.3 are as follows:

1) "Click on Command Area" represents a mouse click on an icon or a menu entry corresponding to a user command. For instance, a click on the right-arrow icon located at the top of the screen display indicates a "Forward" command.

2) "Click on Hypermedia Link" represents a mouse click on a hypermedia link indicated by highlighted text or image.

3) "Back Command" represents a command to display documents retrieved earlier during the current session. Represented by a left-arrow icon, this command enables the user to traverse the history list in reverse order, from the most recent to the earliest documents.

4) "Forward Command" represents a command to display documents retrieved earlier during the current session. Represented by a right-arrow icon, this command enables users to traverse the history list in forward order, from the earliest to the most recent documents.

5) "History Selection Command" represents a command to select a previous document from the history list.

6) "Hotlist Selection Command" represents a command to select a document or Web site from a personalized hotlist.

7) "URL of Activated Link" represents the URL corresponding to an activated hypermedia link.

8) "URL from History Command" represents the URL corresponding to a history command.

9) "URL from Hotlist Command" represents the URL corresponding to a hotlist command.

The final output represented in Graph M.3 is as follows:

1) "URL of Requested Document" represents the URL corresponding to an activated link, a history command, or a hotlist command.

3.2.1.2 User Interface

As noted earlier, the ideal way to capture the user interface of most computer programs is to observe their screen displays and take snapshots. Unfortunately, observing the original screen display of Mosaic was not an option because a working copy of Mosaic Version 1.0 for X Window could not be obtained. Instead, a copy of Mosaic Version 2.6 for X Window was installed and tested. Accordingly, the following analysis of Mosaic's user interface was based on direct observations of Version 2.6 rather than Version 1.0. Although the two versions use slightly different interfaces, the existing literature suggests that the essential characteristics of Mosaic's user interface remain unchanged from Version 1.0 to Version 2.6.

Designed to work within a windows-based operating system, Mosaic incorporates a graphical user interface with integrated support for mouse operations. The interface consists of seven distinct areas within the main Mosaic window (NCSA, "NCSA Mosaic for the X Window System User Guide"):

1) Title field is used to display the title of the current document.

2) Menu bar is used to display program commands by organizing them into various functional menus. The available menus are File, Options, Navigate, Annotate, News, and Documents.

3) Bottom control panel is used to display many of the basic commands as buttons. Some of the commands displayed in the tool bar include Back, Forward, and Reload.

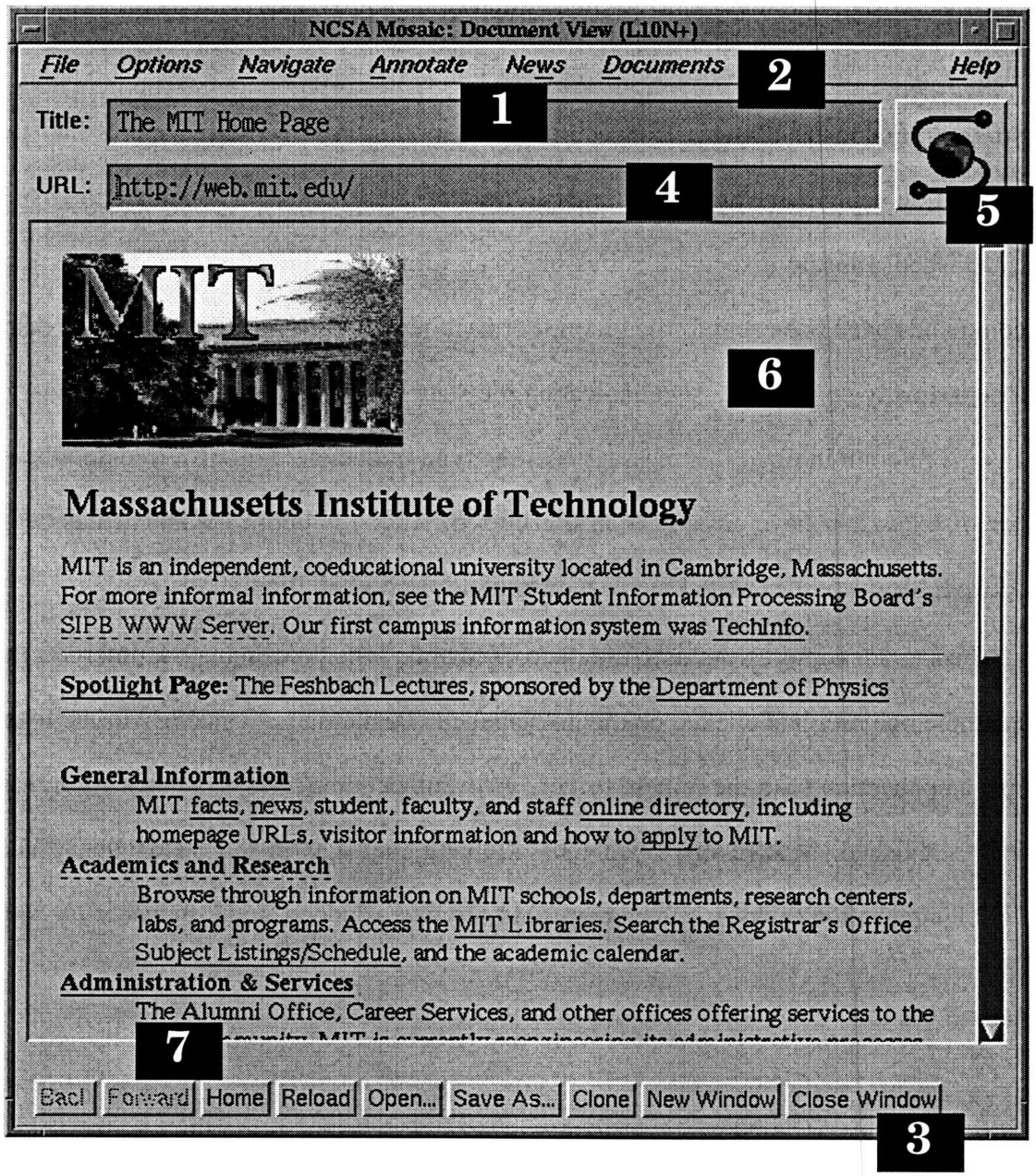
4) URL field is used to indicate the URL of the document currently being displayed.

5) NCSA Mosaic logo is used to indicate the status of a file transfer. The globe begins to spin when a transfer is initiated and continues to spin until the requested document has been displayed or an error message has been returned.

6) Document view area is used to display the text and inlined images of the current document. A scroll bar is provided to accommodate documents that are longer or wider than the available area.

7) Status line is used to display the status messages generated by Mosaic when downloading and displaying the requested document.

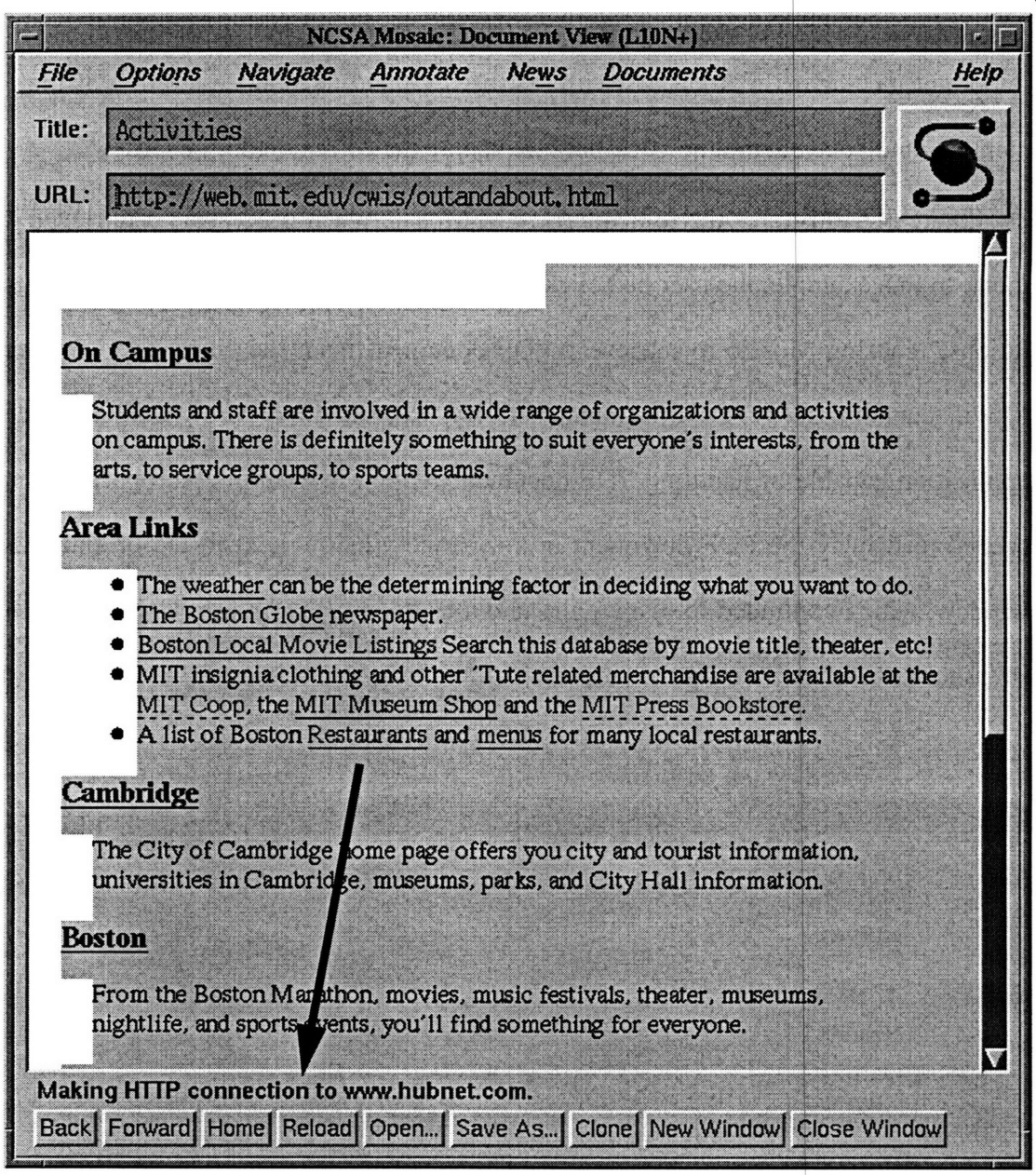
Figure 3.5: Snapshot of Mosaic's Main Window



When accessing the Internet, Mosaic relays various messages to the user through the NCSA Mosaic logo and the status line. As described earlier, Mosaic uses the globe-shaped logo to indicate an Internet session in progress: The globe continues to spin until all of the file transfers have been completed. In addition to the globe, Mosaic uses the status line to display specific and detailed messages regarding the intermediate status of the session in progress. For instance, when connecting to a remote server, Mosaic displays the type of connection being made and the Internet address of the server. After a connection has been established, Mosaic displays a series of status messages indicating that various request commands have been sent to the server. For each request command, Mosaic also displays a periodically updated message indicating the status of the corresponding file transfer. At any given point in time, this message lists the type of data being transferred and the number of bytes that have already been transferred. After the entire document has been retrieved, Mosaic issues a final message indicating the end of all file transfers.

Figure 3.6 captures the user interface of Mosaic when accessing the Internet. In this example, the user had clicked on the link labelled "Restaurants," causing Mosaic to establish a connection with the remote server "www.hubnet.com." As this connection was being established, Mosaic displayed the following message in the status line: "Making HTTP connection to www.hubnet.com." An arrow was added to the snapshot for illustrative effect.

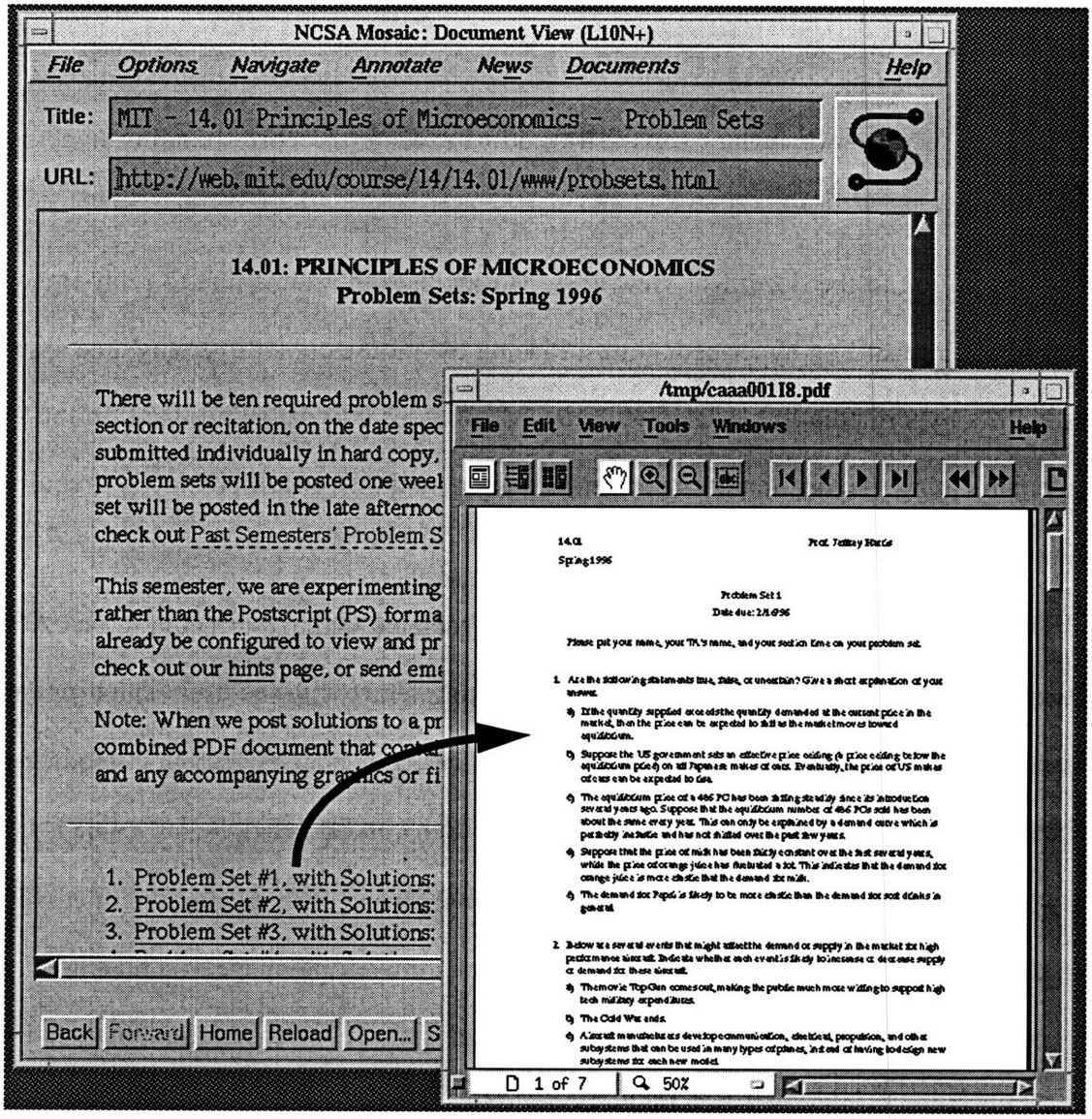
Figure 3.6: Snapshot of Mosaic's Interface When Accessing the Internet



When processing a retrieved document, Mosaic utilizes the following elements of its user interface: the document view area, the status bar, the title field, and the URL field. It uses the document view area to display documents that can be processed natively. Alternately, for documents requiring external viewers, Mosaic uses the status line to display important messages. In either case, it uses the title field to indicate the title of the document being displayed and the URL field to display the corresponding URL.

Figure 3.7 illustrates the user interface of Mosaic when processing a retrieved document. In this example, the user had clicked on the link labelled "Problem Set #1, with Solutions," causing Mosaic to retrieve a PDF document from the current server. Because Mosaic cannot process PDF documents natively, it invoked a copy of Acrobat Reader, an application capable of handling PDF documents. Upon invocation, Acrobat Reader proceeded to display the PDF document in a separate window next to the original Mosaic window, which continued to display the previous document. An arrow was added to the snapshot for illustrative effect.

Figure 3.7: Snapshot of Mosaic's Interface When Processing a Retrieved Document



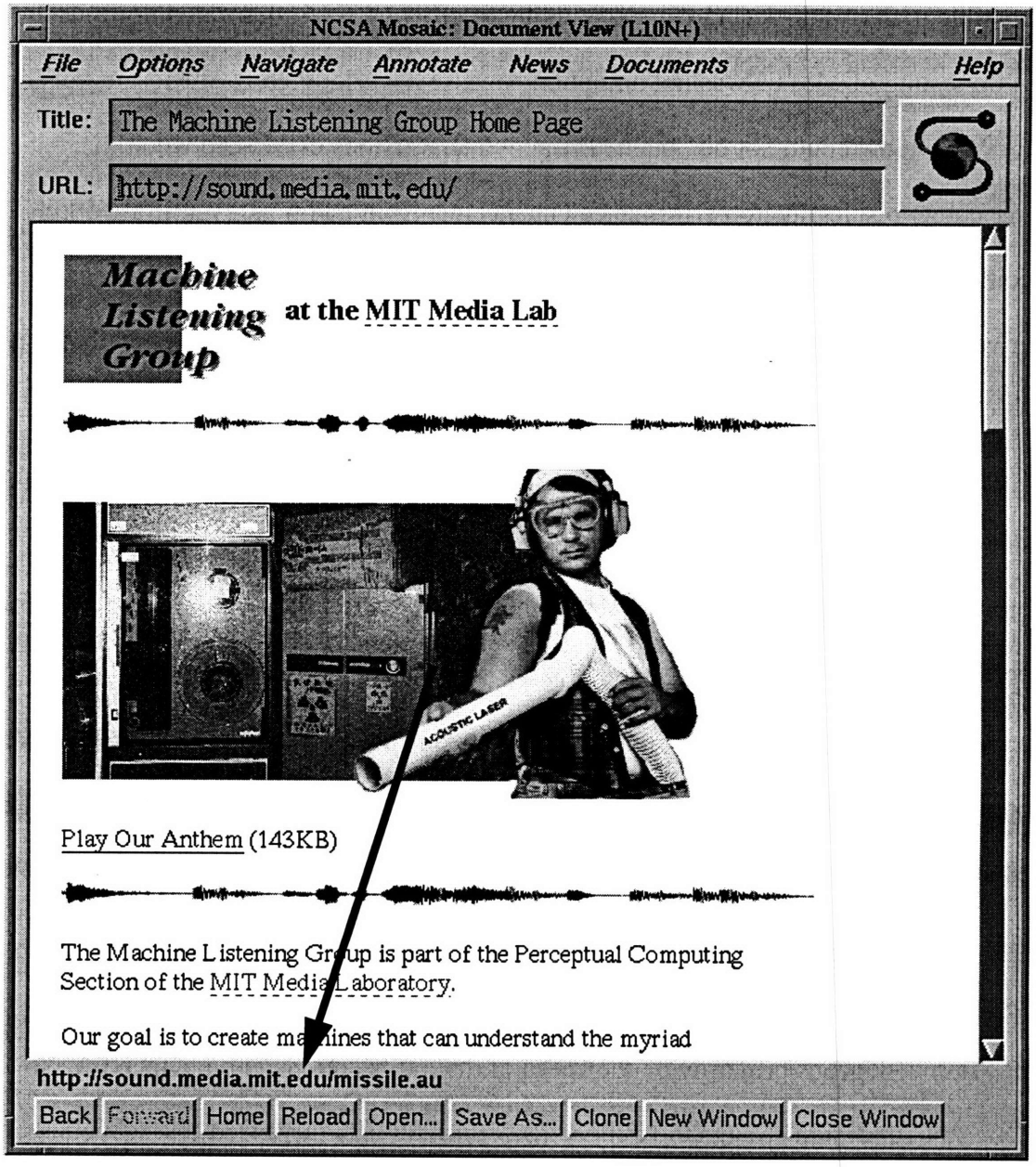
When executing a hypermedia jump, Mosaic utilizes the document view area and the status line. It uses the document view area to display hypermedia documents in HTML format as well as other documents supported natively by Mosaic. When displaying HTML documents, Mosaic underlines or highlights words corresponding to hypermedia links. In addition, it considers images associated with hypermedia links as "hot spots." Clicking on any of these words or images causes Mosaic to execute the corresponding hypermedia jump.

In addition to the document view area, Mosaic uses the status line to relay important information regarding hypermedia links. Placing the mouse cursor over any area associated with a hypermedia link causes Mosaic to display the corresponding URL in the status line, indicating the location of the linked document and the type of connection required to retrieve it.

For other types of commands beside those associated with hypermedia jumps, Mosaic uses the menu bar and the bottom control panel to interact with the user: The user can invoke various commands, such as history and hotlist features, by selecting a valid entry of the menu bar or by clicking on a button in the bottom control panel.

Figure 3.8 illustrates the user interface of Mosaic when executing a hypermedia jump. In this example, the user had placed the mouse cursor over the image of a man holding an audio device. By design, the image was associated with a hypermedia link to an audio file located on the current server. Clicking on any part of the image would cause Mosaic to play back a pre-recorded audio track. Detecting the placement of the cursor over a hot spot, Mosaic proceeded to display the corresponding URL in the status line--in this case, "http://sound.media.mit.edu/missile.au." An arrow was added to the snapshot for illustrative effect.

Figure 3.8: Snapshot of Mosaic's Interface When Executing a Hypermedia Jump



3.2.1.3 Performance

In most cases, the ideal way to capture a computer program's performance is through program testing. By comparing the testing results to those of competing programs, it is possible to derive an accurate understanding of program performance. Unfortunately, program testing of Mosaic was not an option because a working copy of Mosaic Version 1.0 for X Window could not be obtained. Using later versions of Mosaic would have yielded inaccurate results due to performance enhancements; in addition, even if a working copy could be obtained, it would have been difficult to evaluate the testing results because earlier versions of other Web browsers, including those of Navigator, were generally not available.

Fortunately, there are numerous articles and texts comparing the performance of Mosaic Version 1.0 for X Window, at the time of its introduction, with that of existing Web browsers. These articles and texts form the basis for the following analysis of Mosaic's performance.

According to the existing literature, the overall performance of Mosaic was average compared to other Web interfaces (Vaughan-Nichols 193). In fact, all of the Web interfaces at the time exhibited similar levels of performance in terms of accessing the Internet and processing hypermedia links.

When it comes to accessing the Internet, the speed of any Web interface, including that of Mosaic, depends primarily on the speed of the Internet connection, which is beyond the scope of its design (Vaughan-Nichols 182). In addition, the speed also depends on the workload of the current server, which may vary from time to time. Unfortunately, Mosaic does not incorporate any enhancements to overcome these difficulties--resulting in average performance (Ernst 20).

Similarly, when it comes to displaying and processing hypermedia links, there were no discernible differences in performance between Mosaic and existing Web interfaces. Most of the Web interfaces, including Mosaic, employ similar mechanisms for handling hypermedia links. They use either an underlining or a highlighting scheme for locating links and a point-and-click mechanism for activating them.

On the other hand, when it comes to processing and displaying documents, Mosaic does include a number of features designed to enhance performance. First, Mosaic supports image caching within and across documents (NCSA, "Help On Version 0.10"). That is, Mosaic keeps copies of previously displayed images in memory for future reference. If the requested document contains an image that has been cached earlier, Mosaic simply redisplays the cached image instead of downloading another copy from the server, resulting in significant performance gains. Second, Mosaic also supports document caching (Vaughan-Nichols 254-255). As with images, Mosaic keeps copies of previous documents in memory for future access. If the requested document has already been displayed in the current session, as in the case of a Back command, Mosaic simply redisplays the document rather than downloading another copy from the server.

Unfortunately, both of these enhancements matter only when the requested document contains cached images or when it has already been cached in its entirety. More importantly, the potential improvement in performance offered by the above techniques were offset by two of Mosaic's drawbacks. First, Mosaic displays graphics more slowly than other Web interfaces (Vaughan-Nichols 183). As a result, Mosaic incurs significant delays when displaying even cached images and documents. Second, Mosaic attempts to download all of the images first then the text before displaying any part of the document (Vaughan-Nichols 182). Hence, it must retrieve the entire document before displaying it to the user, resulting in costly delays. Taking these drawbacks into account, Mosaic failed to provide

significant improvements in performance, despite incorporating image and document caching.

3.2.2 State of the Art Analysis

In analyzing the state of the art prior to the introduction of Mosaic, the following articles and texts were consulted: Inside the World Wide Web, "World-Wide Web," and "NCSA Mosaic and the World Wide Web: Global Hypermedia Protocols for the Internet." In addition, a search for relevant patents was conducted using LEXIS, a legal database service. Finally, various trade journals and press releases were consulted in order to assess the state of the Internet applications market prior to Mosaic.

As with the rest of the case study, the state of the art analysis was based on Mosaic Version 1.0 for X Window, which was first introduced in April of 1993. The results of the analysis are presented in this subsection.

3.2.2.1 Internet

In the late 1960's, a research division of the U.S. Department of Defense called "Advanced Research Projects Agency" (ARPA) began work on an experimental networking scheme for interconnecting geographically dispersed computers (Vaughan-Nichols 33). Unlike most existing schemes, this scheme promised to be very robust and easily scalable.

At the heart of ARPA's networking scheme was the theory of packet-switching. Unlike traditional networks, a packet-switching network transmits information by dividing the data into small, autonomous packets. Each data packet, in turn, contains a header consisting of an identification number and information regarding the packet's origin and destination. The packets are then routed to their destination through a matrix of possible network connections rather than through a pre-determined path. Under this scheme, the actual path taken by a particular packet is determined dynamically based on network traffic and may

vary from one packet to another. Upon arriving at their destination, the packets are finally reassembled into their original form based on header information.

After years of research and development, the first network to incorporate packet-switching technology was implemented in the fall of 1969; it was named the ARPAnet in recognition of the pioneering research conducted at the Department of Defense (Vaughan-Nichols 34). The network interconnected four Honeywell 516 minicomputers located on university campuses along the west coast and utilized the now defunct Network Control Protocol (NCP). In time, the NCP protocol proved to be insufficiently robust to handle all of ARPA's diverse networks.

Recognizing the need to interconnect networks of different design and implementation, computer scientists began work on a new protocol that could accommodate heterogeneous networks. The result of this research was the Transmission Control Protocol/ Internet Protocol (TCP/IP). Unlike its predecessor NCP, the TCP/IP protocol was robust enough to accommodate different types of networks and computers while maintaining all the advantages of packet-switching. On January 1, 1983, TCP/IP finally replaced NCP as the underlying protocol of the ARPAnet and the Defense Data Network (DDN), giving birth to the Internet (Vaughan-Nichols 35).

Throughout the 1980s, the Internet experienced steady growth world-wide, even though it remained primarily a tool for the scientific and research community (Vaughan-Nichols 36-37). In fact, using the Internet for commercial purposes was expressly prohibited during this time. Then in 1991, a number of major corporations began collaborating on a so-called "Commercial Internet Exchange" (CIX) to enable and promote commercial transactions on the Internet (Vaughan-Nichols 37). Combined with the emergence of other backbones, such as the National Science Foundation Net (NSFnet) and the Advanced Network & Services (ANS), the introduction of CIX marked a turning point in the evolution

of the Internet. For the first time in its history, the Internet began to make significant headway into the business community as well as the general public. By 1994--the year of Mosaic's remarkable growth in popularity, the Internet had become an international phenomenon, with 2 million host computers and an estimated 20 million users (Schatz and Hardin 901).

Although the size and speed of the Internet had improved greatly in the years prior to Mosaic, locating and retrieving information on the Internet remained a difficult task (Ernst 17). Existing Internet tools like FTP, Gopher, and WAIS required extensive knowledge of the numerous addressing schemes, protocols, and data formats underlying the Internet as well as familiarity with the tools themselves. In addition, these tools often required users to know where the desired piece of information was located. More importantly, none of these tools offered a method for relating or linking one piece of information to another. This limitation meant that obtaining information on a single topic would often require numerous and unrelated searches involving any number of different tools and interfaces. As a result, many users, especially novices, were unable to tap the full potential of the Internet (Ernst 17).

3.2.2.2 Hypertext and Hypermedia

As a separate and parallel development to the Internet, hypertext and hypermedia technologies finally began to gain recognition and prominence in the early 1990s. The roots of hypertext and hypermedia can be traced back to 1945 with the publication of "As We May Think." In the article, Vannevar Bush proposed the idea of a machine which would enable users to store and retrieve documents using random associations (Nelson 1/49-1/52). Bush named his theoretical machine MEMEX for "memory extension" (Nelson 1/50). Envi-

sioned as a purely mechanical machine, MEMEX would allow users to retrieve books, records, and communications "with exceeding speed and flexibility" (Nelson 1/50).

The main distinguishing feature of MEMEX is its use of an associative indexing scheme. Unlike conventional schemes, associative indexing enables the user to tie or link any two items together by constructing a permanent pointer. Thereafter, when one of the linked items is on display, the user can instantly retrieve the other by activating the appropriate pointer. Using these pointers to link previously unconnected items, the user can form a trail of information that can be accessed repeatedly in the future.

Because Bush wrote his article in 1945, he envisioned MEMEX as a mechanical system composed of a desk, a keyboard, a set of translucent screens, and series of buttons and levers. Due to the complexities involved in implementing such a system, the idea of MEMEX was largely forgotten until the early 1960s, when Ted Nelson began work on the first hypertext system.

The term "hypertext" refers to the notion of non-sequential writing--that is, "text that branches and allows choices to the reader" (Nelson 0/2). As proposed by Nelson, hypertext systems would enable users to compose documents consisting of many separate paragraphs, each with many branching choices. Depending on the branches taken by the reader, the same document could be accessed in any number of ways.

In his earlier proposals, Nelson designed his hypertext systems using zippered lists. Similar to the associative indexing scheme proposed by Bush, zippered lists are lists which index each other in different sequences. The system that ultimately emerged from Nelson's work is essentially an electronic version of Bush's MEMEX. Instead of physical buttons and levers, however, Nelson's hypertext system would be implemented using computer technology.

Throughout the 1970s and 1980s, Nelson embarked on a series of research projects with the goal of implementing his design for a computer-based hypertext system. Collectively known as "Project Xanadu," these projects led to a number of pioneering observations and insights, yet they failed to yield a commercially successful product (Vaughan-Nichols 16). The remarkable history of Project Xanadu as well as Nelson's views on hypertext and hypermedia in general is described in Literary Machines, written by Nelson himself.

Following the publication of Literary Machines in 1981, other scientists and engineers beside Nelson began developing hypertext and hypermedia systems. Some of these systems were academic experiments, and others were commercial products. Interestingly, most of these systems utilized the notion of links rather than zippered lists. A link is a user-defined connection between two documents and is usually represented as underlined or highlighted text. By activating a link, the user can "jump" instantaneously from one document to another. None of these systems, however, enabled users to retrieve hypertext or hypermedia documents across networks.

The first commercially successful hypertext system was Hypercard by Apple (Vaughan-Nichols 17). Introduced in 1987, Hypercard enabled users to construct notecards that can be retrieved through hypertext links. By using interconnected notecards to organize diverse pieces of information, Hypercard illustrated the power and simplicity of hypertext systems. Following the introduction of Hypercard, a number of CD-ROM applications began to incorporate hypertext and hypermedia technologies. For instance, in the New Grolier MultiMedia Encyclopedia, a popular CD-ROM application, the user can jump from an article on the Apollo 11 moon landing to an actual video of the landing by simply clicking on a hypermedia link (Vaughan-Nichols 17).

On a separate note, a search of the patent database for this time period yielded 37 U.S. patents related to hypertext or hypermedia technology. A list of these patents is included in Appendix A.1. Interestingly, none of these patents discloses a method or apparatus for retrieving hypertext or hypermedia documents across networks.

3.2.2.3 World Wide Web

In 1989, Tim Berners-Lee, a computer programmer with the European Particle Physics Laboratory (CERN) in Geneva, Switzerland, began developing a system that would enable researchers to work on collaborative projects in real time (Vaughan-Nichols 20). Perhaps the greatest obstacle facing Berners-Lee was the physical separation that often existed between the researchers and the information that they needed as well as the separation that existed between the researchers themselves (Vaughan-Nichols 20). For instance, a physicist in Geneva, Switzerland may need to obtain the latest set of data from a nuclear research facility in New Mexico, USA and forward her comments on the data to a colleague in Oxford, England--all in real time.

By this time, the Internet did provide a number of tools for information discovery and retrieval; however, all of these tools utilized different user interfaces and required an in-depth understanding of the Internet. Looking for a more flexible and user-friendly solution, Berners-Lee brought together two well-known and proven ideas: networked information and hypertext (Schatz and Hardin 896). The result was the World Wide Web or the "Web," for short. Berners-Lee formally presented his vision for the Web in a proposal submitted to CERN in October of 1990 (Schatz and Hardin 896).

Originally, Berners-Lee used the term "World Wide Web" to denote a set of protocols that would incorporate the following (Berners-Lee et al. 76):

1) The idea of a "boundless information world in which all items have a reference by which they can be retrieved"

2) A client-server architecture, in which information is requested through a client machine and supplied by a server machine

3) A universal naming and addressing system based on Universal Resource Locators (URLs)

4) A Hypertext Markup Language (HTML) for placing hypertext links within documents

5) A Hypertext Transfer Protocol (HTTP) for transferring information with the efficiency necessary for executing hypertext links

Over time, however, the term "World Wide Web" has come to represent the entire body of data on the Internet that can be accessed by using all or some of the above ideas (Berners Lee et al. 76).

In order to access the Web, the user must run on her machine a "client program," more commonly known as a "Web browser." As an interface to the World Wide Web, a Web browser must provide support for the set of protocols described above, including native support for HTML and HTTP.

The initial prototype developed at CERN was a hypertext browser/editor that utilizes a "what-you-see-is-what-you-get" (WYSIWYG) user interface and ran on NeXTStep workstations. The prototype displays hypertext links as underlined text and supports point-and-click operations using a mouse. In addition, it provides transparent access to servers of all types through a common interface. For non-NeXTStep platforms that do not support a WYSIWYG user interface, CERN also developed and distributed a line mode browser. This browser utilizes a purely textual user interface and presented hypertext links as numbered choices in a menu.

None of the browsers from CERN, however, provided native support for hypermedia documents. Hypermedia documents differ from regular hypertext documents in the fol-

lowing ways (Merrin 151): (1) They contain both textual and nontextual elements; and (2) they allow links to use nontextual elements as anchors. For example, consider a hypothetical hypermedia document on American history. Imagine that this document contains a brief history of the United States and a graphical map of the fifty states. By clicking on the portion of the map corresponding to any of the fifty states (e.g. Texas), the user can jump to another document providing further information on the history of the selected state. In this case, the map acts as an anchor for the hypermedia link.

3.2.3 Points of Novelty Embodied in Mosaic

Given the state of the art prior to its introduction, Mosaic Version 1.0 for X Window embodied the following points of novelty:

1) Novelty of functionality in enabling retrieval and display of hypermedia documents across the Internet

   Mosaic was the first Web browser to support hypermedia documents. With Mosaic, users could retrieve and view multimedia documents stored anywhere on the Internet with a single click of a mouse.

2) Novelty of user interface in displaying documents retrieved from the Internet

   Mosaic was the first Web browser to incorporate a graphical user interface that was both attractive and easy-to-use. In particular, Mosaic's interface for displaying requested documents was novel; according to most users, it had a unique "look-and-feel."

3) Novelty of functionality in supporting hypermedia links across the Internet

   Mosaic was the first Web browser to support links between multimedia objects across the Internet. Unlike its predecessors, Mosaic enabled users to link text, graphics, sound, and even video files--regardless of their physical location. With Mosaic, any object that can be accessed using the Internet can be linked to any other object.

In addition to the above written descriptions, each point of novelty is also depicted using modified versions of the data-flow graphs generated earlier: Graphs M, M.1, M.2, and

M.3. In these graphs, novelty of functionality is represented by dashed lines, and novelty of user interface and novelty of performance are represented by different patterns of shading.

In Graph M, each point of novelty is initially represented as a high-level combination of functionality, user interface, or performance. For instance, the first point of novelty is represented by a dashed rectangle labelled "1" enclosing the transform bubbles "Access the Internet" and "Process Document." This notation indicates the following: (1) The first point of novelty involves a functional combination of Internet access and document processing; and 2) low-level Graphs M.1 and Graphs M.2 should be consulted for additional information.

Each point of novelty is then described in more concrete terms using the appropriate low-level graphs. For instance, the first point of novelty is further depicted in Graphs M.1 and M.2. Graph M.1 includes a dashed rectangle labelled "1" enclosing the transform bubbles "Receive Retrieved File" and "Forward Retrieved File." This notation represents Mosaic's ability to receive data packets from the Internet and assemble them into files. Similarly, Graph M.2 includes a dashed rectangle labelled "1" enclosing the transform bubbles "Process Hypermedia Document" and "Display Requested Document." This notation represents Mosaic's ability to display hypermedia documents in HTML format natively. Taken collectively, these graphical notations indicate that the first point of novelty embodied in Mosaic is a novel combination of the ability to retrieve files from the Internet and the ability to display hypermedia documents.

The remaining points of novelty are represented using a similar set of notations. It is important to note, however, that these notations are intended to facilitate analysis, not to replace written descriptions. Ideally, each point of novelty should be described using both graphical notations and written descriptions.

Figure 3.9: Notations Used in the Graphs of Novelty

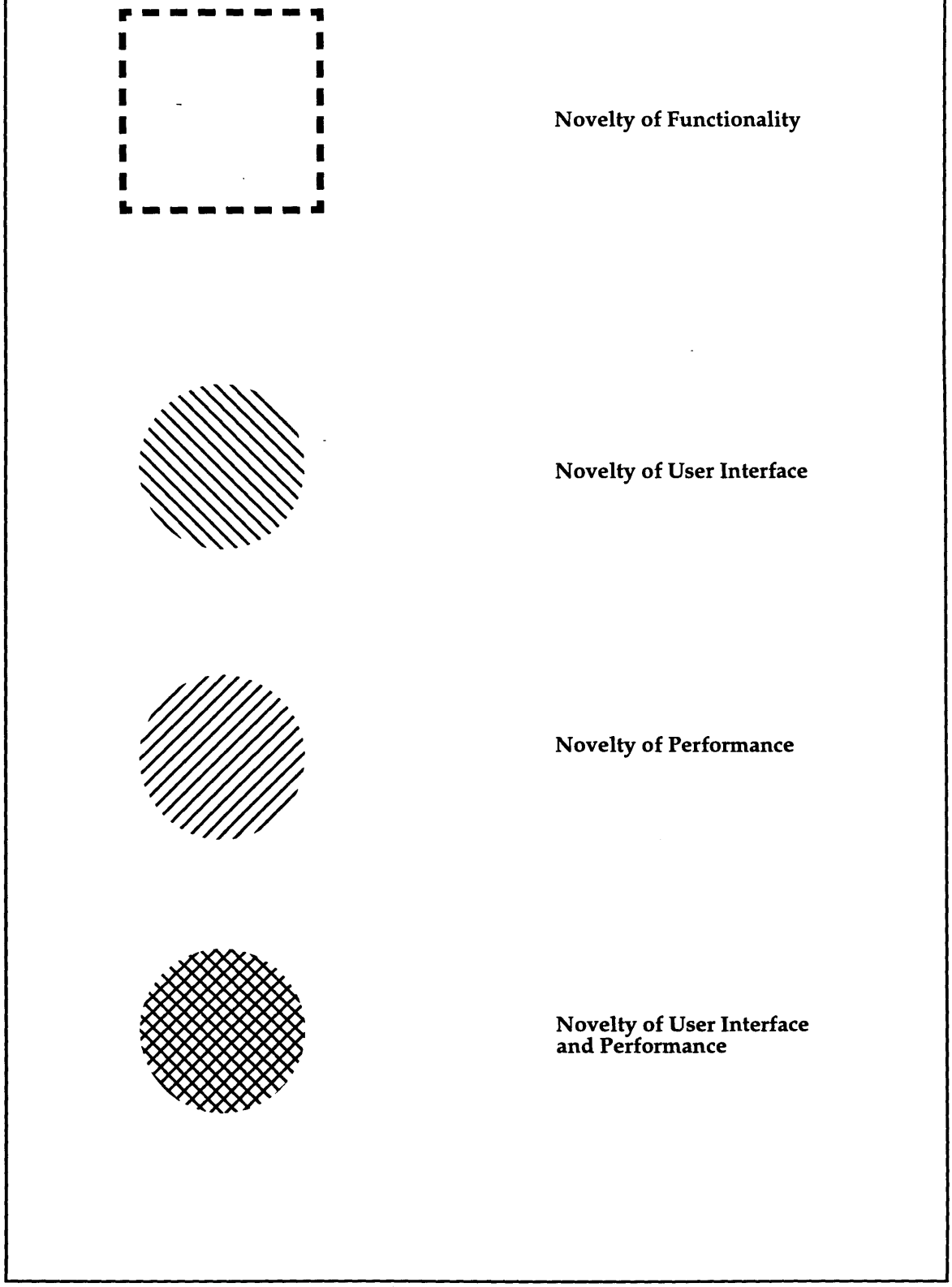


Figure 3.10: Graph M (Modified): Overview of Mosaic's Functionality

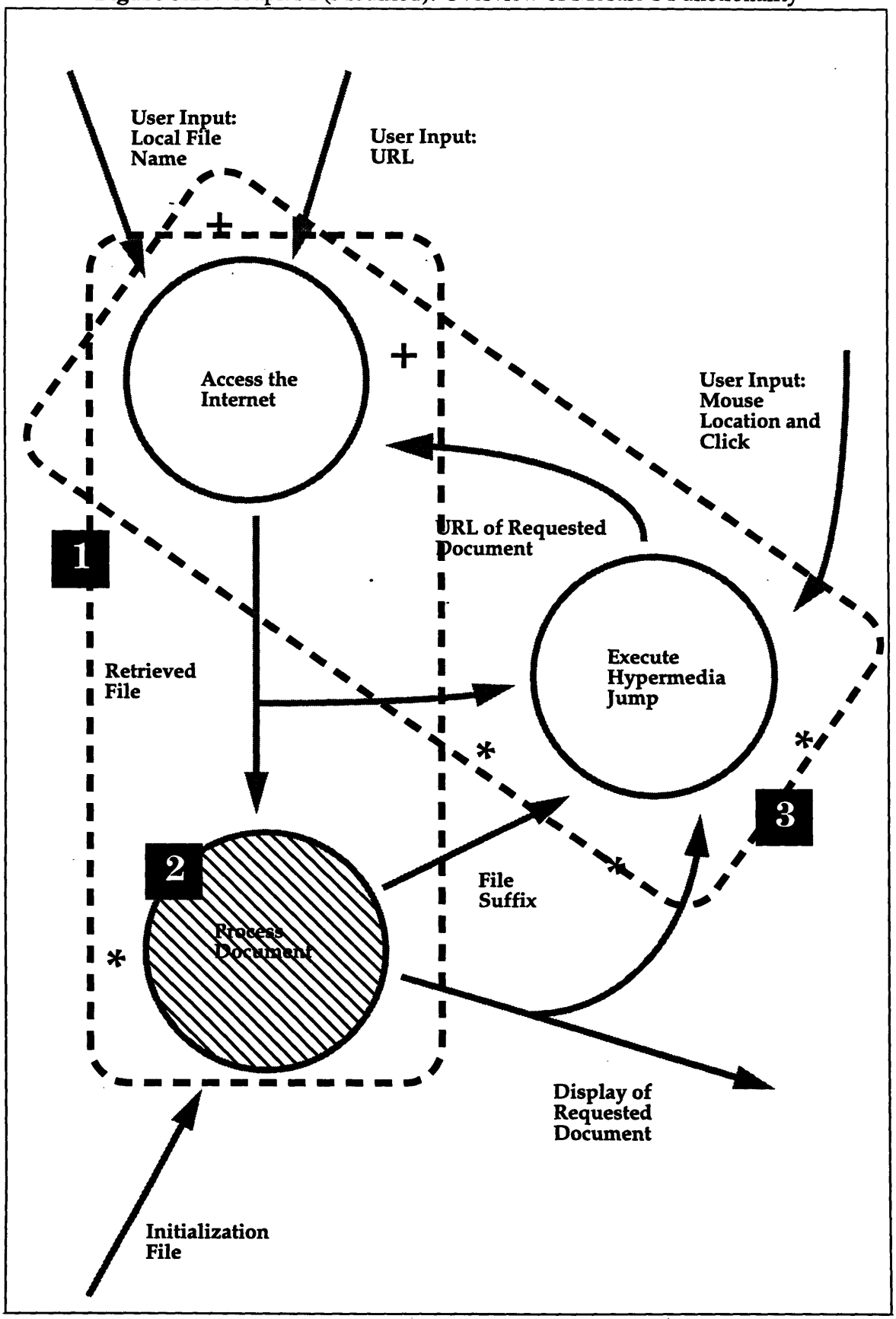


Figure 3.11: Graph M.1 (Modified): Access the Internet

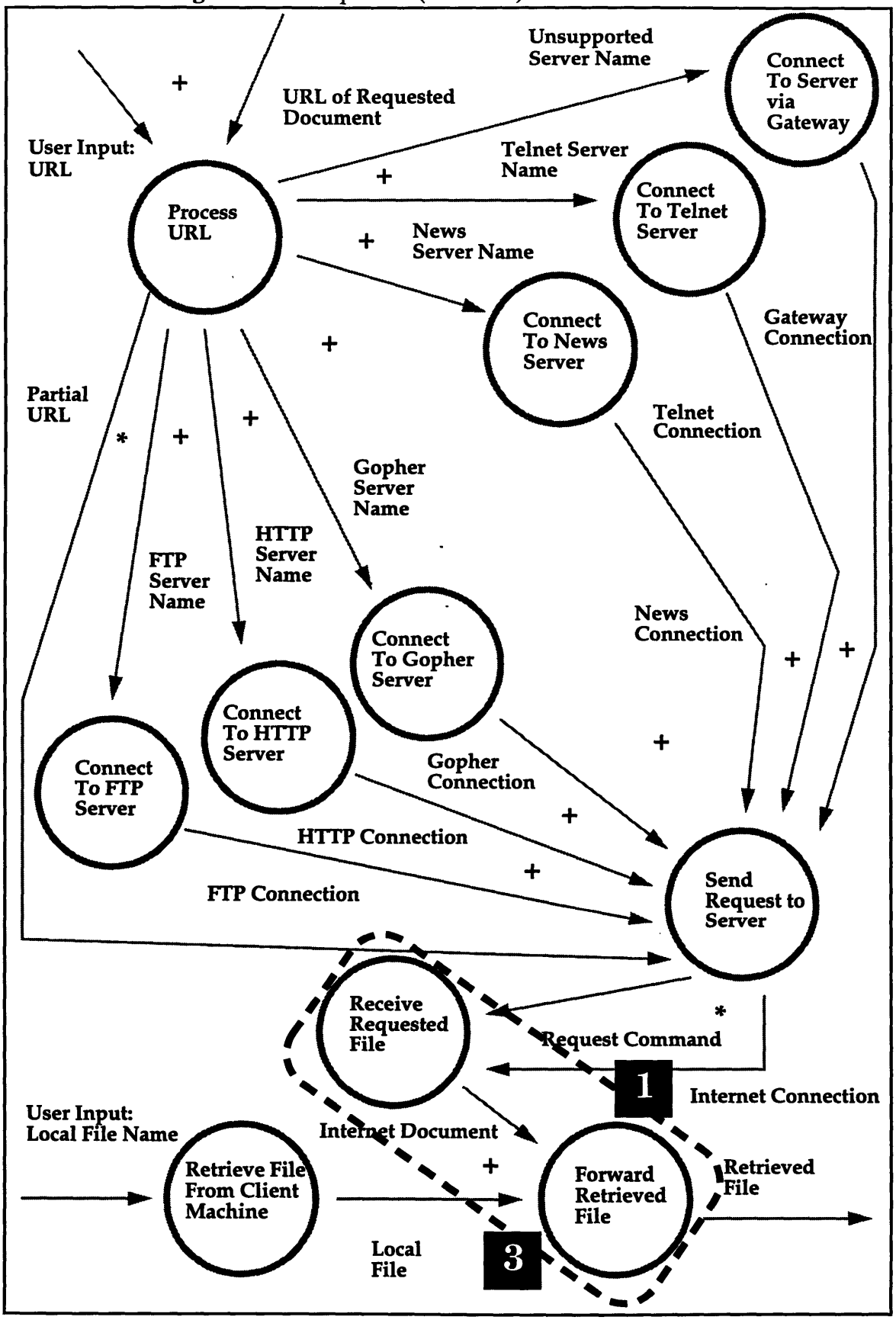


Figure 3.12: Graph M.2 (Modified): Process Document

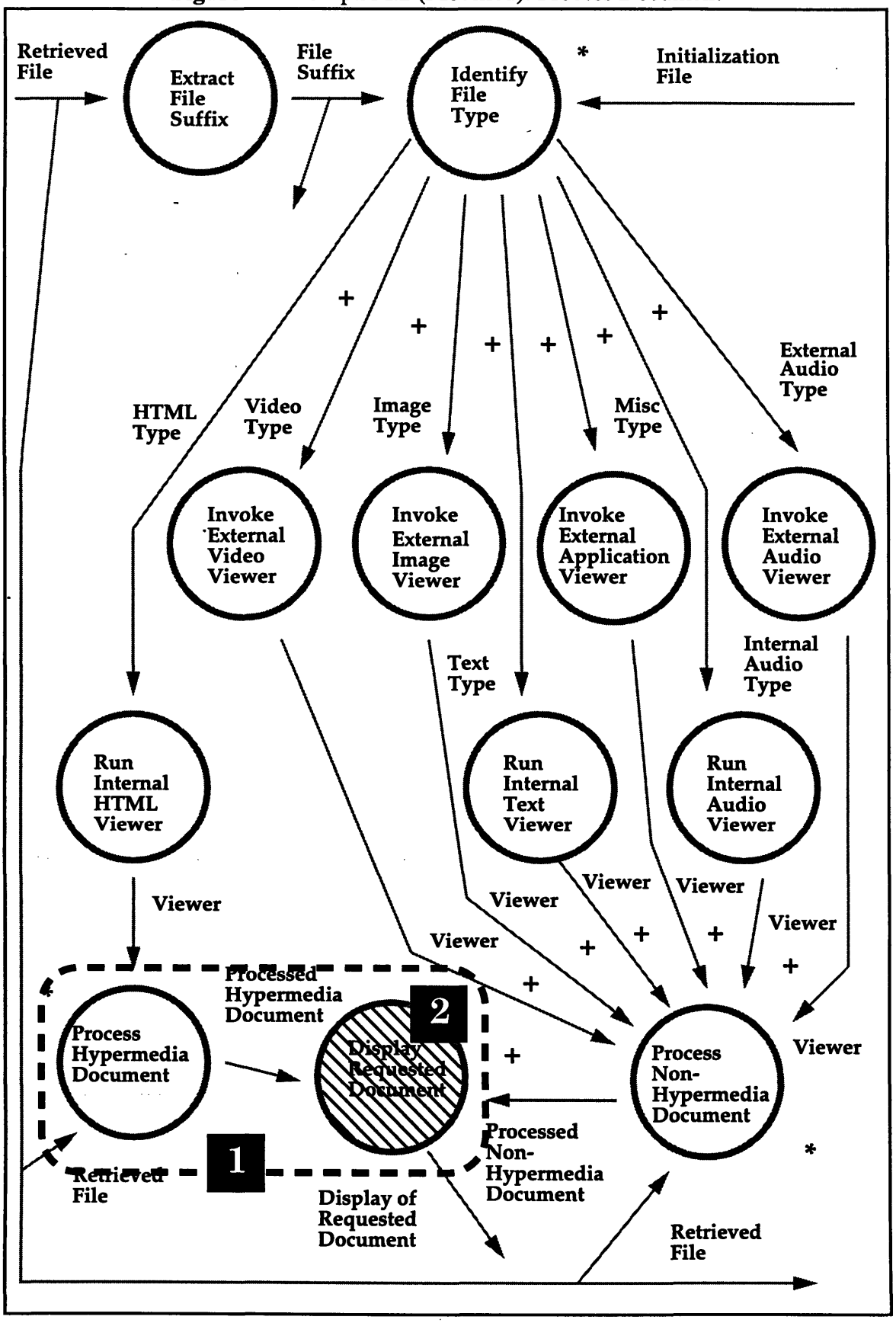
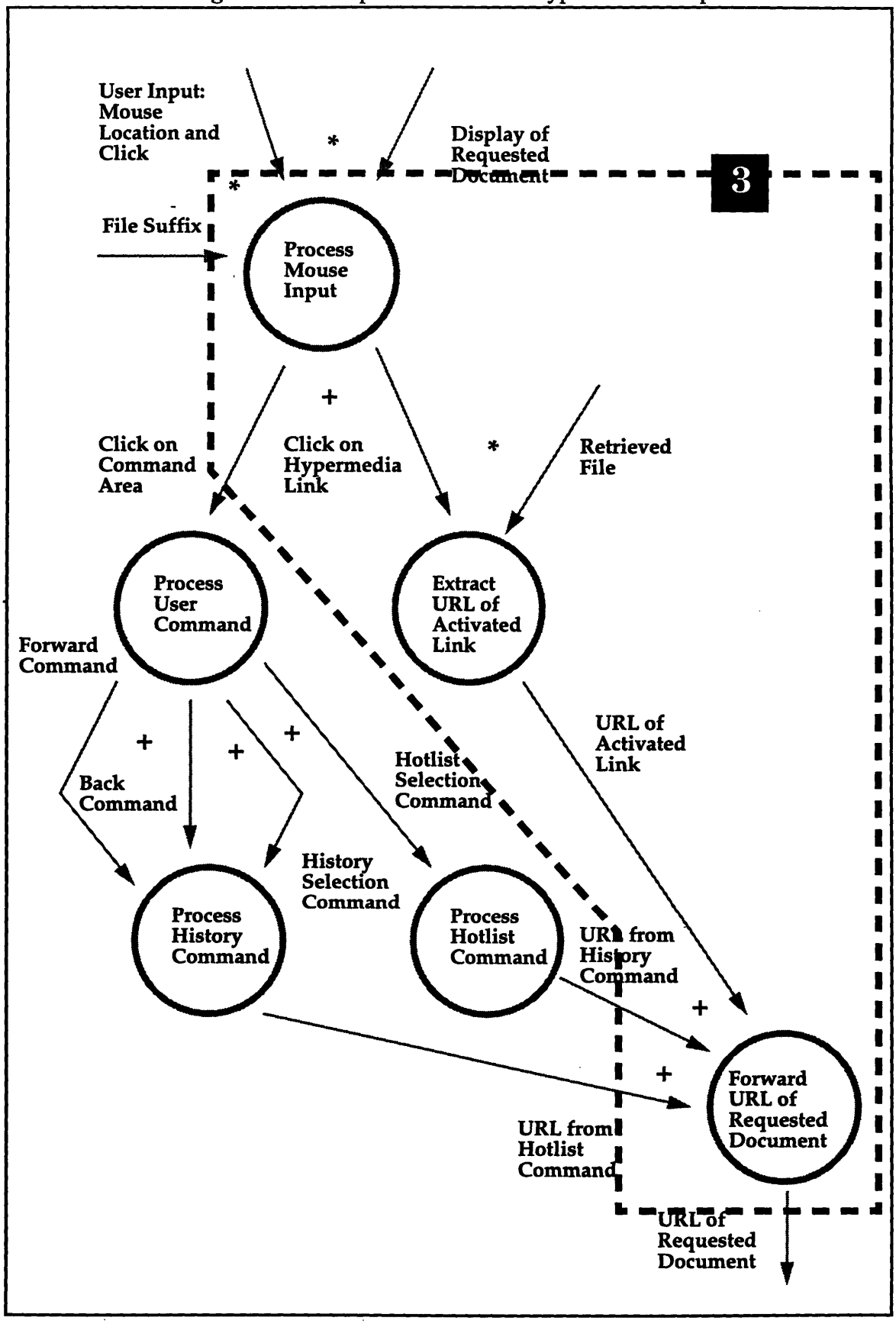


Figure 3.13: Graph M.3: Execute Hypermedia Jump



3.2.4 Commercial Significance Analysis

As the last stage of the analysis, each point of novelty embodied in Mosaic was analyzed for its commercial significance. In the process, the following articles and texts were consulted: Inside the World Wide Web and "NCSA Mosaic and the World Wide Web: Global Hypermedia Protocols for the Internet."

Since its introduction in January of 1993, Mosaic has experienced phenomenal growth in popularity and use. According to some estimates, there were more than one million copies of Mosaic by spring of 1994 alone (Schatz and Hardin 897). During the same time period, the Web itself grew by three orders of magnitude (Schatz and Hardin 897).

Most experts have attributed Mosaic's extraordinary success to a number of factors including the growth of the Internet itself and the availability of Mosaic as freeware (Schatz and Hardin 901). The most important factor, however, may be the various points of novelty embodied in Mosaic. Schatz and Hardin writes: "The user interface that WWW promotes is extremely simple: display a document with embedded links to other documents that can be fetched by pointing and clicking. Easy access to pictorial material within documents was finally feasible, both from a multimedia display and from a network speed standpoint... Mosaic/WWW had a combination of availability and features and a timing of access and information that put them 'over the top' into the feedback loop that is essential for successful propagation of network information systems" (901).

Based on these observations, it is safe to conclude that all three points of novelty embodied in Mosaic contributed directly to the increased use of Mosaic as well as the increased growth of the Web. Thus, all of them satisfy the standard of commercial significance, as defined in Chapter 2.

3.2.5 Behavioral Innovations Embodied in Mosaic

In summary, Mosaic Version 1.0 for X Window embodied the following behavioral innovations:

1) Innovation of functionality in enabling retrieval and display of hypermedia documents across the Internet

2) Innovation of user interface in displaying documents retrieved from the Internet

3) Innovation of functionality in supporting hypermedia links across the Internet

All of the above innovations represent an advance in the state of the art that produces a novel and useful result of commercial significance.

3.3 Netscape Navigator Web Browser

In December of 1994, Netscape Communications Corporation introduced Version 1.0 of its Netscape Navigator for the Apple Macintosh, X Window, and Microsoft Windows platforms (Netscape Communications, Corp., "Netscape Press Release 12/15/94"). Like its predecessor Mosaic, Netscape Navigator--"Navigator," for short--is a dedicated Web browser with native HTML and HTTP support. With the rising wave of public interest in the Internet and the World Wide Web, Navigator quickly replaced Mosaic as the most popular Web browser. As noted earlier in the Background section, Navigator currently owns over 80 percent of the market share and promises to continue its dominance well into the future.

The phenomenal success of Navigator can be attributed to several factors: First, Navigator was created by many of Mosaic's original developers including Mark Andreeson, the person most responsible for initiating the Mosaic project at NCSA (Ernst 20). Consequently, Navigator looks and feels remarkably like Mosaic, allowing Netscape Communications to build on the existing base of Mosaic users (Vaughan-Nichols 302). Second, unlike its predecessor Mosaic, Navigator was designed for a commercial rather than an ac-

ademic user base (Netscape Communications Corp., "Netscape's Mission and Milestones"). Consequently, Navigator boasts a number of significant improvements in functionality and performance. On the whole, these improvement make Navigator a much more robust and efficient program than Mosaic. Finally, in developing Navigator, Netscape Communications was aiming to make wide-spread commercialization of the Internet not only possible, but inevitable (Netscape Communications Corp., "Netscape's Mission and Milestones"). In pursuing this goal, Netscape Communications has formed strategic alliances with a number of highly visible companies, further promoting the popularity of Navigator (Netscape Communications Corp., "Netscape's Mission and Milestones"). Driven by all of the above factors, Navigator has emerged as the Web browser of choice for both business and home users.

3.3.1 Behavioral Analysis

According to Steven Vaughan-Nichols et al. in Inside the World Wide Web, "Netscape Communications' Netscape Navigator (Mac, Unix, Windows, Windows 95) is without question, the leading Web browser today. It's faster than the other browsers, loaded with features, and available in versions for more operating systems than any other browser. There's little to dislike and a great deal to love about Netscape" (194).

The above comments suggest that a behavioral analysis of Navigator should focus on the performance of Navigator as well as it features. Furthermore, the analysis should take into consideration the unique relationship between Navigator's program behavior and that of its predecessor Mosaic.

As with Mosaic, the program behavior of Navigator was analyzed along the dimensions of functionality, user interface, and performance. The analysis was based entirely on

the proposed scheme for identifying behavioral innovations in software. The results are presented in this subsection.

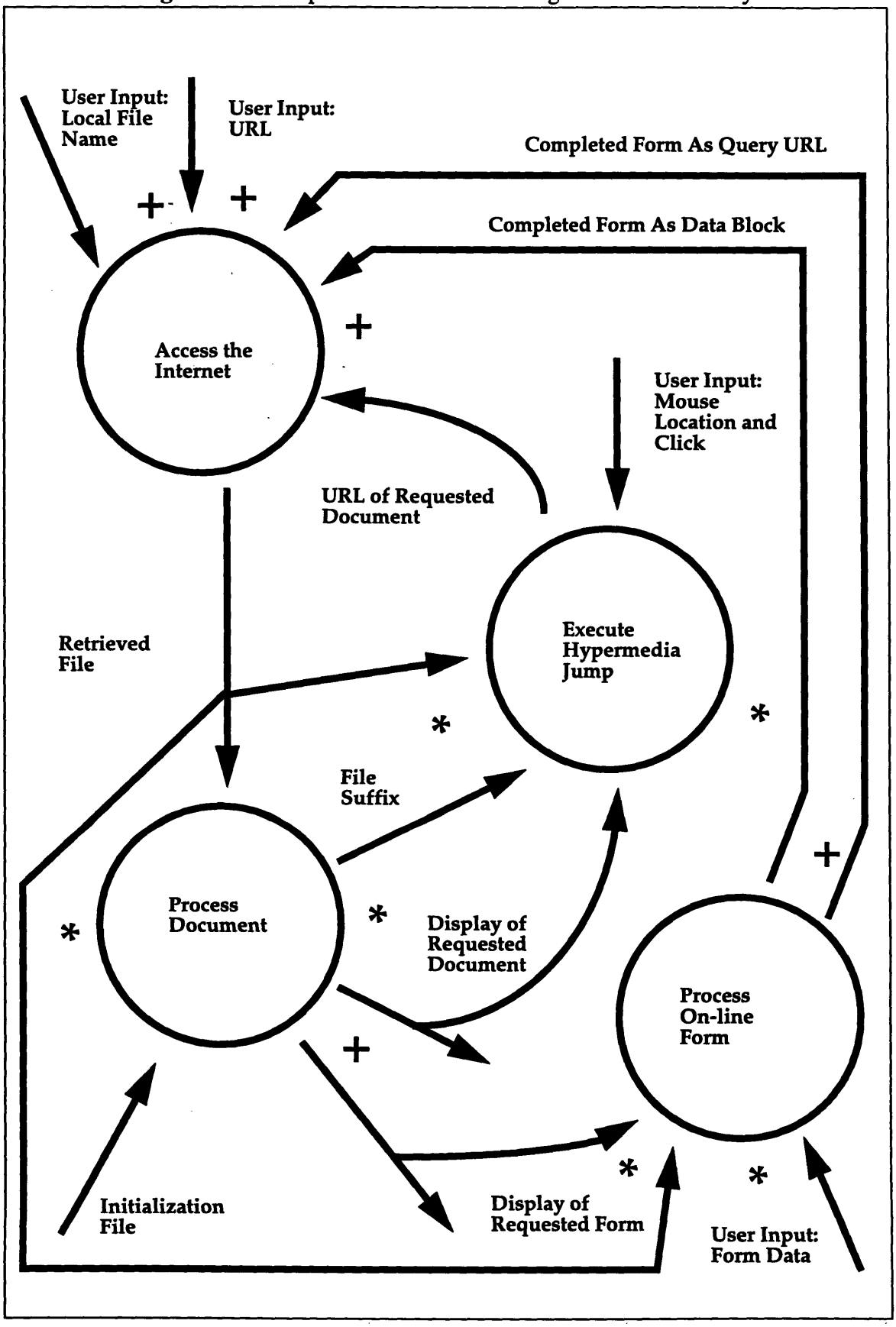
The following articles and texts were consulted in analyzing the program behavior of Navigator: Using Netscape, Inside the World Wide Web, The Mosaic Roadmap, and various documents downloaded from the Web sites of NCSA and Netscape Communications.

Because Netscape Communications no longer distributes older versions of Navigator, a working copy of Navigator Version 1.0 for X Window could not be obtained for program testing. Instead, a copy of Navigator Version 2.0 for X Window was installed and tested in order to become familiar with Navigator's program behavior. Nevertheless, Version 1.0 remained the focus of this case study.

3.3.1.1 Functionality

Analyzing Navigator as a collection of black boxes, a set of data-flow graphs representing its functionality was constructed. There are six data-flow graphs in all--one high-level graph labelled "Graph N" and four low-level graphs labelled "Graph N.1," "Graph N.1.1," "Graph N.2," "Graph N.3," and "Graph N.4." Graph N depicts the overall functionality of Navigator and serves as a guide for the remaining graphs.

Figure 3.14: Graph N: Overview of Navigator's Functionality



Graph N comprises four transform bubbles, each representing a major unit of functionality embodied in Navigator:

1) "Access the Internet" represents Navigator's ability to locate and retrieve information stored anywhere on the Internet through HTTP connections as well as through existing Internet tools like FTP and Gopher.

2) "Process Document" represents Navigator's ability to recognize and display a wide range of information including text, graphics, sound, and video.

3) "Execute Hypermedia Jump" represents Navigator's ability to execute a hypermedia jump across the Internet through the single click of a mouse.

4) "Process On-line Form" represents Navigator's ability to support the use of an on-line fill-out form. Fill-out forms are special HTML objects that allow users to enter information within text boxes and send them immediately to a destination server.

   According to the HTML specification, forms should be specified using a dedicated FORM tag consisting of the following attributes: ACTION, METHOD, and ENCTYPE. ACTION specifies the URL of the destination server, and METHOD indicates the method to be used in submitting the form contents. For the METHOD attribute, there are two choices: GET and POST. Finally, ENCTYPE specifies the encoding to be used in assembling the form contents.

The inputs represented in Graph N are as follows:

1) "User Input: Local File Name" represents a user-specified name of a file stored on the client machine.

2) "User Input: URL" represents the user-specified Uniform Resource Locator of information stored anywhere on the Internet.

3) "User Input: Mouse Location and Click" represents the user-specified location and click of a mouse.

4) "Initialization File" represents a file specifying the configuration of Navigator's document processing features.

5) "User Input: Form Data" represents the form contents submitted by the user.

The intermediate outputs represented in Graph N are as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

2) "File Suffix" represents the last element of a file's name, indicating the file type.

3) "URL of Requested Document" represents the Uniform Resource Locator of information stored anywhere on the Internet.

4) "Completed Form As Query URL" represents form data assembled as a query URL. In the case of a GET method, the form contents are assembled into a query URL of the following form: action?name=value&name=value&name=value. The term "action" represents the URL specified by the ACTION attribute. Similarly, the term "name" indicates the symbolic name of a text entry field, and the term "value" indicates the value submitted by the user.

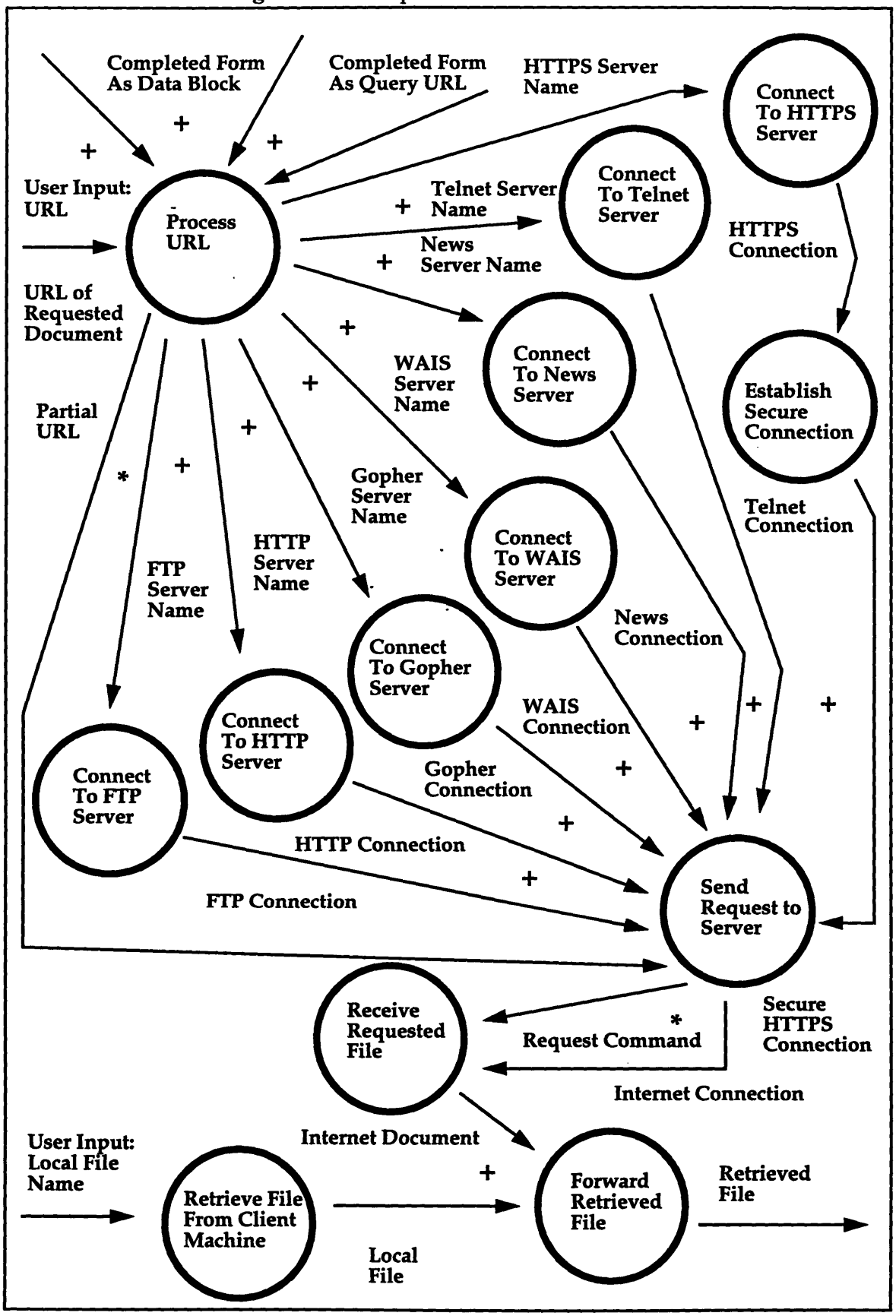
5) "Completed Form As Data Block" represents form data assembled as a data block. In the case of a POST method, the form contents are assembled in the same way as in the case of a GET method. The difference lies in the way the assembled data is sent to the destination server. In this case, it is sent in a data block through a POST operation.

The final outputs represented in Graph N are as follows:

1) "Display of Requested Document" represents a visual display or audio playback of the requested document.

2) "Display of Requested Form" represents a visual display of the requested on-line fill-out form.

Figure 3.15: Graph N.1: Access the Internet



Representing the function of accessing the Internet, Graph N.1 consists of the following transform bubbles:

1) "Process URL" represents Navigator's ability to recognize different types of network protocols based on the prefix of a URL.

2) "Connect to Telnet Server" represents Navigator's ability to support Telnet connections natively.

3) "Connect to News Server" represents Navigator's ability to support Network News Transfer Protocol (NTTP)-based connections natively.

4) "Connect to Gopher Server" represents Navigator's ability to support Gopher-based connections natively.

5) "Connect to HTTP Server" represents Navigator's ability to support HTTP-based connections natively.

6) "Connect to FTP Server" represents Navigator's ability to support FTP-based connections natively.

7) "Connect to WAIS Server" represents Navigator's ability to support WAIS-based connections natively.

8) "Connect to HTTPS Server" represents Navigator's ability to support HTTPS-based connections natively. Developed by Netscape, HTTPS is a special protocol for implementing a Secure Sockets Layer (SSL) underneath a regular HTTP protocol.

   SSL is an open standard for providing data security. Layered between an application protocol (e.g. HTTP and Telnet) and the TCP/IP protocol, SSL provides a comprehensive set of security features including server authentication, data encryption, message integrity, and optional client authentication. In providing the above set of features, SSL incorporates well-known and proven technologies such as public key encryption and signed certificates.

   Specifying a different protocol for establishing secure connections--HTTPS instead of just HTTP--prevents accidental exposure of sensitive data (Netscape Communications Corp., "Netscape Data Security"). With this simple scheme, non-SSL browsers can gracefully refuse to submit form contents that are expected to be submitted securely. For instance, if a fill-out form requires the user to submit her credit card information, the form should be specified using an HTTPS URL. This way, a non-SSL browser is barred from even trying to submit the form. On the oth-

er hand, if the form were to be specified using an HTTP URL, the browser would still attempt to submit the form, exposing the credit card information over the Internet.

9) "Establish Secure Connections" represents Navigator's ability to establish a secure connection based on the SSL standard. This transform bubble is further described in Graph N.1.1.

10) "Send Request to Server" represents Navigator's ability to request the transfer of a file located on a remote server.

11) "Receive Requested File" represents Navigator's ability to retrieve a requested file using one of the Internet protocols or services discussed above.

12) "Retrieve File From Client Machine" represents Navigator's ability to retrieve a requested file located on the client machine through a local file service.

13) "Forward Retrieved File" represents Navigator's ability to forward a retrieved file to one of its internal document processing utilities or to an external viewer application.

The inputs represented in Graph N.1 are as follows:

1) "User Input: Local File Name" represents a user-specified name of a file stored on the client machine.

2) "User Input: URL" represents the user-specified Uniform Resource Locator of information stored anywhere on the Internet.

3) "Completed Form As Query URL" represents form data assembled as a query URL.

4) "Completed Form As Data Block" represents form data assembled as a data block.

5) "URL of Requested Document" represents Uniform Resource Locator of information stored anywhere on the Internet.

The intermediate outputs represented in Graph N.1 are as follows:

1) "Telnet Server Name" represents the Internet address of a Telnet server.

2) "Telnet Connection" represents a connection to a Telnet server.

3) "News Server Name" represents the Internet address of a Usenet News server.

4) "News Connection" represents a connection to a Usenet News server.

5) "Gopher Server Name" represents the Internet address of a Gopher server.

6) "HTTP Server Name" represents a connection to a Gopher server.

7) "FTP Server Name" represents the Internet address of an FTP server.

8) "FTP Connection" represents a connection to an FTP server.

9) "WAIS Server Name" represents the Internet address of a WAIS server.

10) "WAIS Connection" represents a connection to a WAIS server.

11) "HTTPS Server Name" represents the Internet address of an HTTPS server.

12) "HTTPS Connection" represents a connection to an HTTPS server.

13) "Partial URL" represents the remainder of the URL string following the prefix and the server name.

14) "Request Command" represents a command requesting a remote server to send a document to the client machine.

15) "Internet Connection" represents an Internet connection using one of the application protocols supported by Navigator (e.g. HTTP).

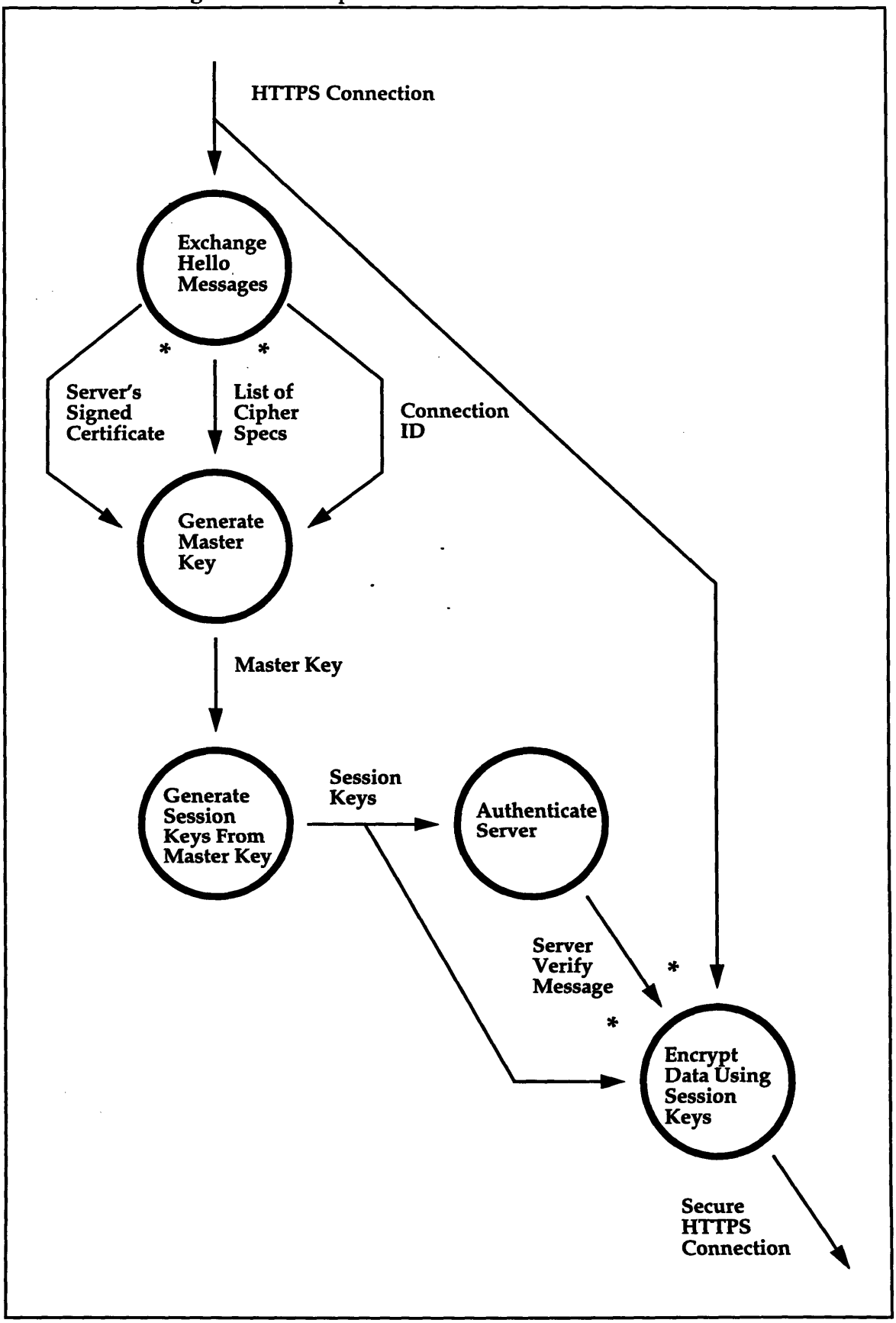
16) "Internet Document" represents a document retrieved from a remote server located somewhere on the Internet.

17) "Local File" represents a document retrieved from the client machine through a local file service.

The final output represented in Graph N.1 is as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

Figure 3.16: Graph N.1.1: Establish Secure Connection



The main distinguishing feature of Navigator is its ability to establish secure Internet connections through the use of encryption technology. Without added security measures, the Internet is vulnerable to fraud and other types of misuse by intermediaries or other users (Netscape Communications Corp., "On Security"). Because the Internet is based on a packet-switching model of networks, data traveling between a host machine and a server is likely to traverse numerous computer systems located in different places and operated by different organizations or corporations. Each of these systems represents a potential security threat, making the Internet a dangerous place to conduct commercial transactions or to engage in other types of sensitive communications.

An effective solution to the lack of security on the Internet is data encryption. Encryption refers to the act of transforming a set of data in such a way that it is no longer readable to anyone except those with the appropriate key. By encrypting all incoming and outgoing data, secure connections enable users to exchange information privately with a remote server, making it possible to conduct commercial transactions over the Internet safely and reliably. Using data encryption, Navigator establishes secure Internet connections by implementing a special HTTPS (i.e. HTTP plus SSL) protocol in accordance with the SSL standard (Netscape Communications Corp., "On Security").

Representing the function of establishing a secure connection, Graph N.1.1 consists of the following transform bubbles:

1) "Exchange Hello Messages" represents Navigator's ability to send and receive identifying messages with an SSL-enabled HTTPS server. Navigator initiates a secure connection by sending a CLIENT-HELLO message to an HTTPS server. In response, the server returns a SERVER-HELLO message containing the server's signed certificate, a list of cipher specifications, and the connection ID.

2) "Generate Master Key" represents Navigator's ability to generate a master key based on the information contained in the SERVER-HELLO message. Generated once per session, the master key is used, in turn, to generate a pair of session keys

for every connection corresponding to the current session. The session keys are then used to encrypt the incoming and outgoing data.

3) "Generate Session Keys From Master Key" represents Navigator's ability to generate a pair of session keys from a master key. For each HTTPS connection, Navigator uses the master key corresponding to the current session ID to generate a new pair of session keys. A session is simply a series of related HTTPS connections.

4) "Authenticate Server" represents Navigator's ability to authenticate an HTTPS server. To authenticate a server is to verify its claimed identity. With Navigator, server authentication is performed through the use of public key encryption and signed certificates, as specified in the SSL specification.

Private key encryption is a cryptography technique that utilizes a pair of asymmetric keys for encryption and decryption. Each pair of keys consists of a public key and a private key: The public key is distributed publicly, and the private key is kept secret. When data is encrypted using the public key, it can only be decrypted using the private key, and vice versa.

This asymmetric property allows two parties (e.g. a client and a server) to verify each other's identity. For instance, suppose Alice wants to authenticate Bob, who has a pair of keys, one public and one private (Netscape Communications Corp., "Using RSA Public Key Cryptography for Internet Security"). She can do this by sending a random message to Bob and requesting that Bob encrypt the message using his private key and return the encrypted version back to Alice. Upon receiving the encrypted version of her message, Alice can decrypt it using Bob's previously published public keys. If the decrypted message matches the original message she sent to Bob, then Alice knows that she's communicating with Bob.

In general, it is not wise to encrypt arbitrary messages with one's private key and send them over the Internet because the encrypted value can be used to reverse engineer the private key (Netscape Communications Corp., "Using RSA Public Key Cryptography for Internet Security"). As an alternative, Bob, in the above example, can construct a digest of the original message and encrypt it instead. A digest is a transformation of the original message that exhibits the following useful properties: (1) It is difficult to reverse; and (2) it yields unique results for different messages. By encrypting and sending a digest of Alice's random message rather than the message itself, Bob can protect himself against attempts at reverse engineering his private key. In this case, Alice can authenticate Bob by decrypting his version of the message, computing a digest of her original message, and then comparing the two values.

There is one problem with the above scheme for authenticating Bob. Prior to authentication, an impostor could lie to Alice and submit his public key as Bob's (Netscape Communications Corp., "Using RSA Public Key Cryptography for Internet Security"). This impostor can then encrypt Alice's message and pretend to be Bob. In this case, Alice would have no way of knowing that the person with whom she's communicating is not Bob.

In order to protect against such attempts, Navigator requires all HTTPS servers to submit a signed certificate obtained from a certificate authority. As a reputable third-party organization, a certificate authority confirms the authenticity of each certificate request and issues a unique certificate consisting of the following: the certificate issuer's name, whom the certificate is being issued for, the public key of the server, and a set of time stamps. The certificate is then encrypted or "signed" using the certificate issuer's private key.

Assuming that the certificate issuer's public key is widely distributed and verified, Alice, as a Navigator user, can examine the signed certificate of any HTTPS server, including that of Bob, to obtain its public keys in a safe and reliable manner. After obtaining Bob's public key from his certificate, Alice can then verify the identity of anyone who claims to be Bob by following the procedure described below (A= Alice and B = person claiming to be Bob) (Netscape Communications Corp., "Using RSA Public Key Cryptography for Internet Security"):

1) A sends to B: ("Hello").
2) B sends to A: ("Hi, I'm Bob"; Bob's certificate).
3) A decrypts Bob's certificate using the public key of the certificate issuer.
4) A sends to B: ("Prove it").
5) B sends to A: ("Alice, this is Bob"; a digest of "Alice, this is Bob" encrypted using Bob's private key).
6) A decrypts the encrypted portion of B's message using Bob's public key as specified in his certificate.
7) A computes a digest of the clear portion of B's message ("Alice, this is Bob") using the same function that B used earlier.
8) A compares the two values.
9) If there is a match, A sends to B: ("Ok, Bob").
10) Otherwise, A sends to B: ("I cannot verify you as Bob").

The above procedure, more or less, is used by Navigator to provide server authentication.

5) "Encrypt Data Using Session Keys" represents Navigator's ability to decrypt incoming data and encrypt outgoing data using session keys.

The inputs represented in Graph N.1.1 are as follows:

1) "HTTPS Connection" represents an initial connection with an HTTPS server. The connection, at this point, is not secure.

The intermediate outputs represented in Graph N.1.1 are as follows:

1) "Server's Signed Certificate" represents a digital certificate issued and signed by a trusted certificate authority. A certificate contains the following: the certificate issuer's name, whom the certificate is being issued for, the public key of the server, and a set of time stamps.

2) "List of Cipher Specs" represents a list containing the type of encryption algorithms supported by the server. Navigator Version 1.0 recognizes the following algorithms: RC4, RC2, IDEA, and DES.

3) "Connection ID" represents a string of randomly generated bytes used to identify an HTTPS connection. The length must be between 16 and 32 bytes, inclusive.

4) "Master Key" represents a secret key generated by Navigator for each HTTPS session. At the beginning of each session, Navigator generates a new master key and sends it to the server in two parts: (1) a clear portion; and (2) a secret portion, which is encrypted using the server's public key. Thereafter, Navigator checks the session ID of each request for an HTTPS connection. If it matches a previous session ID, then both Navigator and the server proceed to use the appropriate master key in cache. Otherwise, Navigator generates a new master key and forwards it to the server.

5) "Session Keys" represents a pair of keys used to encrypt and decrypt data transfers during an HTTPS connection. For each HTTPS connection, Navigator uses the master key corresponding to the current session ID to generate a new pair of session keys. Each pair contains the following keys: (1) a "SERVER-READ-KEY," also known as the "CLIENT-WRITE-KEY," for outgoing data; and (2) a "SERVER-WRITE-KEY," also known as the "CLIENT-READ-KEY," for incoming data.

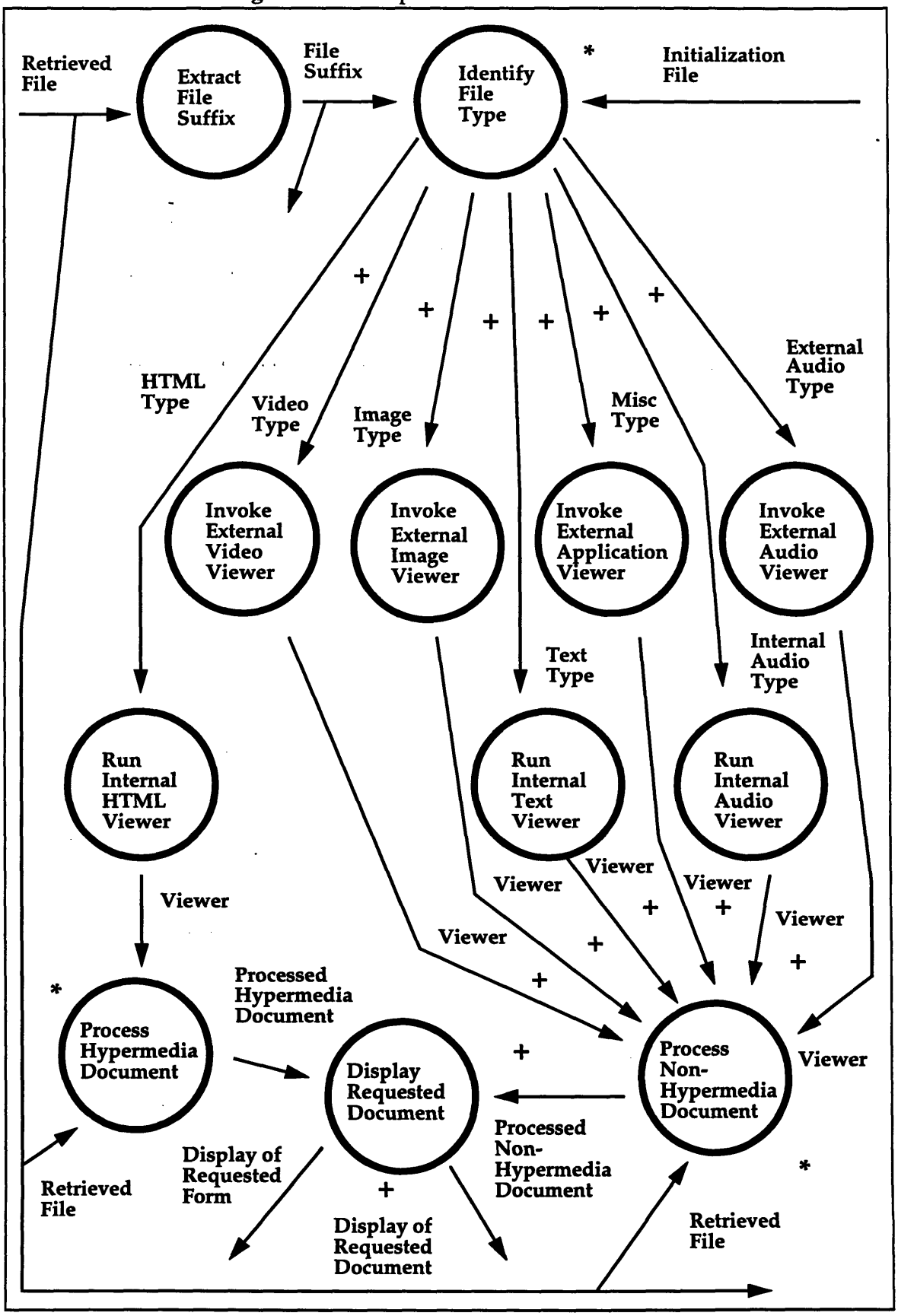
6) "Server Verify Message" represents a message sent by an HTTPS server to allow authentication by Navigator. Usually, the SERVER-VERIFY-MESSAGE consists of two parts: the challenge data and an encrypted version of the challenge data. In the previous example of Alice and Bob, the following is the SERVER-VERIFY-MESSAGE: ("Alice, this is Bob"; a digest of "Alice, this is Bob" encrypted using Bob's private key).

The final output represented in Graph N.1.1 is as follows:

1) "Secure HTTPS Connection" represents a secure connection with an HTTPS server. The connection has been established by successfully following the SSL protocol.

   With a secure HTTPS connection, all incoming and outgoing data are encrypted prior to and during transmission. Both types of data are decrypted only at the endpoints of the connection by Navigator and the HTTPS server--effectively ensuring privacy. In addition, because HTTPS connections are established through the SSL standard, data encryption and decryption are transparent to the application protocols (e.g. HTTP, Telnet, or FTP) and invisible to the user.

Figure 3.17: Graph N.2: Process Document



Representing the function of processing a retrieved document, Graph N.2 consists of the following transform bubbles:

1) "Extract File Suffix" represents Navigator's ability to extract the suffix corresponding to a retrieved file.

2) "Identify_File Type" represents Navigator's ability to recognize various types of files based on the file suffix.

3) "Invoke External Video Viewer" represents Navigator's ability to process video files by invoking an external viewer.

4) "Invoke External Image Viewer" represents Navigator's ability to process image files by invoking an external viewer.

5) "Invoke External Audio Viewer" represents Navigator's ability to process audio files by invoking an external viewer.

6) "Invoke External Application Viewer" represents Navigator's ability to process special types of files by invoking external viewers.

7) "Run Internal HTML Viewer" represents Navigator's ability to process HTML documents natively.

8) "Process Hypermedia Document" represents Navigator's ability to recognize and process hypermedia documents in HTML format.

9) "Run Internal Text Viewer" represents Navigator's ability to process text files natively.

10) "Run Internal Audio Viewer" represents Navigator's ability to process audio files in AU format natively.

11) "Process Non-Hypermedia Document" represents Navigator's ability to process a wide range of non-hypermedia documents by invoking either an internal or external viewer.

12) "Display Requested Document" represents Navigator's ability to display the requested document using a number of peripheral devices--including a monitor in the case of text, image, or video files; and a speaker in the case of audio files.

The inputs represented in Graph N.2 are as follows:

1) "Retrieved File" represents a file retrieved from the client machine or from a remote server machine.

2) "Initialization File" represents a file specifying the configuration of Navigator's document processing features.

The intermediate outputs represented in Graph N.2 are as follows:

1) "File Suffix" represents the last element of a file name, indicating the file type.

2) "HTML Type" represents a hypermedia document type in HTML format, indicated by the file suffix ".html."

3) "Video Type" represents a video document type.

4) "Image Type" represents an image document type.

5) "Text Type" represents a text document type.

6) "External Audio Type" represents an audio document type not supported natively by Navigator.

7) "Internal Audio Type" represents an audio document type supported natively by Navigator.

8) "Misc Type" represents miscellaneous document types that can be supported by invoking external viewers.

9) "Viewer" represents one of the various external viewers supported by Navigator.

10) "Processed Hypermedia Document" represents a processed version of a hypermedia document that is ready to be displayed. In addition to text and graphics, the HTML document may also contain on-line fill-out forms.

11) "Processed Non-Hypermedia Document" represents a processed version of a non-hypermedia document that is ready to be displayed.

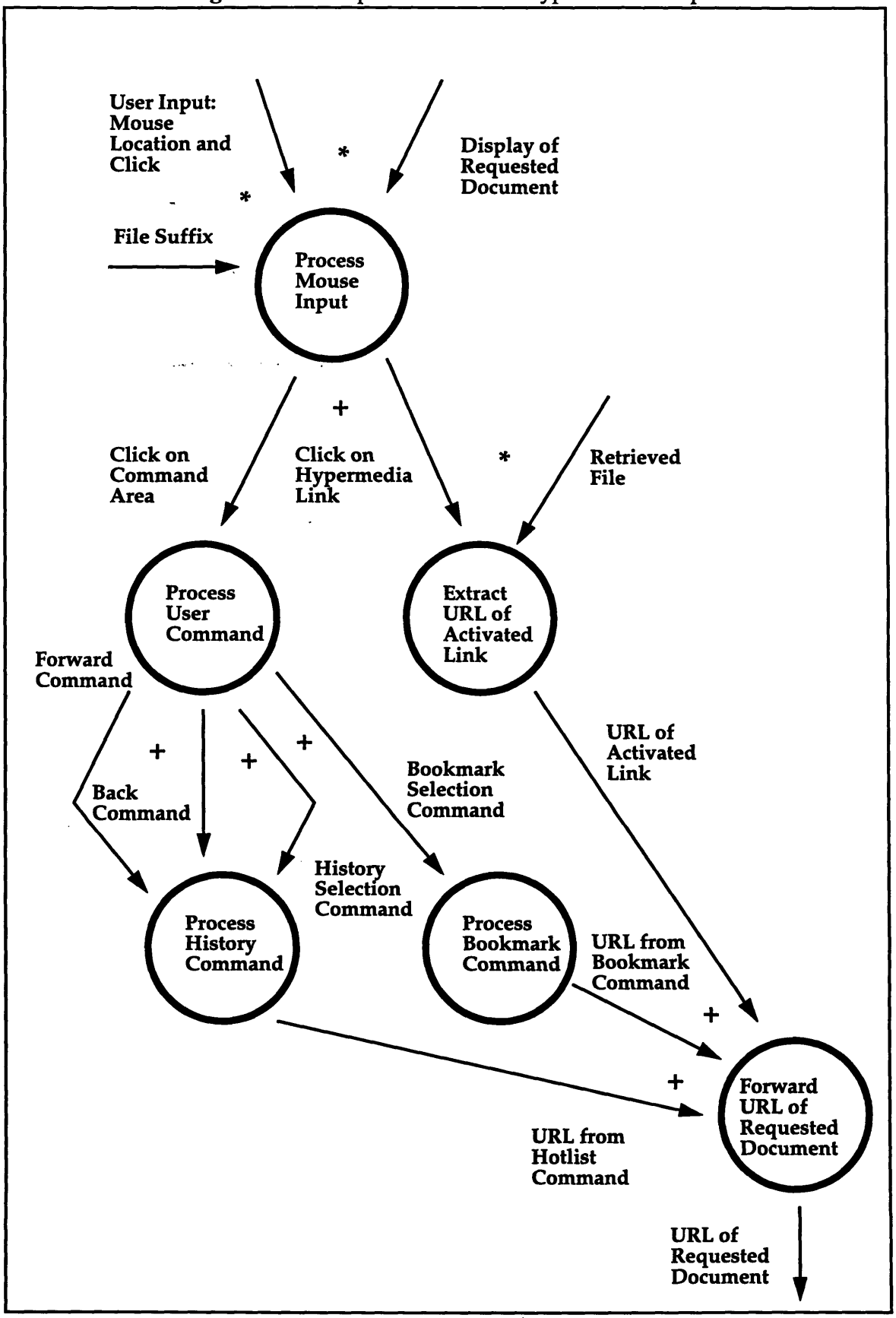
The final outputs represented in Graph N.2 are as follows:

1) "File Suffix" represents the last element of a file name, indicating the file type.

2) "Display of Requested Document" represents a screen display of the requested document, in the case of text and image files, or a playback of the requested document, in the case of audio and video files.

3) "Display of Requested Form" represents a screen display of the requested on-line form, in the case of special HTML documents.

Figure 3.18: Graph N.3: Execute Hypermedia Jump



Representing the function of executing a hypermedia jump, Graph N.3 consists of the following transform bubbles:

1) "Process Mouse Input" represents Navigator's ability to translate the location and click of a mouse into recognizable commands.

2) "Process User Command" represents Navigator's ability to recognize and process mouse clicks on screen areas that correspond to user commands.

3) "Extract URL of Activated Link" represents Navigator's ability to extract the URL associated with an activated link by referring to the retrieved HTML file.

4) "Process History Command" represents Navigator's ability to keep track of the documents displayed during a session.

5) "Process Bookmark Command" represents Navigator's ability to maintain a personalized list of documents and Web sites from session to session. Navigator's bookmark command can be viewed as an enhanced version of Mosaic's hotlist feature.

6) "Forward URL of Requested Document" represents Navigator's ability to forward the URL of the next document to be retrieved.

The inputs represented in Graph N.3 are as follows:

1) "User Input: Mouse Location and Click" represents the user-specified location and click of a mouse.

2) "Display of Requested Document" represents a screen display of the requested document, in the case of text and image files, or a playback of the requested document, in the case of audio and video files.

3) "File Suffix" represents the last element a file name, indicating the file type.

4) "Retrieved File" represents a file retrieved from the client machine or from a remote server machine.

The intermediate outputs represented in Graph N.3 are as follows:

1) "Click on Command Area" represents a mouse click on an icon or a menu entry corresponding to a user command.

2) "Click on Hypermedia Link" represents a mouse click on a hypermedia link indicated by highlighted text or image.

3) "Back Command" represents a command to display documents previously displayed during the current session.

4) "Forward Command" represents a command to display documents previously displayed during the current session.

5) "History Selection Command" represents a command to select a previous document from the history list.

6) "Bookmark Selection Command" represents a command to select a document or Web site from a list of bookmarks.

7) "URL of Activated Link" represents the URL corresponding to an activated link.

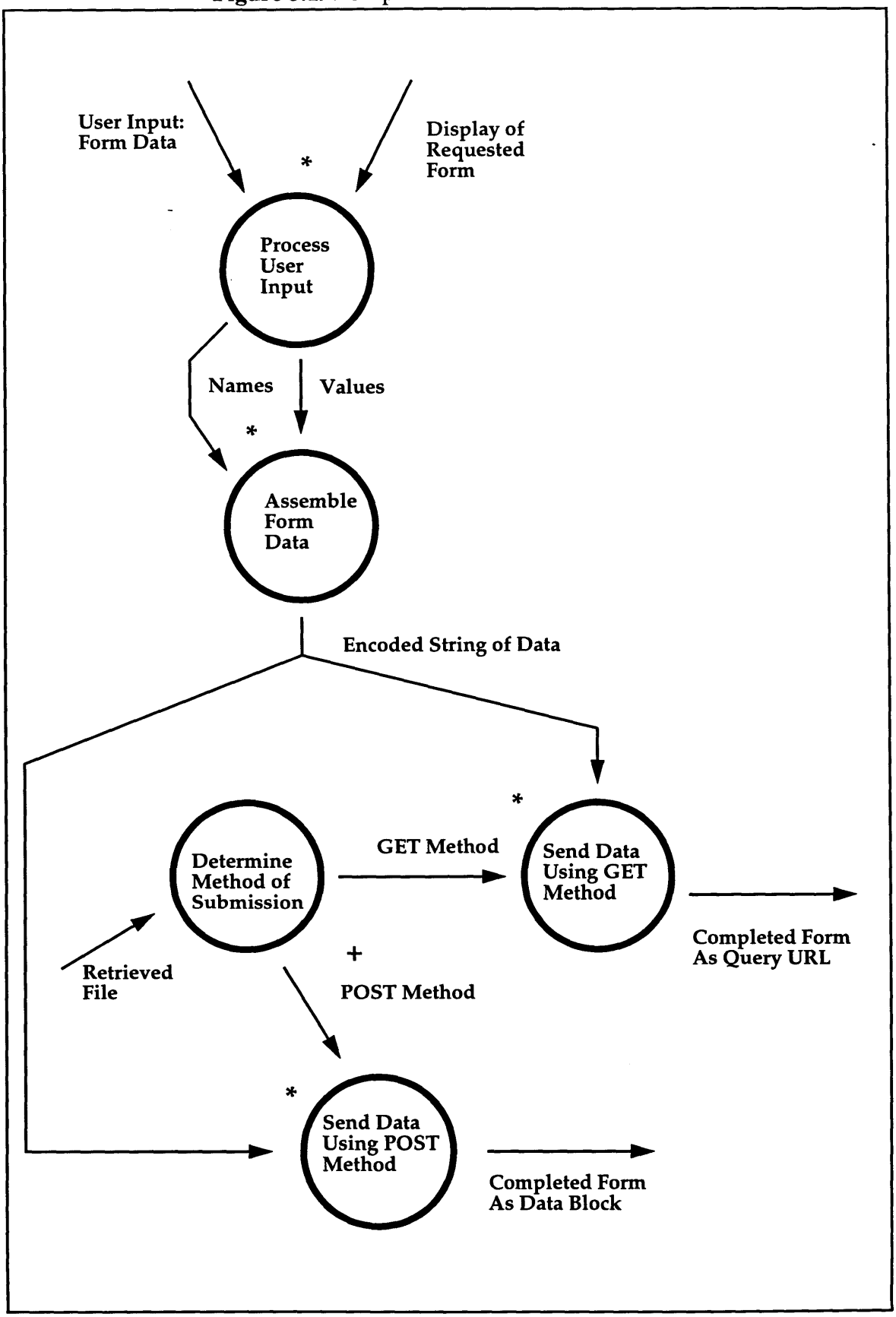
8) "URL from History Command" represents the URL corresponding to a history command.

9) "URL from Bookmark Command" represents the URL corresponding to a bookmark command.

The final output represented in Graph N.3 is as follows:

1) "URL of Requested Document" represents the URL corresponding to an activated link, a history command, or a bookmark command.

Figure 3.19: Graph N.4: Process On-line Form



In accordance with the latest HTML specifications, Navigator provides full support for on-line fill-out forms. A fill-out form consists of a set of text entry fields where users can directly enter values. For instance, consider an HTML document that contains a fill-out form for credit card purchases. With this document, the user can initiate a purchase by placing the cursor directly on the entry field labelled "Credit Card Number" and typing the number of her credit card. Similarly, the user can enter the appropriate values for entry fields labelled "Expiration Date" and "Amount of Purchase." After all the values have been entered, the user can forward them to the desired server by simply clicking a button labelled "Submit." Upon detecting a click on the submit button, Navigator assembles the corresponding form data and sends it to the destination server in one of two ways: Depending on the selected method of submission, the Navigator can append the data to the query URL directly or post the data as a data block.

In brief, on-line fill-out forms provide a convenient and fast way for servers to request and receive information from their clients. Possible applications include on-line shopping, on-demand database searches, and off-site registrations (NCSA, "Mosaic for X version 2.0 Fill-Out Form Support").

Representing the function of processing an on-line form, Graph N.4 consists of the following transform bubbles:

1) "Process User Input" represents Navigator's ability to recognize and process the entry of data corresponding to on-line fill-out forms.

2) "Assemble Form Data" represents Navigator's ability to assemble the contents of a form upon detecting a request for submission. Navigator assembles the form contents by constructing a string of text consisting of pairs of names and values separated by special characters.

3) "Determine Method of Submission" represents Navigator's ability to support two distinct methods for submitting form data: the GET and the POST methods. Selecting the GET method instructs Navigator to append the form contents to the URL of the server, as in the case of a normal query. Alternately, selecting the

POST method instructs Navigator to send the form contents to the server in a data block, rather than as part of the URL.

4) "Send Data Using GET Method" represents Navigator's ability to send form data as a query URL.

5) "Send Data Using POST Method" represents Navigator's ability to send form data as a data block.

The inputs represented in Graph N.4 are as follows:

1) "User Input: Form Data" represents form data submitted by the user.

2) "Display of Requested Form" represents a visual display of the requested on-line fill-out form.

3) "Retrieved File" represents a file retrieved from the client machine or from a remote server.

The intermediate outputs represented in Graph N.4 are as follows:

1) "Names" represents labels corresponding to text entry fields of a fill-out-form.

2) "Values" represents data corresponding to text entry fields of a fill-out form, as submitted by the user.

3) "Encoded String of Data" represents a string of text consisting of pairs of names and values separated by special characters.

4) "GET Method" represents a recognition by Navigator that the form data should be submitted using the GET method.

5) "POST Method" represents a recognition by Navigator that the form data should be submitted using the POST method.

The final outputs represented in Graph N.4 are as follows:

1) "Completed Form As Query URL" represents the contents of a form assembled as a query URL.

2) "Completed Form As Data Block" represents the contents of a form assembled as a data block.

3.3.1.2 User Interface

In most cases, the user interface of a computer program can be best understood by observing its screen displays and taking snapshots. Unfortunately, a working copy of Navigator Version 1.0 for X Window could not be obtained for program testing. Instead, a copy of Navigator Version 2.0 was installed and tested. Accordingly, the following analysis of Navigator's user interface was based primarily on the testing results of Version 2.0 rather than that of Version 1.0. Although the two versions utilize slightly different interfaces, the existing literature indicates that the essential characteristics of Navigator's user interface remain unchanged from Version 1.0 to Version 2.0. The results of the analysis are presented in this subsection.

In many respects, Navigator looks and feels like its predecessor Mosaic (Vaughan-Nichols 302). Both programs incorporate a graphical user interface with integrated mouse support and utilize a screen layout with pull-down menus and command icons. They are also designed to work within a windows-based operating system like X Window. Finally, they even use similar icons in the upper right hand corner as status indicators: Mosaic uses a spinning globe, and Navigator uses a capital "N" with animated meteors in the background.

Navigator's user interface consists of ten distinct areas within the main window (Vaughan-Nichols 302-305):

1) Window title is used display the title of the document currently being displayed.

2) Menu bar is used to display program commands by organizing them into various functional menus. The available menus are File, Edit, Go, Bookmarks, Options, Directory, and Window.

3) Toolbar buttons are used to display many of the basic commands as buttons. Some of the commands displayed as toolbar buttons include Back, Forward, and Reload.

4) URL field is used to indicate the URL of the document currently being displayed.

5) Netscape logo is used to indicate the status of a file transfer. During a transfer, animated meteors fly across the logo. Clicking on this logo stops a transfer in progress and returns the user to the Netscape Communications's home page.

6) Content area is used to display the actual text and inlined images of the current document. A scroll bar is provided to accommodate documents that are longer or wider than the available area.

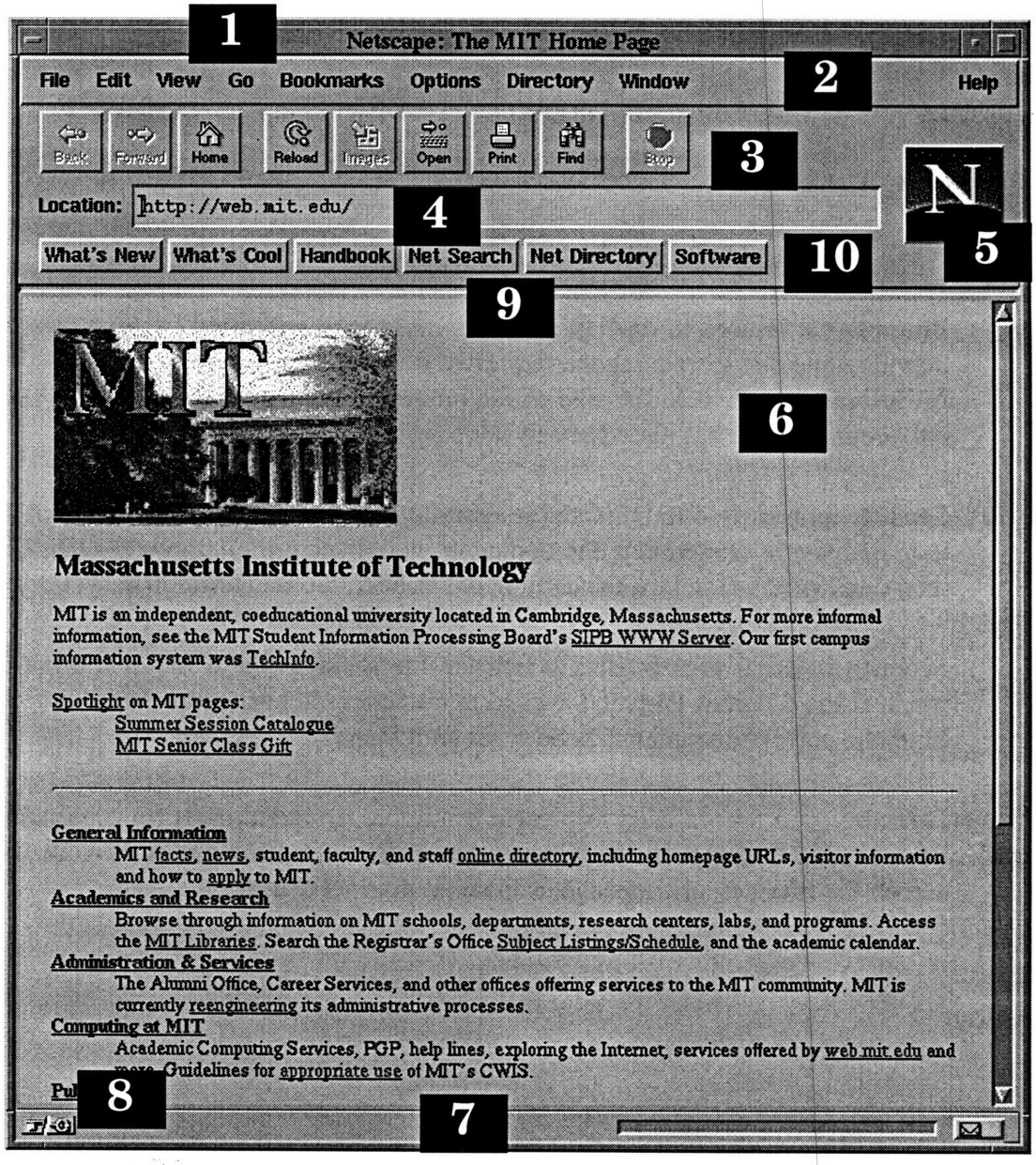
7) Progress bar is used to display the status messages generated by Navigator when downloading and displaying the requested document. These messages contain useful information such as the size of the requested document and the percentage of the document that has already been retrieved.

8) Security icon is used to indicate the security status of the current document. If the icon displays a broken key, the document is not secure. On the other hand, if the icon displays a whole key, the document has been secured through data encryption.

9) Security colorbar is also used to indicate the security status of the current document. If the colorbar is grey, the current document is not secure. If the colorbar is blue, the current document has been secured through encryption.

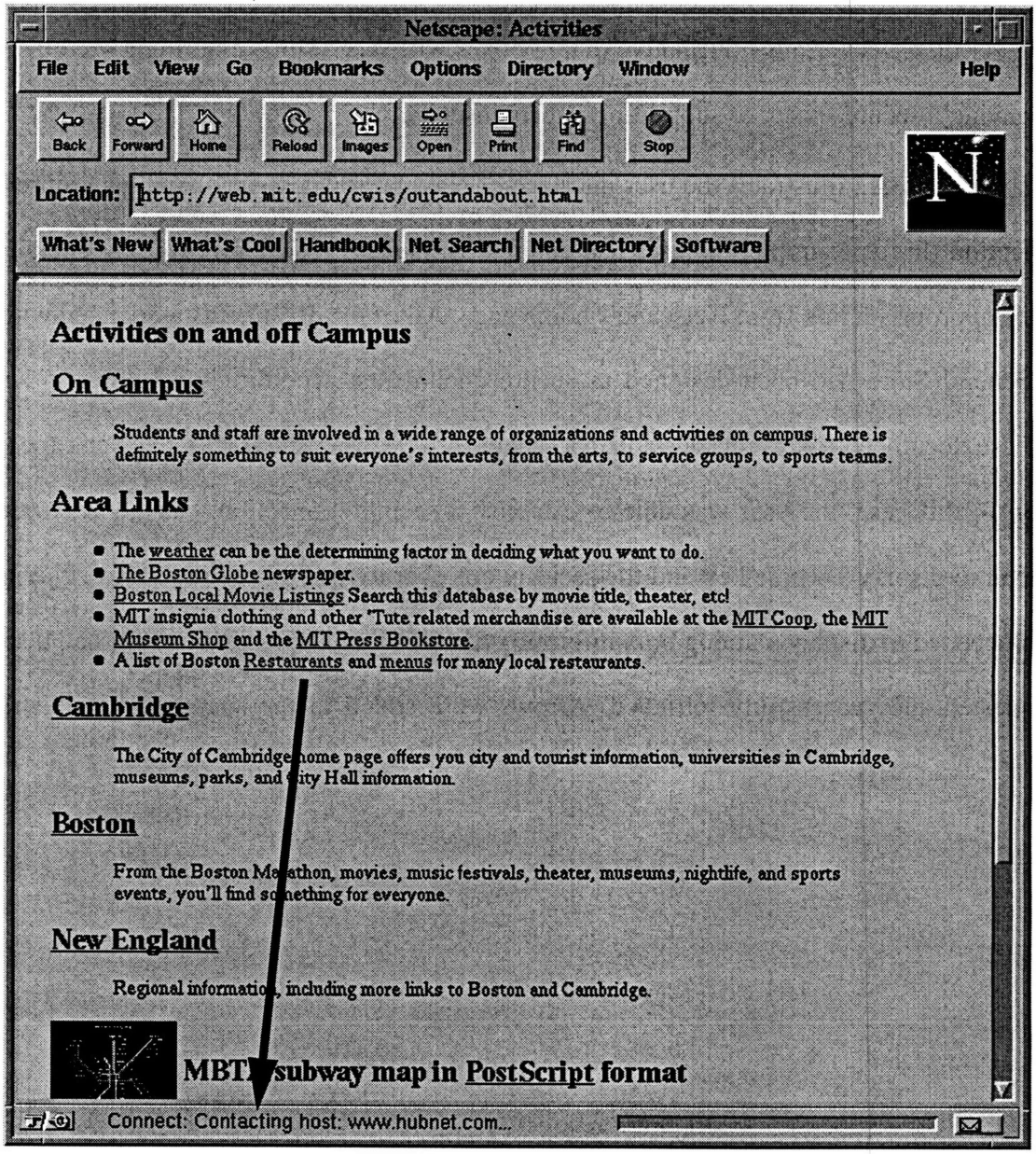
10) Directory buttons are used to present a number of useful tools for navigating the Internet. By clicking on these buttons, the user can learn about new Web sites, search the Internet, or choose sites from an Internet directory.

Figure 3.20: Snapshot of Navigator's Main Window



Navigator and Mosaic utilize similar interfaces for accessing the Internet. See Subsection 3.2.1.2 for a discussion of the interface and a description of the snapshot in Figure 3.21.

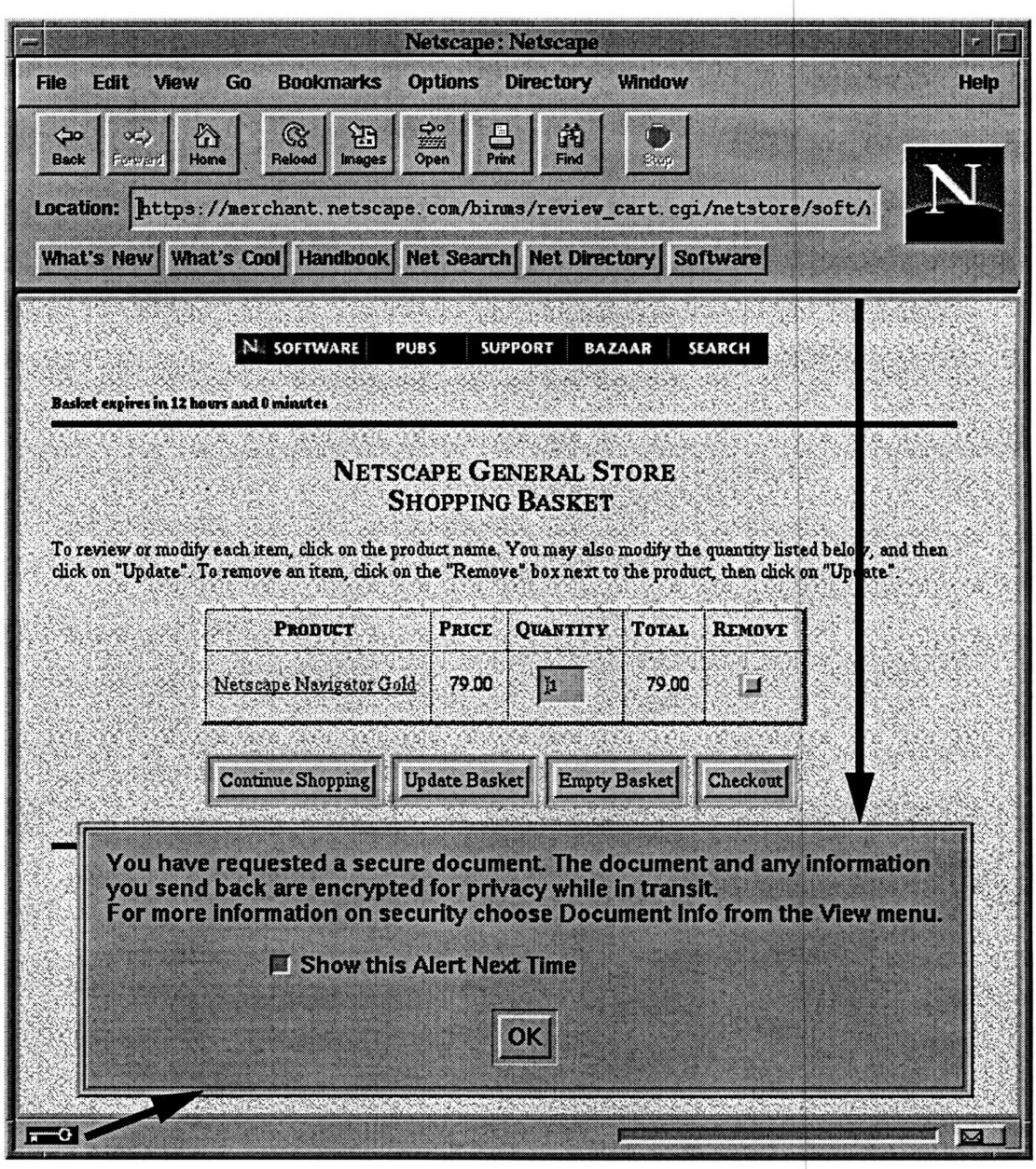
Figure 3.21: Snapshot of Navigator's Interface When Accessing the Internet



When establishing a secure connection, Navigator utilizes the security icon and the security colorbar to relay important security information. If the document is not secure, the security icon displays a broken key, and the colorbar turns grey. On the other hand, if the document is secure, the security icon displays a whole key, and the colorbar turns blue. In addition, Navigator provides in the File menu a so-called "Document Information" entry, which contains detailed information about the state of the current document. Finally, Navigator displays various informative dialog boxes when the user is entering a secure space, leaving a secure space, or submitting a form insecurely.

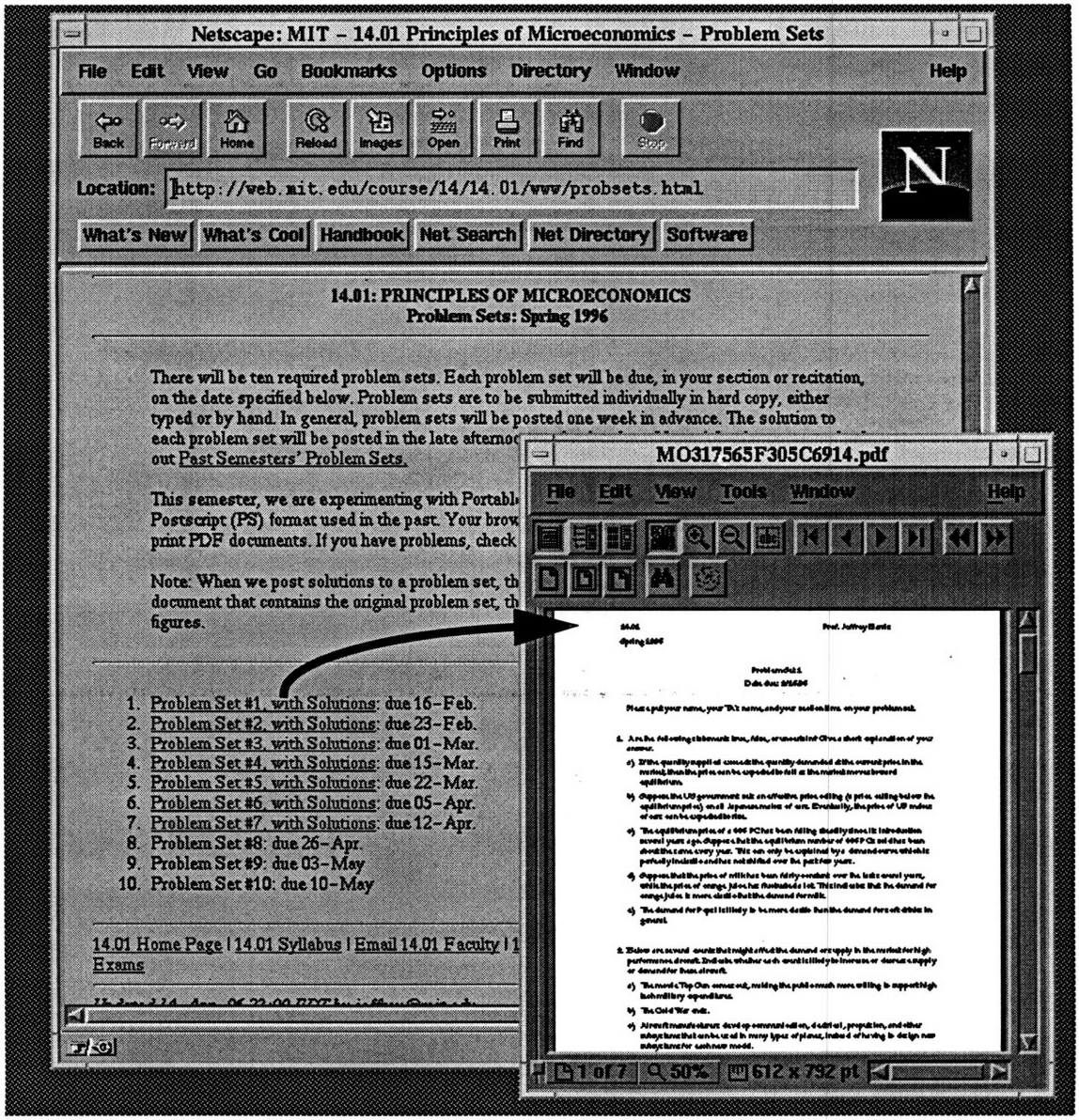
Figure 3.22 illustrates the user interface of Navigator when establishing a secure connection. In the example, the user had entered the Netscape General Store by clicking on the appropriate link from Netscape's home page. All of the documents associated with the General Store had been designed as secure documents; accordingly, in order to access these documents, users must establish a secure HTTPS connection with the Netscape server. In this case, the user was able to establish a secure connection, causing the security icon to display a whole key and the security colorbar to turn blue. In response, Navigator proceeded to display a dialog box, informing the user that a secure document had been requested and successfully retrieved. Arrows were added to the snapshot for illustrative effect.

Figure 3.22: Snapshot of Navigator's Interface When Establishing a Secure Connection



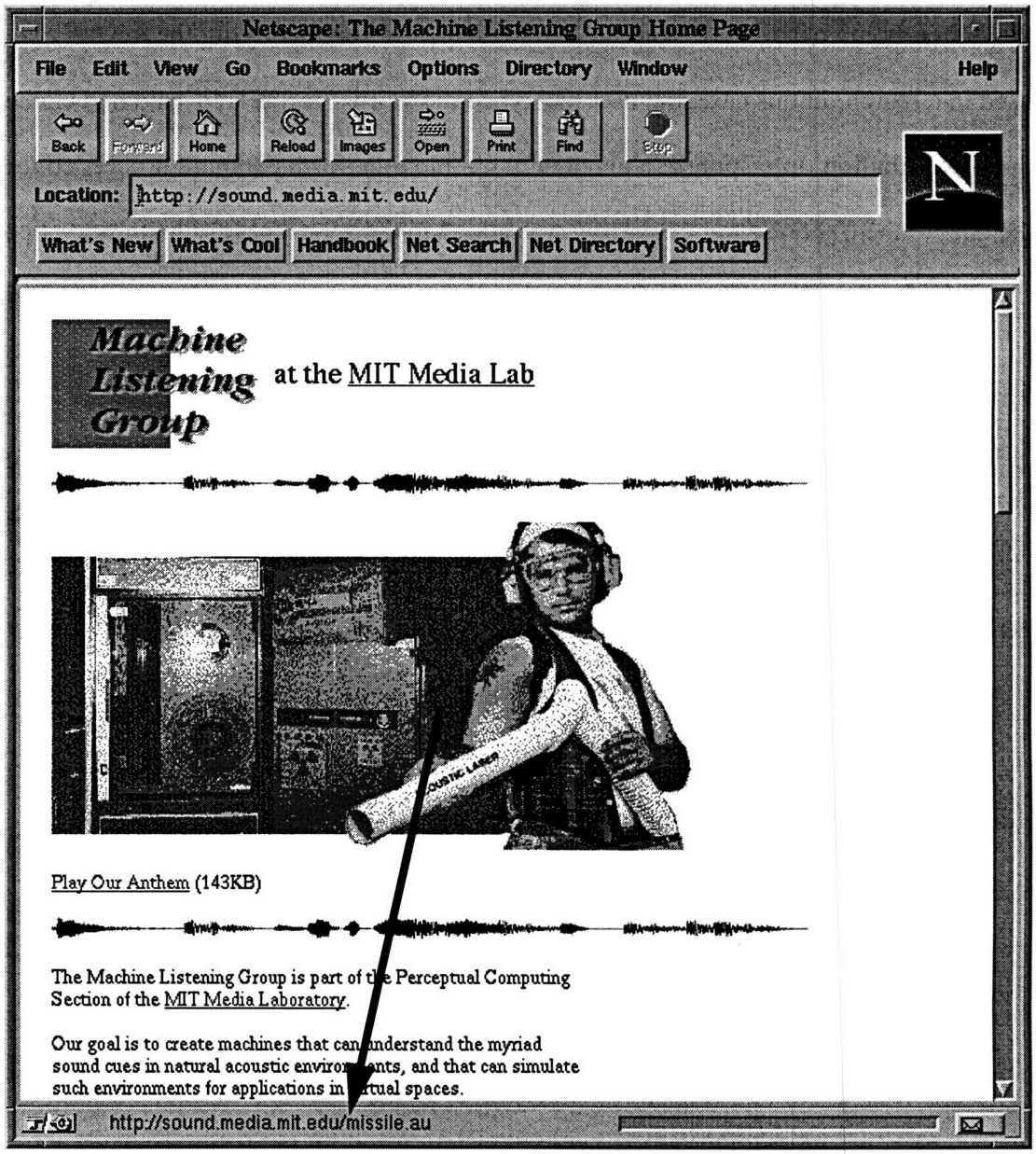
As with Internet access, Navigator and Mosaic utilize similar interfaces for document processing. See Subsection 3.2.1.2 for a discussion of the interface and a description of the snapshot in Figure 3.23.

Figure 3.23: Snapshot of Navigator's Interface When Processing a Retrieved Document



Similarly, Navigator and Mosaic employ almost identical interfaces for displaying and processing a hypermedia link. See Subsection 3.2.12 for a discussion of the interface and a description of the snapshot in Figure 3.24.

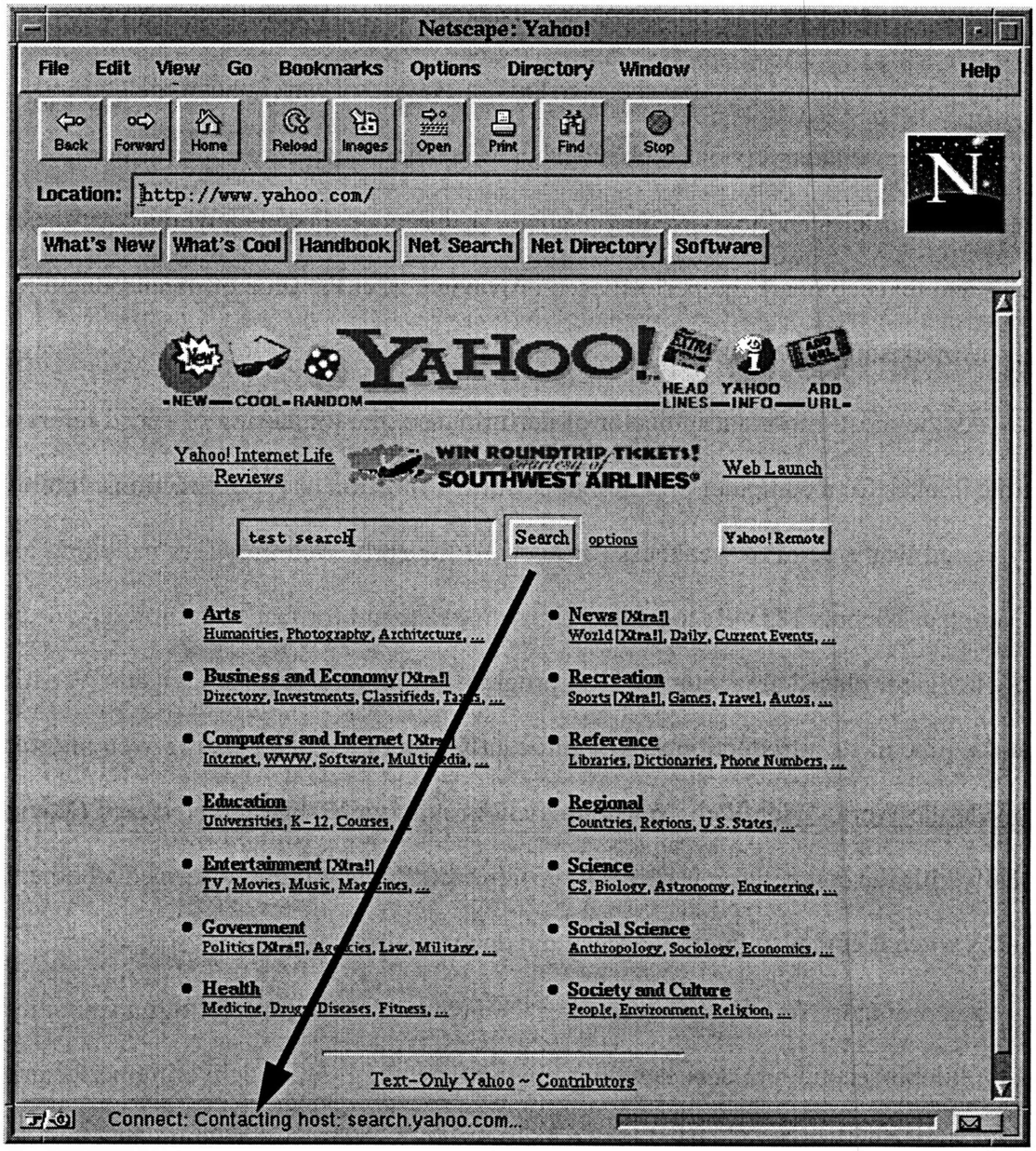
Figure 3.24: Snapshot of Navigator's Interface When Executing a Hypermedia Jump



Finally, when processing an on-line fill-out form, Navigator utilizes the same interface as when processing normal hypermedia documents. The only difference is that the user is allowed to type directly into the field entry areas of an on-line form and send the data to the destination server by clicking on a submit button.

Figure 3.25 illustrates the user interface of Navigator when processing an on-line form. In this example, the user had invoked Yahoo, a popular Internet search service. On the Yahoo home page, there is an on-line fill-out form for specifying the key words to be used in a search. In this case, the user had typed the words "test search" and had clicked on the submit button. In response, Navigator assembled the form data and proceeded to send the data to the Yahoo server. An arrow was added to the snapshot for illustrative effect.

Figure 3.25: Snapshot of Navigator's Interface When Processing an On-line Form



3.3.1.3 Performance

As in the case of analyzing the user interface, the best way to evaluate the performance of a computer program is through program testing. Unfortunately, Navigator could not be tested for performance, because a working copy of Navigator Version 1.0 for X Window could not be obtained. Using other versions of Navigator would have yielded inaccurate results because later versions embodied additional enhancements in performance. Instead, numerous articles and texts were consulted for reference. Fortunately, many of these articles and texts compare the performance of Navigator, at the time of its introduction, with those of existing Web browsers.

As the most important indicator of performance, the dimension of speed refers to the time it takes for a computer program to perform a function or set of functions. Interestingly, speed was a point of weakness for many of the early Web browsers including Mosaic (Vaughan-Nichols 183). Recognizing a growing demand for fast Web browsers, designers of Navigator placed great emphasis on program speed. As a result, Navigator Version 1.0, at the time of its introduction, was significantly faster than any existing Web interface. In Inside the World-Wide Web, Vaughan-Nichols et al. write: "Where Netscape (Navigator) really kills the competition is speed" (195). In fact, Navigator outperformed all other interfaces when it comes to Internet access and document processing.

According to Netscape press releases, Navigator was "optimized to run smoothly over 14.4 kilobit/second modems as well as higher bandwidth lines, delivering performance at least ten time that of other network browsers" (Netscape Communications Corp., "Netscape Press Release 10/13/94"). One optimization used in Navigator is loading all of the text before retrieving the images (Vaughan-Nichols 182). Unlike Mosaic, which attempts to download all of the graphics first, Navigator retrieves the text first then proceeds to retrieve the images. This simple scheme enables Navigator to display the text of a doc-

ument first and then load the images one by one, giving the illusion of faster access. In addition, Navigator supports multiple network accesses to the Internet, enabling users to download several documents or images simultaneously (Netscape Communications Corp., "Netscape Press Release 10/13/94"). Through these optimizations, Navigator offered significant improvements in performance when accessing the Internet. The improvements were particularly noticeable with systems utilizing slow modem connections (Ernst 21).

In addition to offering faster Internet access, Navigator also outperformed all of the existing Web interfaces when it comes to processing and displaying documents (Vaughan-Nichols 195). Navigator's superior performance can be largely attributed to four enhancements. One enhancement is Navigator's ability to display information as soon as it arrives, allowing users to view a document piece by piece rather than waiting for the entire document to be loaded (Vaughan-Nichols 183).

In addition, Navigator supports continuous document streaming, which enables users to interact with documents as they are being downloaded (Netscape Communications Corp., "Netscape Press Release 10/13/94"). With this enhancement, the user can click on a displayed link and initiate a hypermedia jump well before the document has finished loading. Combined with the optimizations for accessing the Internet, continuous document streaming has proven to be a significant time saver.

Another enhancement is the use of a sophisticated cache manager, which enables Navigator to cache both images and documents for future access (Ernst 92-94). The cache manager supported a two-tier caching scheme consisting of a primary cache area in main memory and a larger secondary cache area on hard disk.

Finally, Navigator processes and displays graphics faster than any existing Web browser including Mosaic (Vaughan-Nichols 183). One reason is that Navigator, unlike other browsers, provides native support for the JPEG format. JPEG is one of the most, if

not the most, popular graphics format on the Web: It is widely used in HTML documents containing inline graphics. Another reason is that Navigator can display GIF graphics very quickly. GIF is probably the second most popular graphics formats on the Web. In summary, all of these enhancements added up to a significant performance edge for Navigator when it comes to processing documents.

According to the available literature, there were no discernible differences in performance between Navigator and existing Web interfaces when it comes to processing hypermedia links. Like other Web interfaces, Navigator uses an underlining/highlighting scheme for locating links and a point-and-click mechanism for activating them. Similarly, Navigator did not offer a significant performance advantage when it comes to processing interactive forms.

3.3.2 State of the Art Analysis

In analyzing the state of the art prior to the introduction of Navigator, the following articles and texts were consulted: Inside the World Wide Web and documents downloaded from the Web sites of NCSA, Spyglass, Inc., and RSA Data Security, Inc. In addition, a patent search was conducted using LEXIS. As with the rest of the case study, the state of the art analysis was based on Navigator Version 1.0 for X Window, which was officially released in December of 1994. The results of the analysis are presented in this subsection.

3.3.2.1 Web Browsers

During the year and a half between the introduction of Mosaic and the release of Navigator, the Web market began to mature into a thriving industry, with Mosaic leading all other browsers in functionality and popularity. In December of 1993, the NCSA released Version 2.0 of Mosaic, which embodied the following improvements in functionality and performance (NCSA, "NCSA Mosaic for the X Window System 2.0"):

1) Support for on-line fill-out forms

2) Support for user authentication

3) Direct WAIS access

4) Enhanced inlined image caching, including customizability of cache size

5) Support for delayed image loading

6) Performance speedups in displaying hypermedia documents

7) Improved system resource management

Due to these improvements and the growing interest in the Internet, the release of Version 2.0 greatly increased the popularity of Mosaic.

Although Mosaic's increased popularity brought acclaim and visibility to the NCSA, there were also drawbacks. Because the NCSA was originally founded to provide super-computing resources to scientists and researchers, it was ill-equipped and understaffed to provide technical assistance to thousands of individual and corporate users of Mosaic (Vaughan-Nichols 23). In order to lessen the burden on its resources and staff, the NCSA began to give several companies (e.g. Mosaic Communications, Spry, and Quarterdeck) the right to develop commercial versions of Mosaic, hoping that these companies would shoulder the responsibility of supporting Mosaic users. Unfortunately, the NCSA was equally under-prepared to handle the growing demand for commercial licenses and the complexities involved in licensing (Vaughan-Nichols 23).

Finally, in August of 1994, the NCSA entered a long-term agreement with Spyglass Inc., which gave Spyglass all future commercial licensing rights to Mosaic as well as the responsibilities involved in supporting and licensing Mosaic (Vaughan-Nichols 23). By October of 1994, Spyglass had distributed 10 million copies of its Enhanced NCSA Mosa-

ic, a commercial caliber version of Mosaic first introduced in May of 1994 (Spyglass, Inc., "Spyglass Newsflash 10/17/94"). Enhanced NCSA Mosaic Release 1.0 featured a number of improvements and enhancements including the following (Spyglass, Inc., "Spyglass Newsflash 8/24/94"):

1) Increased reliability

2) Feature-matched versions across all three desktop platforms

3) Dramatic improvement in performance

4) User interface improvements

5) Reduced memory requirements

6) Easy installation

7) Support for multiple windows

During the same time period, Thomas R. Bruce of the Cornell University Legal Information Institute introduced a competing Web browser called "Cello." Unlike most of the commercial browsers available at the time of its introduction, Cello does not utilize any portion of Mosaic's source code. As a result, it looks and feels very different from Mosaic (Vaughan-Nichols 186). For instance, Cello does not provide toolbar buttons; instead, all of the commands are accessed through pull-down menus. Cello also uses underscoring rather than highlighting to indicate hypermedia links.

Apart from its unique interface, however, Cello offered nothing new in terms of functionality or performance (Vaughan-Nichols 186). Like all Web browsers, Cello provides native support for HTTP and HTML. It also offers Telnet, electronic mail, and FTP access through pull-down menus--all standard features in most Web browsers. In addition, Cello offered average, if not below-average, performance in accessing the Internet and process-

ing hypermedia documents (Vaughan-Nichols 187). In particular, Cello does not render graphics well; as a result, many HTML pages appear crude and jagged with Cello. On the positive side, however, Cello does require less resources than its Mosaic-based counterparts. It can be configured to run on machines with as little as 2MB of RAM, whereas others require at least 4MB.

It is important to note that none of the Web browsers available during this time--including Cello, Mosaic Version 2.0, and Enhanced NCSA Mosaic Release 1.0--incorporates any security features. With all of these browsers, incoming and outgoing data are transmitted in the clear without the option of data encryption. As a result, conducting commercial transactions on the Web remained dangerous and unreliable.

3.3.2.2 Data Encryption

Long before the birth of the World Wide Web, the technology of data encryption--generally referred to as "cryptography"--had been researched in depth and used in a wide range of applications. Originally developed as a branch of applied mathematics, cryptography was quickly adopted by organizations and corporations requiring a secure environment for conducting highly sensitive transactions. Promoted and funded by various sectors of the banking and defense industries, researchers began to develop highly reliable and efficient systems for conducting secure transactions.

Historically speaking, much of the pioneering research in cryptography was conducted at MIT and Stanford University (RSA Data Security, Inc., "RSA Press Release 3/4/96"). The research at these institutions led to the following four landmark patents:

1) 4,200,770 Cryptographic Apparatus and Method

2) 4,218,582 Public Key Cryptographic Apparatus

3) 4,405,829 Cryptographic Communications System and Method

4) 4,424,414 Exponential Cryptographic Apparatus

These patents are significant in that they cover all known methods of practicing public key encryption, a cryptographic technique based on pairs of asymmetric keys. Public key encryption can be used to design highly reliable and efficient systems for data security and authentication.

The licensing rights to the above four patents were initially held by Public Key Partners (PKP). Over the years, PKP proceeded to license its patents to a number of companies including Cylink and RSA Data Security, Inc., which was founded by the inventors of the 4,405,829 patent in 1982 (RSA Data Security, Inc., "RSA Press Release 3/4/96").

By December of 1994, RSA Data Security, Inc. had become the world's largest and most recognized company for cryptography (RSA Data Security, Inc., "RSA Press Release 3/4/96"). At the time, RSA encryption and authentication technologies were being installed and used by millions worldwide (RSA Data Security, Inc., "RSA Press Release 3/4/96"). As a testament to its popularity, RSA's encryption technology is embedded in such diverse products as Microsoft Windows, Intuit's Quicken, and Lotus Notes. None of the products prior to Navigator, however, combined cryptography with hypermedia links and Internet access.

A search of the patent database using LEXIS yielded 152 patents related to cryptography. A list of these patents is included in Appendix A.2. None of these patents covers a method or apparatus for combining cryptographic techniques with hypermedia links and Internet access.

3.3.3 Points of Novelty Embodied in Navigator

Considering the state of the art prior to its introduction, Navigator Version 1.0 for X Window embodied the following points of novelty:

1) Novelty of functionality in enabling transparent encryption of data transfers across the Internet that have been initiated using hypermedia links

   Navigator was the first Web browser to incorporate security features. Navigator enables users to encrypt incoming and outgoing data transfers with a remote server, making commercial transactions on the Internet secure and reliable.

2) Novelty of performance in accessing the Internet through hypermedia links

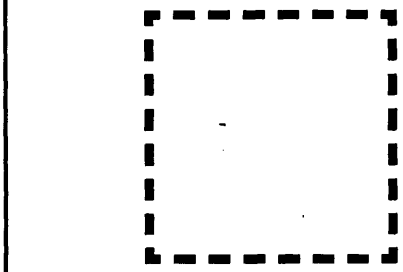
   Through various optimizations, Navigator offered significant improvements in performance when retrieving hypermedia documents from remote servers on the Internet.

3) Novelty of performance in processing hypermedia documents retrieved across the Internet

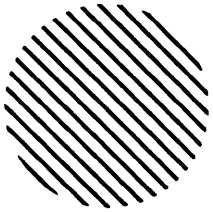
   Through various enhancements, Navigator also offered significant improvements in performance when processing and displaying hypermedia documents retrieved from remote servers on the Internet.

Each of the above points of novelty is depicted using modified versions of the data-flow graphs generated earlier: Graphs N, N.1, N.1.1, N.2, N.3, and N.4. In these graphs, novelty of functionality is represented by dashed lines, and novelty of user interface and novelty of performance are represented by different patterns of shading.

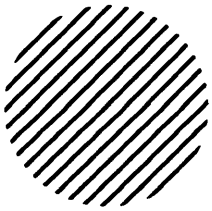
Figure 3.26: Notations Used in the Graphs of Novelty



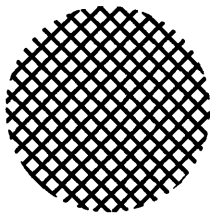
Novelty of Functionality



Novelty of User Interface



Novelty of Performance



Novelty of User Interface and Performance

Figure 3.27: Graph N (Modified): Overview of Navigator's Functionality

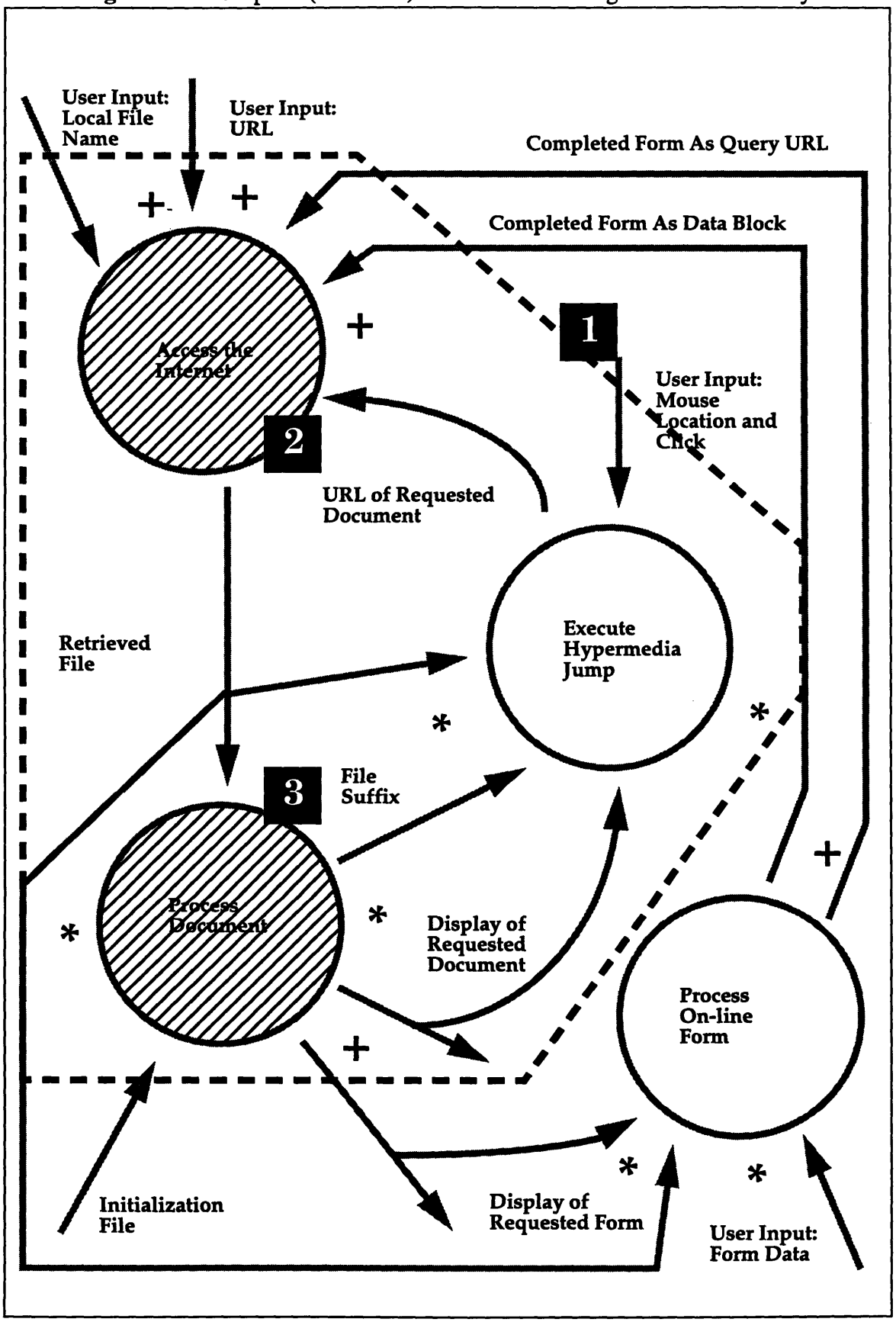


Figure 3.28: Graph N.1 (Modified): Access the Internet

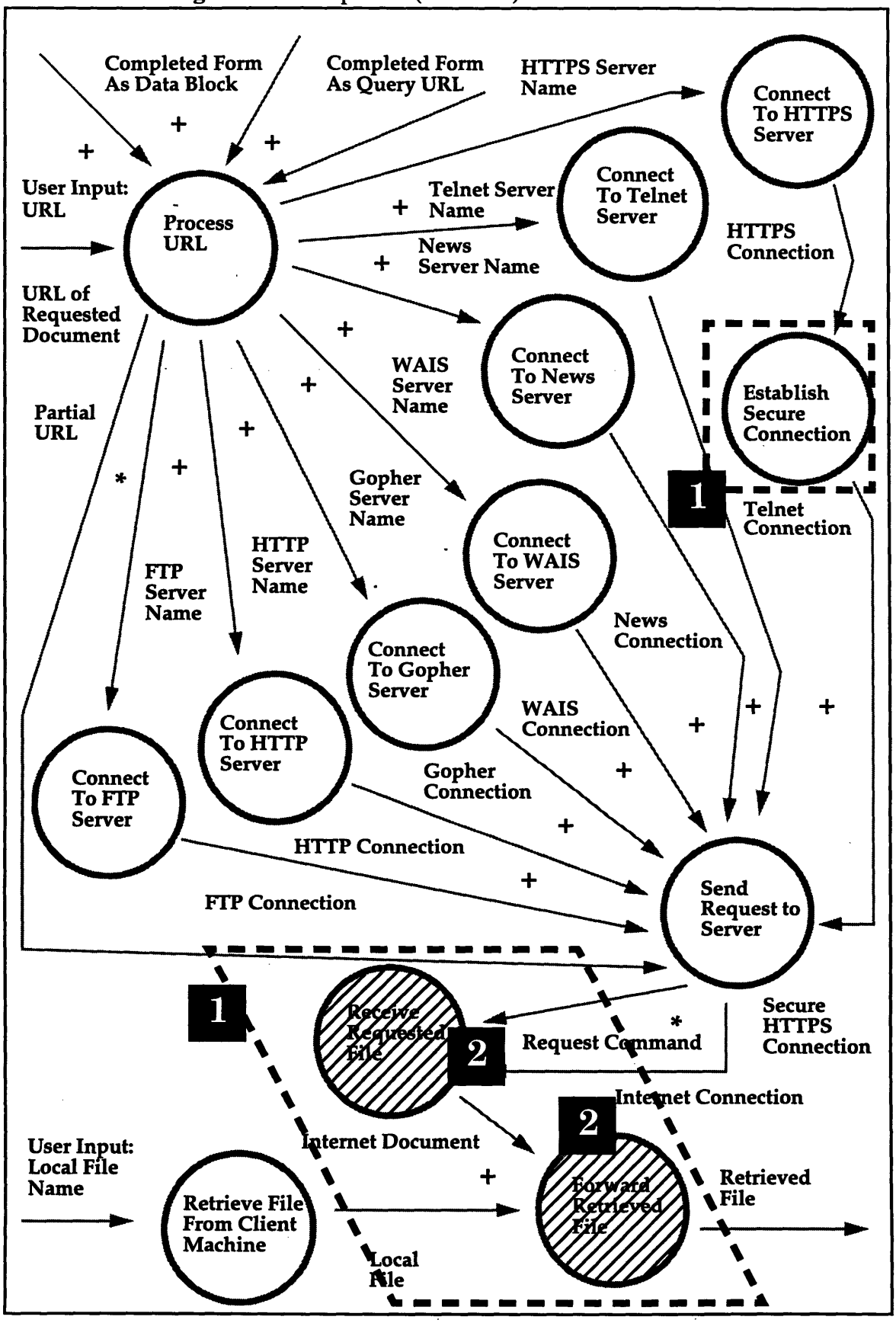


Figure 3.29: Graph N.1.1 (Modified): Establish Secure Connection

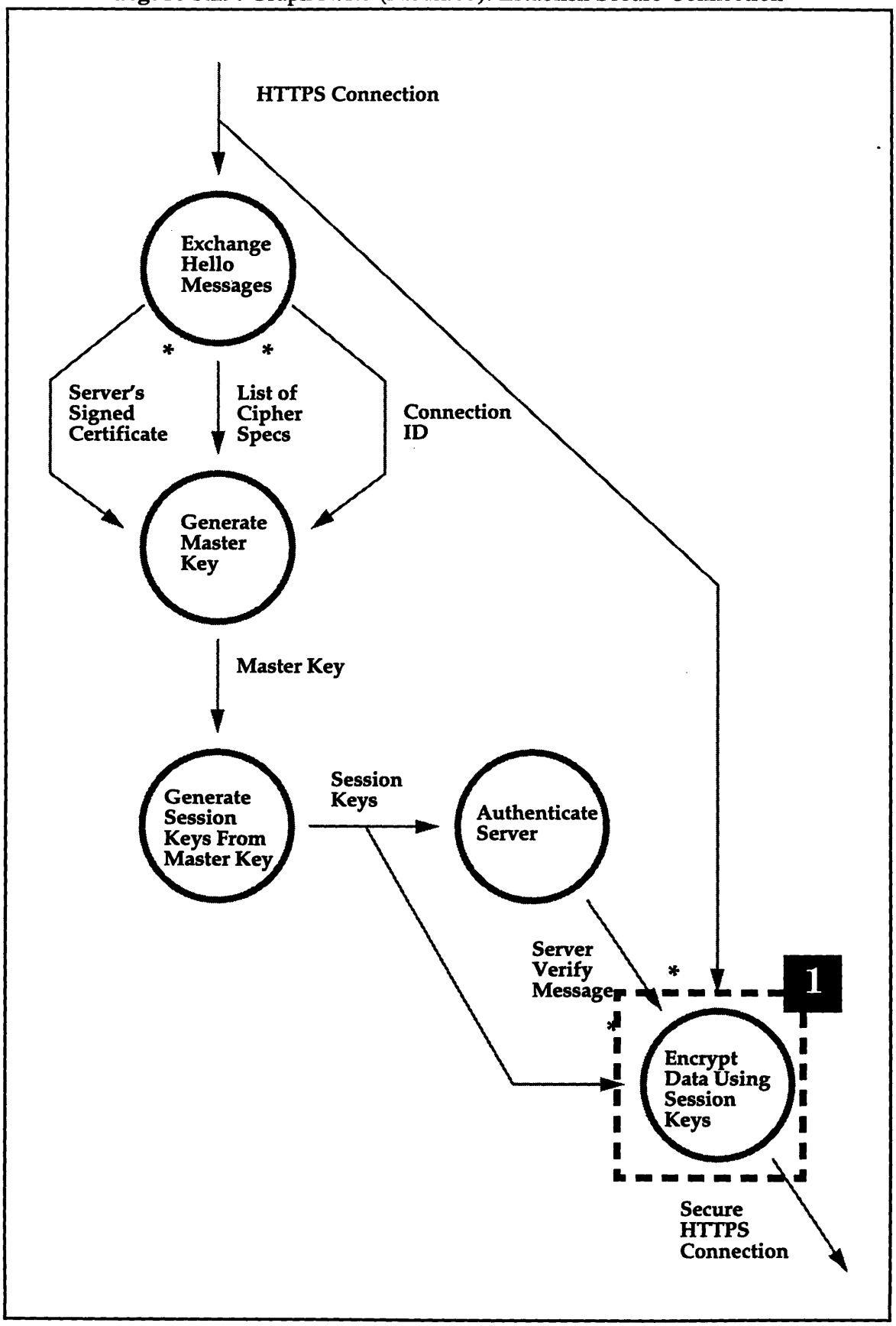


Figure 3.30: Graph N.2 (Modified): Process Document

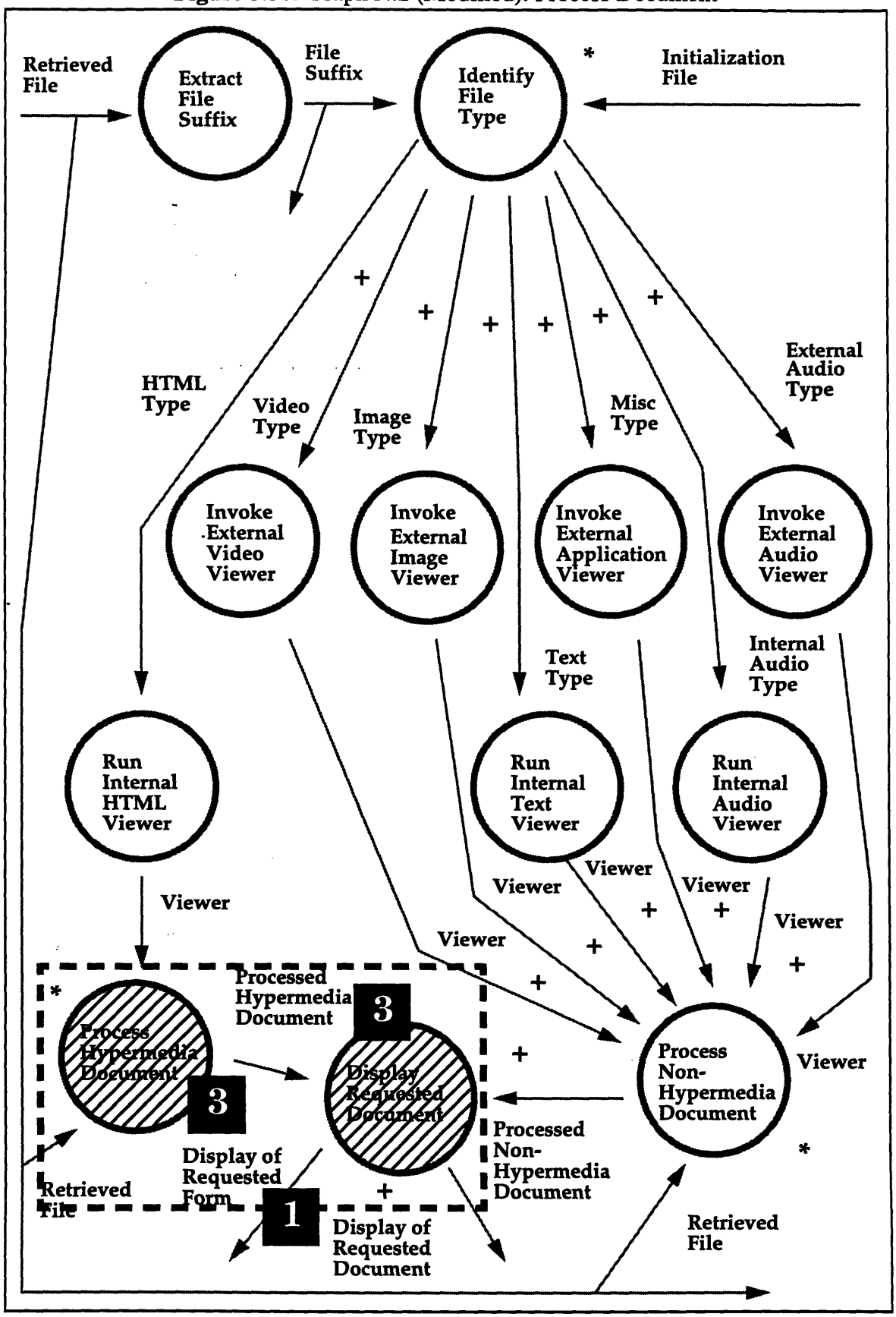


Figure 3.31: Graph N.3 (Modified): Execute Hypermedia Jump

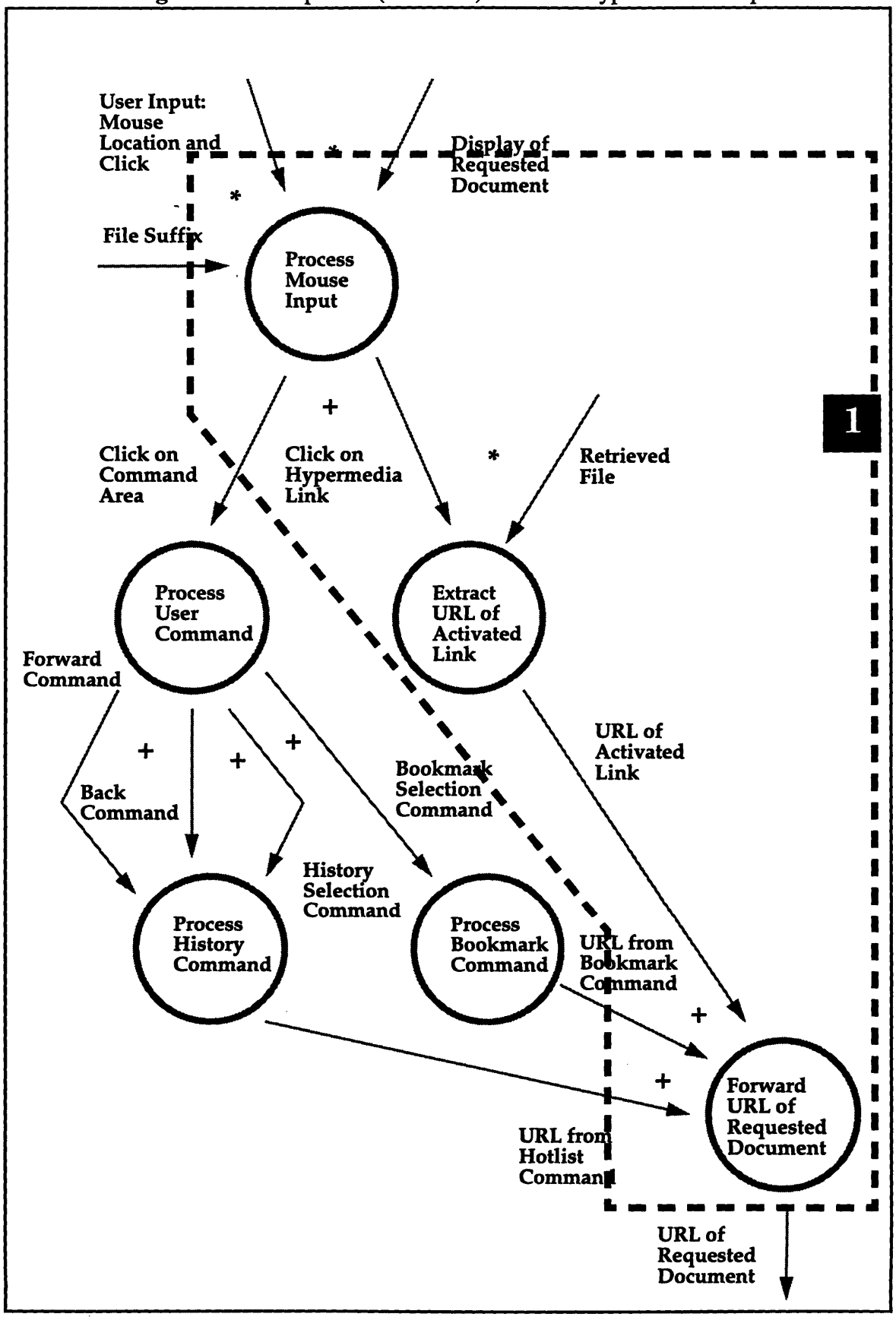
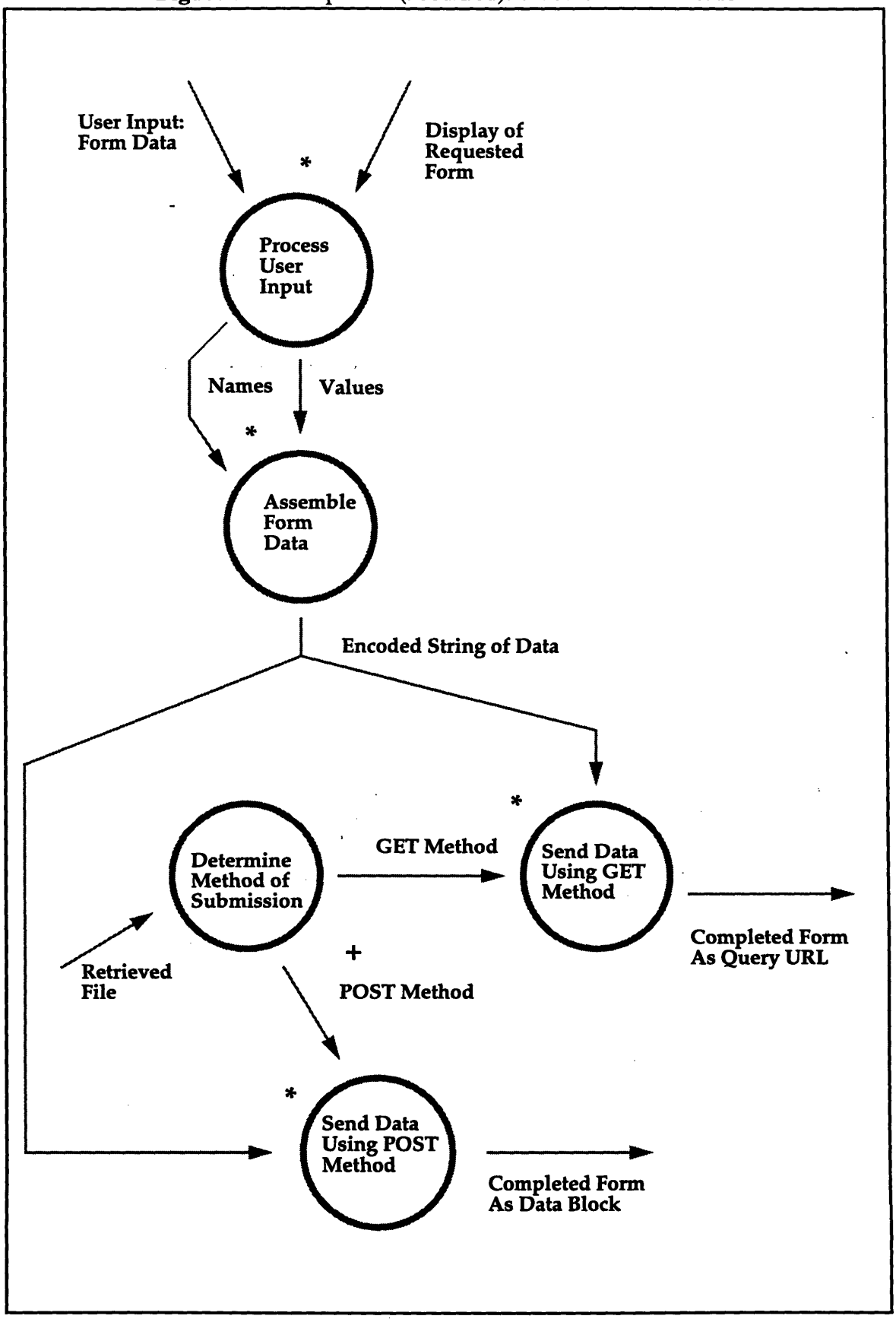


Figure 3.32: Graph N.4 (Modified): Process On-line Form



3.3.4 Commercial Significance Analysis

Following the state of the art analysis, each point of novelty embodied in Navigator was analyzed for its commercial significance. The following texts were consulted: Using Netscape and Inside the World Wide Web.

Since its introduction, Navigator has dominated the Web browser market--capturing over 80 percent of the market share. In Inside the World Wide Web, Vaughan-Nichols et al. write: "The future of Web browsing appears to be Netscape [Navigator], and with a product this good, that's great news" (196). As with Mosaic, the tremendous success of Navigator can be attributed to a number of factors, including the fact that Navigator was distributed for free to individual users and nonprofit organizations. The most important factor, however, may be that Navigator does so well what Mosaic fails or struggles to do.

First, Navigator was much faster than Mosaic when it comes to accessing the Internet or processing retrieved documents. According to Warren Ernst in Using Navigator, Mosaic "was so slow that it wasn't practical unless the computer it was running on was directly wired into the Internet" (20). The poor performance of Mosaic greatly limited its commercial potential because most home and business users were accessing the Internet through dialup modems, which are significantly slower than direct Internet connections using T1 or T3 lines. As a result, these users would encounter frustrating delays when retrieving and displaying Web documents containing large images or long sound bytes. Navigator, on the other hand, was "optimized to run smoothly over 14.4 kilobit/second modems as well as higher bandwidth lines," allowing home and business users to "surf" the Internet with speed (Netscape Communications Corp., "Netscape Press Release 10/13/94"). Thus, Navigator introduced the average computer user, not just the research scientist, to the true power of the World Wide Web: the power to locate and retrieve information stored anywhere in the world--all in the blink of an eye.

In addition to increased speed, Navigator also commercialized the Internet by providing the ability to establish secure connections. Navigator was the first Web browser to offer transparent support for securing Internet connections as well as authenticating servers. Prior to the advent of Navigator, Web browsers offered two choices to their users when it comes to conducting commercial transactions on the Internet: (1) send sensitive information in the clear and hope that no one is monitoring the connection; or (2) encrypt the information manually and hope that the receiving party is who she claims to be. Clearly, both of these options are undesirable. As an alternative, Navigator offered a reliable way of securing Internet connections without burdening the user: Both data encryption and server authentication are done in the background (Ernst 11). With the growing number of companies offering products and services on the Internet, Navigator's security features proved to be a key selling point.

Based on the above observations, it appears that all three points of novelty embodied in Navigator contributed directly to the increased use and sale of Navigator. Thus, they satisfy the standard of commercial significance, as defined in Chapter 2.

3.3.5 Behavioral Innovations Embodied in Navigator

To summarize, Navigator Version 1.0 for X Window embodied the following behavioral innovations:

1) Novelty of functionality in enabling transparent encryption of data transfers across the Internet that have been initiated using hypermedia links

2) Novelty of performance in accessing the Internet through hypermedia links

3) Novelty of performance in processing hypermedia documents retrieved across the Internet

As demonstrated by the preceding analysis, all of the above innovations represent an ad-

vance in the state of the art that produces a novel and useful result of commercial significance.

Chapter 4

Conclusion

4.1 Discussion of the Results

The following discussion is organized around three sets of questions. First, does the proposed scheme for identifying behavioral innovations satisfy its design goals and principles? Second, is it useful in supporting anti-cloning measures? Finally, how well does it support existing and proposed schemes for describing computer programs?

4.1.1 Observations on the Feasibility of the Proposed Scheme for Identifying Behavioral Innovations

The three-stage scheme proposed in this study represents a systematic way of identifying innovations in program behavior. In order to evaluate the feasibility of this scheme, I conducted a case study of two Web browsers: NCSA Mosaic and Netscape Navigator. The results of the case study support the following conclusions:

First, the proposed scheme can be used to capture the behavioral innovations of a computer program without referring to its internal design. For both Mosaic and Navigator, I was able to construct detailed data-flow graphs of their functionality based solely on user manuals and program testing. Similarly, I was able to analyze their user interface and performance based strictly on program testing and the existing literature. It was not necessary to consult their source codes or other design documents.

Second, the proposed scheme provides a highly consistent way of identifying behavioral innovations. For both Mosaic and Navigator, I was able to follow the same analytical procedure: (1) Describe the program behavior using data-flow graphs and snapshots of screen displays; (2) filter the novel elements of program behavior from existing technolo-

gy; and (3) analyze each novel element for its commercial significance. In both cases, I was able to generate a list of protectable innovations in program behavior.

Finally, the proposed scheme supports a technically coherent view of behavioral innovations. Instead of relying on broad generalizations or marketing slogans, each stage of the proposed scheme requires an in-depth investigation of the underlying technology. As a case in point, many critics have labelled both Mosaic and Navigator as "innovative" software without articulating their innovative elements in precise and technical terms. Consequently, it is difficult to determine if Mosaic or Navigator should have been legally protected from behavioral clones; and if so, whether the patent regime or a market-oriented regime should have been used. As noted in the Introduction, merely labelling a program as "innovative" says nothing about the type of legal protection it may or may not require. In contrast, under the proposed scheme, I was able to identify the behavioral innovations embodied in Mosaic and Navigator and to describe these innovations in a technically coherent manner. As a result, it is now possible to consider the actual merits and drawbacks of providing anti-cloning protection to Mosaic or Navigator.

In summary, the Mosaic-Navigator case study has demonstrated the feasibility of implementing the proposed scheme for identifying innovations in program behavior. If adopted, the proposed scheme would provide a consistent and technically coherent framework for identifying the behavioral innovations embodied in computer programs without referring to their internal design elements. As stated earlier in the Assumptions and Limitations section, more test cases are needed to validate this finding.

4.1.2 Observations on the Usefulness of the Proposed Scheme for Identifying Behavioral Innovations

The primary motivation for this study was the current debate on the lack of anti-cloning protection within the software industry. Given the rising demand for increased protec-

tion, policymakers are likely to consider and possibly implement anti-cloning measures in the near future. It remains uncertain, however, whether the patent regime or a new market-oriented regime should serve as the framework for such measures.

Underlying this larger question are two related issues that must be adequately explored before implementing anti-cloning measures: (1) how to identify behavioral innovations embodied in computer programs, and (2) how to identify behavioral clones in the marketplace. The first issue was explicitly addressed by this study and led to the development of the proposed scheme for identifying innovations in program behavior. The second issue, however, was only realized during the course of this study.

In some cases, market-destructive clones are easy to spot. For example, in the 1980s, Paperback Software introduced a behavioral clone of the popular Lotus 1-2-3 spreadsheet called "VP-Planner" (Samuelson et al. 2397). What is interesting is that Paperback chose to promote its VP-Planner as a "work-alike" of Lotus 1-2-3, claiming that the two products were indistinguishable in their behavior. In effect, Paperback was admitting that its product was a clone (Samuelson et al. 2397). In other cases, however, it is difficult to determine whether a given product represents a potentially market-destructive clone or a legitimate competitor. Furthermore, some clones, known as "follow-on innovations," embody behavioral innovations of their own and thus may merit legal protection.

The above observations underscore the need for a systematic method of distinguishing market-destructive clones from legitimate competitors and follow-on innovations. Incidentally, the proposed scheme for identifying behavioral innovations provides an excellent starting point for developing such a method.

As a case in point, consider the case study on Mosaic and Navigator Web browsers. In the course of identifying their behavioral innovations, I generated two sets of data-flow

graphs representing the points of novelty embodied in each program. These graphs are illustrated in Figures 3.9-3.13 and Figures 3.26-3.32.

By comparing the two sets of data-flow graphs, it is possible to determine whether Navigator was a potentially market-destructive clone of Mosaic or its legitimate competitor. First, consider the first set of data-flow graphs corresponding to Mosaic (i.e. Figures 3.9-3.13). In these graphs, I was able to represent three distinct points of novelty: (1) novelty of functionality in enabling retrieval and display of hypermedia documents across the Internet; (2) novelty of user interface in displaying documents retrieved from the Internet; and (3) novelty of functionality in supporting hypermedia links across the Internet. Because all of the above elements contributed directly to the increased use of Mosaic, they can also be considered as protectable innovations.

Next, consider the second set of data-flow graphs corresponding to Navigator (i.e. Figures 3.26-3.32). Upon examination, it is apparent that all three of the behavioral innovations embodied in Mosaic are incorporated by Navigator. In fact, Navigator offers the same combination of functionality as Mosaic when it comes to retrieving hypermedia documents and supporting hypermedia links across the Internet. Furthermore, Navigator uses a user interface that is almost identical to the one used by Mosaic. In light of these considerations, it is safe to conclude to that Navigator achieves behavioral equivalence with Mosaic and thus should be regarded as a clone.

Upon further examination, however, it is equally clear that Navigator incorporates behavioral innovations of its own. As noted in Section 3.3.4, Navigator does so well what Mosaic fails or struggles to do. In fact, at the time of its introduction, Navigator was not only faster than Mosaic but also made commercial transactions on the Internet safe and reliable. Through increased performance and added functionality, Navigator raised the levels of competition and innovation within the Web browser market.

Consequently, it would be unfair to characterize Navigator as a mere clone of Mosaic; instead, Navigator should be classified as a valuable follow-on innovation. Whether anti-cloning measures should have been applied to Navigator is a separate question, best left to policymakers. Nevertheless, it is clear that the behavioral innovations embodied in both Mosaic and Navigator promote consumer welfare and thus lie within the scope of legal protection.

In conclusion, the preceding analysis suggests that the proposed scheme is not only useful in identifying innovations in program behavior, but also in distinguishing market-destructive clones from legitimate competitors and follow-on innovations. As stated earlier, more test cases are needed to validate this finding.

4.1.3 Comments on Describing Behavioral Innovations

Although the primary focus of this study was on identifying behavioral innovations in software, I also sought to address the issue of describing those innovations, once they have been identified. The issue of describing behavioral innovations is important because intellectual property regimes, both existing and proposed, must incorporate a scheme for describing the objects of protection. Without such a scheme, it is impossible to define the terms and scope of the protection being sought: Just as a deed circumscribes the property rights of a landowner, a descriptive scheme is necessary to delineate the protection rights of an author, inventor, or an innovator--as the case may be.

In Section 2.3, I presented a critical survey of the descriptive schemes used in the copyright and patent regimes. In brief, I concluded that none of the existing schemes is especially well-suited for describing behavioral innovations in program behavior: The copyright scheme discloses too little information, whereas the patent scheme discloses too much. I am not suggesting that copyright and patent regimes cannot be used to protect be-

havioral innovations in software--only that the descriptive schemes used in these regimes must be modified in light of the special properties of computer programs and the innovative nature of the software industry.

In Section 2.4.1, I offered a sketch of a prototype scheme for describing behavioral innovations in software, to be used under a market-oriented regime. Designed with computer programs in mind, the suggested scheme offers a number of advantages over existing schemes, including compatibility with trade secrecy protection. In addition, it retains the strengths of existing schemes by incorporating many of the terms and formats used in the patent regime and by supporting a fast examination process similar to the one used in the copyright regime.

Whether existing regimes or a new market-oriented regime is to be used, I believe that the proposed scheme for identifying innovations in program behavior contains a number of useful insights and mechanisms for describing computer programs. First, the proposed scheme correctly emphasizes the behavioral aspects of computer programs. Likewise, descriptive schemes should avoid analyzing computer programs as purely textual works. Second, the proposed scheme introduces data-flow graphs as a mechanism for capturing both the functionality and novelty of computer programs: See Figures 3.13-3.19 and Figures 3.26-3.32. Descriptive schemes should utilize similar graphs in order to represent behavioral innovations in a consistent and technically coherent manner. Finally, the proposed scheme places heavy emphasis on analyzing the state of the art prior to the introduction of a computer program. As noted earlier, existing databases cover only a small fraction of all the behavioral innovations embodied in computer programs. Accordingly, instead of relying strictly on patent or copyright searches, descriptive schemes should require a more in-depth assessment of the marketplace.

4.2 Suggestions for Further Research

On January 16, 1996, the Supreme Court of the United States rendered an evenly-divided decision in the case of *Lotus Development Corp. vs. Borland International, Inc.* (Barrett). By upholding the earlier ruling of a federal appeals court in Boston, the justices effectively denied copyright protection to the "menu command hierarchy" of computer programs. Most modern computer programs provide a hierarchy of menus from which users can select commands to access program features. This system of menus constitutes an integral part of the user interface and an important component of program behavior.

In the *Lotus v. Borland* case, Lotus claimed that its copyright in Lotus 1-2-3 was infringed by Borland when Borland copied the menu command hierarchy of Lotus 1-2-3 into its Quattro and Quattro Pro spreadsheets (Lotus Development Corp. v. Borland, Inc.). In doing so, Borland did not copy any of Lotus's source or object code; it only imitated the words and organization of Lotus's menu command hierarchy. Interestingly, both Quattro and Quattro Pro offered a native user interface, which was completely different than the one used in Lotus 1-2-3. Users could only achieve Lotus compatibility by invoking an alternate user interface called "Lotus Emulation Interface": With it, users could interact with Quattro or Quattro Pro as if they were using Lotus 1-2-3. As noted by the appeals court, "In effect, Borland allowed users to choose how they wanted to communicate with Borland's spreadsheet programs: either by using menu commands designed by Borland, or by using the commands and command structure used in Lotus 1-2-3 augmented by Borland-added commands" (Lotus Development Corp. v. Borland, Inc.).

In response, Borland countered that the menu command hierarchy of Lotus 1-2-3 constituted a "method of operation," akin to the buttons on a videocassette player (Barrett). By statutory definition, methods of operation lie beyond the scope of copyright protection.

Thus, according to Borland, its Quattro and Quattro Pro spreadsheets did not infringe Lotus's copyright in Lotus 1-2-3.

The stakes were very high for both sides: Lotus was asking for more than $100 million in damages from Borland (Barrett). Apart from the immediate concerns of the parties involved, the future of intellectual property protection was also at stake (Marshall). The basic question was whether copyright law should be used to protect elements of program behavior--in this case, the menu command hierarchy--from unauthorized cloning by a second comer. The federal appeals court ruled in the negative: Copyright protection should be limited to the textual elements of a computer program. By a 4-to-4 vote, the Supreme Court concurred.

According to most legal experts, the Supreme Court ruling will not have any binding effect on future cases because it was not accompanied by an opinion (Barrett). Furthermore, the evenness of the vote lends little weight to the ruling. Neal Goldman, one of the attorneys for Lotus, commented, "Intellectual property lawyers are disappointed at not having more guidance" (Marshall).

Beyond the Supreme Court ruling, some critics, including Samuelson et al., argue that Quattro, as a "near-clone" of Lotus 1-2-3, should have been regulated for a market-preserving period (Samuelson et al. 2397). Others, including Kenneth Arrow--an economics professor at Stanford University and a Nobel laureate, contend that follow-on innovations like Quattro should be encouraged rather than regulated in order to avoid monopoly effects in the software market (Marshall).

In many respects, the *Lotus v. Borland* case presents an excellent opportunity to evaluate the findings of this study. First, the case has been scrutinized for years by legal critics and industry observers alike. As a result, the volume of existing literature on this case is considerable. Furthermore, the large number of amicus curiae briefs suggest that the is-

sues underlying this case have great significance to both the software industry and the legal community. Most importantly, the facts of the case parallel the situation in the Mosaic-Navigator case study. Both Mosaic and Lotus 1-2-3 were pioneer products, giving rise to thriving markets noted for their high levels of innovation and competition. Similarly, both Navigator and Quattro were follow-on innovations--building on their predecessors and, at the same time, advancing the state of the art. According to Borland, "[f]rom the time of its initial release... Quattro included enormous innovations over competing spreadsheet products" (Lotus Development Corp. v. Borland, Inc.).

For all of the above reasons, it would be interesting and instructive to apply the proposed scheme for identifying behavioral innovations to the *Lotus v. Borland* case. It would then be possible to determine whether the proposed scheme can be generalized to include computer programs other than Web browsers. It would also be possible to determine whether the proposed scheme offers additional clarity to an actual legal dispute involving intellectual property claims.

Other possible test cases include the *Whelan Associates v. Jaslow Dental Laboratory* case, which revolved around the unauthorized appropriation of a computer program's information flow (Samuelson et al 2359), and the ongoing series of lawsuits filed by Apple Computers over its windows-based operating system (Samuelson et al. 2410). All of these test cases represent opportunities to validate and generalize the findings of this study.

Appendix A

Patent Search Results

A.1 Patents Related to Hypertext or Hypermedia Technology

Patent No.	*Title*
5,159,669	Automatically creating a second workspace operation record including history data and a unit ID based on a first workspace operation
4,905,163	Intelligent optical navigator dynamic information presentation and navigation system
4,974,173	Small-scale workspace representations indicating activities by other users
5,089,956	Method of distributing related documents to be identified end users in an information processing system
5,048,869	Hypertext book attachment
4,982,344	Accelerating link creation
5,101,345	Method of filing stapled documents with a staple relationship involving one or more application programs
5,251,294	Accessing, assembling, and using bodies of information
5,408,.655	User interface system and method for traversing database
5,204,947	Application independent (open) hypermedia enablement service
5,297,249	Hypermedia link marker abstract and search services
5,355,472	System for substituting tags for non-editable data sets in hypertext documents and updating web files containing links between data sets corresponding to changes made to the tags
5,367,621	Data processing method to provide a generalized link from a reference point in an on-line book to an arbitrary multimedia object which can be dynamically updated
5,107,443	Private regions within a shared workspace
5,101,364	Method and facility for dynamic video composition and viewing
5,412,774	Apparatus for and method of displaying a data item of a database using the display function of a selected data item
5,404,534	Method and apparatus for extensible object-oriented inter-application link management
5,446,891	System for adjusting hypertext links with weighed user goals and activities
5,179,718	Method of filing having a directed relationship through defining a staple relationship within the context of a folder document
5,408,599	Editing apparatus for simultaneously editing two different types of data with a single edit command
5,442,786	Method for recording user interaction with a computer database to generate reports
5,380,043	Hypertext book attachment
5,428,740	Applying successive data group operations to an active data group
5,379,375	Automatic location of screen objects through the use of relational position

	data
5,428,731	Interactive multimedia delivery engine
5,418,948	Concept matching of natural language queries with a database of document concepts
5,418,908	System for automatically establishing a link between an electronic mail item and a remotely stored reference through a place mark inserted into the item
5,367,619	Electronic data entry system employing an expert system to facilitate generation of electronic data forms with complex interrelationships between fields and subforms
5,239,617	Method and apparatus providing an intelligent help explanation paradigm paralleling computer user activity
5,206,951	Integration of data between typed objects by mutual, direct invocation between object managers corresponding to object types
5,265,065	Method and apparatus for information retrieval from a database by replacing domain specific stemmed phases in a natural language to create a search query
5,283,864	Computer apparatus and method for graphical flip book
5,313,646	Method and apparatus for translucent file system
5,315,697	Method for linking program execution status information
5,316,341	Hypertext book attachment
5,339,390	Operating a processor to display a stretched continuation of a workspace
5,339,391	Computer display unit with attribute enhanced scroll bar
5,008,843	Representation of collaborative multi-user activities relative to shared structured data objects in a networked workstation environment
5,058,008	Mail system with personalized training for users
4,914,586	Garbage collector for hypermedia systems
4,974,173	Small-scale workspace representations indicating activities by other users
5,333,237	Hypermedia structured knowledge base system
5,421,012	Multitasking computer system for integrating the operation of different application programs which manipulate data objects of different types
5,444,823	Intelligent search engine for associated on-line documentation having questionless case-based knowledge base
5,440,678	Method of and apparatus for creating a multi-media footnote
5,220,657	Updating local copy of shared data in a collaborative system
5,185,699	Method and apparatus for maintaining plant operation procedures
5,200,893	Computer aided text generation method and system
5,345,548	Character display apparatus for displaying multiple-font characters with high speed

A.2 Patents Related to Cryptography

Patent No.	*Title*
4,238,853	Cryptographic communication security for single domain networks
4,228,321	Privacy transmission system with remote key control
4,218,738	Method for authenticating the identity of a user of an information system
4,218,582	Public key cryptographic apparatus and method

4,206,315	Digital signature system and apparatus
4,203,166	Cryptographic file security for multiple domain networks
4,200,770	Cryptographic apparatus and method
4,193,131	Cryptographic verification of operational keys used in communication networks
3,962,539	Product block cipher system for data security
3,958,081	Block cipher system for data security
4,107,458	Cipher computer and cryptographic system
4,558,176	Computer systems to inhibit unauthorized copying, unauthorized usage, and automated cracking of protected software
4,549,308	Secure mobile radio telephony
4,525,844	Method for interchanging n partial bands
4,514,592	Cryptosystem
4,438,824	Apparatus and method for cryptographic identity verification
4,424,414	Exponentiation cryptographic apparatus and method
4,405,829	Cryptographic communications system and method
4,399,323	Fast real-time public key cryptography
4,393,269	Method and apparatus incorporating a one-way sequence for transaction and identity verification
4,352,952	Data security module
4,281,215	Method and apparatus for securing data transmissions
4,979,188	Spectrally efficient method for communicating an information signal
4,974,191	Adaptive natural language computer interface system
4,969,188	Process and apparatus for the protection of secret elements in a network of encrypting devices with open key management
4,953,209	Self-verifying receipt and acceptance system for electronically delivered data objects
4,944,007	Public key diversification method
4,937,574	Data conversion
4,918,728	Data cryptography operations using control vectors
4,908,861	Data authentication using modification detection codes based on a public one way encryption function
4,907,274	Intelligent work station
4,893,339	Secure communication system
4,891,781	Modulo arithmetic processor chip
4,888,798	Modular software security
4,881,264	Digital signature system and method based on a conventional encryption function
4,876,716	Key distribution method
4,868,877	Public key/signature cryptosystem with enhanced digital signature certification
4,866,769	Hardware assist for protecting PC software
4,864,615	Reproduction of secure keys by using distributed key generation data
4,862,156	Video computer system including multiple graphics controllers and associated method
4,837,822	Cryptographic based electronic lock system and method of operation

4,827,514	Method and apparatus to detect and recover a pseudo-random sequence
4,827,507	Duplex analog scrambler
4,816,655	Method and apparatus for checking the authenticity of individual-linked documents and the identity of the holders thereof
4,811,394	Variable starting state scrambling circuit
4,811,377	Secure transfer of radio specific data
4,803,726	Bit synchronization method for a digital radio telephone system
4,799,061	Secure component authentication system
4,797,921	System for enciphering or deciphering data
4,760,600	Cipher system
4,757,534	Code protection using cryptography
4,757,532	Secure transport of information between electronic stations
4,751,733	Substitution permutation enciphering device
4,748,576	Pseudo-random binary sequence generators
4,733,345	Computer-telephone security device
4,723,284	Authentication system
4,720,859	Method and system for the mutual encyphered indentification between data communicating stations and stations for use with such method and system
4,689,606	Data encoding/decoding circuit
4,668,103	Polygraphic encryption-decryption communications system
4,658,093	Software distribution system
4,612,414	Secure voice transmission
4,567,600	Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission
5,438,622	Method and apparatus for improving the security of an electronic codebook encryption scheme utilizing an offset in the pseudorandom sequence
5,434,919	Compact endorsement signature systems
5,430,800	Facsimile apparatus
5,428,683	Method and apparatus for fingerprinting and authenticating magnetic media
5,425,103	Variable-key cryptography system
5,418,854	Method and apparatus for protecting the confidentiality of passwords in a distributed data processing system
5,416,842	Method and apparatus for key-management scheme for use with internet protocols at site firewalls
5,416,841	Cryptography system
5,414,771	System and method for the creation of random sequences and for the cryptographic protection of communications
5,410,598	Database usage metering and protection system and method
5,408,505	Method and apparatus for process control, tension control, and testing of magnetic media
5,406,628	Public key authentication and key agreement for low-cost terminals
5,400,403	Abuse-resistant object distribution system and method
5,398,285	Method for generating a password using public key cryptography
5,384,850	Security apparatus and system for retail environments
5,377,270	Cryptographic authentication of transmitted messages using pseudorandom numbers

5,375,169	Cryptographic key management method and apparatus
5,373,561	Method of extending the validity of a cryptographic certificate
5,373,558	Designated-confirmer signature systems
5,371,794	Method and apparatus for privacy and authentication in wireless networks
5,369,707	Secure network method and apparatus
5,369,706	Resynchronizing transmitters to receivers for secure vehicle entry using cryptography or rolling code
5,369,702	Distributed cryptographic object method
5,365,586	Method and apparatus for fingerprinting magnetic media
5,351,302	Method for authenticating objects identified by images or other identifying information
5,351,297	Method of privacy communication using elliptic curves
5,345,508	Method and apparatus for variable-overhead cached encryption
5,341,425	Methods and apparatus for uniquely encrypting data at a plurality of data transmission sites for transmission to a reception site
5,339,182	Method and apparatus for quantum communication employing nonclassical correlations of quadrature-phase amplitudes
5,338,043	Cryptographic guessing game
5,335,280	Random sum cipher system and method
5,323,146	Method for authenticating the user of a data station connected to a computer system
5,319,776	In transit detection of computer virus with safeguard
5,315,658	Fair cryptosystems and methods of use
5,301,247	Method for ensuring secure communications
5,299,263	Two-way public key authentication and key agreement for low-cost terminals
5,299,262	Method for exponentiating in cryptographic systems
5,297,800	Ultra-enigma code game
5,274,707	Modular exponentiation and reduction device and method
5,272,755	Public key cryptosystem with an elliptic curve
5,265,164	Cryptographic facility environment backup/restore and replication in a public key cryptosystem
5,261,002	Method of issuance and revocation of certificates of authenticity used in public key networks and other systems
5,253,294	Secure transmission system
5,251,258	Key distribution system for distributing a cipher key between two subsystems by one-way communication
5,243,649	Apparatus and method for quantum mechanical encryption for the transmission of secure communications
5,241,599	Cryptographic protocol for secure communications
5,241,596	Facsimile apparatus, and method of controlling same
5,239,583	Method and apparatus for improved security using access codes
5,237,611	Encryption/decryption apparatus with non-accessible table of keys
5,233,653	Apparatus and method for enciphered facsimile transmission and reception
5,231,662	Method and device for enciphering data to be transferred and for deciphering the enciphered data, and a computer system comprising such a

	device
5,228,084	Security apparatus and system for retail environments
5,222,136	Encrypted communication system
5,210,710	Modulo arithmetic processor chip
5,204,902	Cellular telephony authentication arrangement
5,204,901	Public key cryptographic mechanism
5,201,000	Method for generating public and private key pairs without using a passphrase
5,200,999	Public key cryptosystem key management based on control vectors
5,175,765	Robust data broadcast over a distributed network with malicious failures
5,164,988	Method to establish and enforce a network cryptographic security policy in a public key cryptosystem
5,161,193	Pipelined cryptography processor and method for its use in communication networks
5,159,634	Cryptosystem for cellular telephony
5,150,411	Cryptographic system allowing encrypted communication between users with a secure mutual cipher key determined without user interaction
5,142,578	Hybrid public key algorithm/data encryption algorithm key distribution method based on control vectors
5,136,647	Method for secure time-stamping of digital documents
5,121,408	Synchronization for entry to a network in a frequency hopping communication system
5,120,939	Databaseless security system
5,073,933	X window security system
5,062,001	Gray scale system for visual displays
5,054,066	Error correcting public key cryptographic method and program
5,046,094	Server-aided computation method and distributed information processing unit
5,029,208	Cipher-key distribution system
5,018,196	Method for electronic transaction with digital signature
5,016,276	Common cryptokey generation system and communication system using common cryptokeys
5,016,275	Buffered encryption/decryption
5,010,573	Cryptographic system by blocs of binary data
5,003,596	Method of cryptographically transforming electronic digital data from one form to another
4,995,081	Method and system for personal identification using proofs of legitimacy
4,984,272	Secure file handling in a computer operating system
RE34,954	Method for secure time-stamping of digital documents
RE34,161	Memory cartridge and information processor unit using such cartridge
5,465,299	Electronic document processing system and method of forming digital signature
5,455,865	Robust packet routing over a distributed network containing malicious failures
5,455,862	Apparatus and method for encrypting communications without exchanging an encryption key

5,455,407	Electronic-monetary system
5,454,040	Digital control of analog systems
5,453,601	Electronic-monetary system
5,452,358	Method and apparatus for improving the security of an electronic codebook encryption scheme utilizing a data dependent encryption function
5,450,491	Authenticator card and system
5,448,641	Postal rating system with verifiable integrity

References

Aharonian, Gregory. "Software Patent Statistics for 1994." http://www.eff.org/pub/Intellectual_property/1094_softpatent.stats, 1996.

Barrett, Paul M. "Justices Affirm Borland's Victory in Copyright Dispute with Lotus." Wall Street Journal 17 Jan. 1996, sec C: 20-21.

Berners-Lee, Tim, et al. "The World-Wide Web." Communications of the ACM Aug. 1994: 76-82.

Calderwood, Richard. "Patents: What Are They? and How Do I Read Them?" Pamphlet. Intel Corp., 1994.

Clark, Peter, and Neil Staunton. Innovation in Technology and Organization. New York: Routledge, 1989.

Davis, Randall. "Intellectual Property and Software: The Assumptions Are Broken." Address Before the Proceedings of the WIPO Worldwide Symposium on the Intellectual Property Aspects of Artificial Intelligence. 25-27 Mar. 1991.

Duval, Beverly K., and Linda Main. "Exploring the Internet with Mosaic." Library Software Review Win. 1994: 269-279.

Ernst, Warren. Using Netscape. Indianapolis: Que, 1995.

Haustein, Heinz-Dieter, and Harry Maier. Innovation and Efficiency: Strategies for a Turbulent World. Oxford: Pergamon Press, 1985.

Henry, B., ed. Forecasting Technological Innovation. Boston: Kluwer Academic, 1991.

Hickman, Brian D. "Claiming Software." Presentation. Blakely, Sokoloff, Taylor & Zafman, Sunnyvale. 17 Nov. 1995.

Hill, Christopher T. "Effects of Governmental Regulation on Technological Innovation." Technological Innovation and Economic Development: Has the U.S. Lost the Initiative?--Proceedings on a Symposium on Technological Innovation, Washington, D.C., April 19-20, 1976. Ed. Kenneth F. Gordon. Washington: Energy Research and Development Administration, 1976.

Holt, Knut. Innovation: A Challenge to the Engineer. New York: Elsevier, 1987.

Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995).

Marshall, Jonathan. "Lotus Case Residue to Last Years." San Francisco Chronicle 22 Jan. 1996, sec. B: 1-2.

Massachusetts Computer Software Council, Inc. "Software Industry 1993 Business Practices Survey." Interviews with Software Company CEOs. 1993.

Merrin, Robin. The Mosaic Roadmap. San Francisco: Sybex, 1995.

National Center for Supercomputing Applications (NCSA). "Help On Version 0.10." http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/help-on-version-0.10.html, 1996.

---. "Introduction to NCSA Mosaic for X." http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/d2-intro.html, 1996.

---. "Mosaic Awards." http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/Awards/MosaicAwards.html, 1996.

---. "Mosaic for X version 2.0 Fill-Out Form Support." http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html, 1996.

---. "NCSA Mosaic for the X Window System User Guide." http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/UserGuide/, 1996.

---. "NCSA Mosaic for the X Window System 2.0." http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/help-on-version-2.0.html, 1996.

Netscape Communications Corporation. "Navigator Business Update." http://search.netscape.com/misc/developer/conference/proceedings/gs2/sld005.html, 1996.

---. "Netscape Data Security." http://search.netscape.com/newsref/ref/netscape-security.html, 1996.

---. "Netscape's Mission and Milestones." http://home.netscape.com/comprod/netscape_mission.html, 1996.

---. "Netscape Press Release 10/13/94." http://home.netscape.com/newsref/pr/newsrelease1.html, 1996.

---. "Netscape Press Release 12/15/94." http://home.netscape.com/newsref/pr/newsrelease8.html, 1996.

---. "Netscape Press Release 3/5/96." http://home.netscape.com/newsref/pr/newsrelease101.html, 1996.

---. "On Security." http://search.netscape.com/newsref/ref/internet-security.html, 1996.

---. "Using RSA Public Key Cryptography for Internet Security." http://search.netscape.com/newsref/ref/rsa.html, 1996.

Nelson, Theodor H. Literary Machines 93.1. Sausalito: Theodor Nelson, 1993.

Nimmer, Raymond T. The Law of Computer Technology. Boston: Warren Gorham Lamont: 1992.

RSA Data Security, Inc. "About RSA." http://www.rsa.com/ABOUTRSA/, 1996.

---. "RSA Press Release 3/4/96." http://www.rsa.com/cylink2.html, 1996.

Samuelson, Pamela, et al. "A Manifesto Concerning the Legal Protection of Computer Programs." Columbia Law Review 94 (1994): 2308-2431.

Schatz, Bruce R., and Joseph B. Hardin. "NCSA Mosaic and the World Wide Web: Global Hypermedia Protocols for the Internet." Science 265 (1994): 895-901.

Spyglass, Inc. "Newsflash 8/24/94." http://www.spyglass.com/newsflash/releases/aug2494_1.html, 1996.

---. "Newsflash 10/17/94." http://www.spyglass.com/newsflash/releases/oct1794_1.html, 1996.

Udell, Gerald G., Michael F. O'Neill, and Kenneth G. Baker. Guide to Invention and Innovation Evaluation. Portland: University of Oregon, 1977.

U.S. Patent and Trademark Office. "Proposed Amendments to Rules of Practice to Implement 20-Year Patent Term and Introduction of Provisional Applications as Contained in the Uruguay Round Agreements Act." http://www.uspto.gov/web/20year/hearings/transcript.html, 1996.

Vaughan-Nichols, Stephen, et al. Inside the World Wide Web. Indianapolis: New Riders Publishing, 1995.

Yourdon, Edward, and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Englewood Cliffs: Prentice-Hall, 1979.