NISTIR 4404

NEW NIST PUBLICATION

December 1990

DYNAMIC CHARACTERISTICS OF HYPERTEXT

Richard Furuta
P. David Stotts

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Office Systems Engineering Group
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 4404

DYNAMIC CHARACTERISTICS OF HYPERTEXT

Richard Furuta
P. David Stotts

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards
and Technology
National Computer Systems Laboratory
Office Systems Engineering Group
Gaithersburg, MD 20899

August 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
John W. Lyons, Director

Dynamic Characteristics of Hypertext

Richard Furuta and P. David Stotts
University of Maryland[*]
and
National Institute of Standards and Technology[†]

Abstract

We consider a *dynamic hypertext* as one that changes even in the absence of user activity. Hypertexts are comprised of structure, content, and context, and dynamic operations may affect any of them. Dynamic operations are synchronous (related to the hypertext's description) or asynchronous. We call the former *timers* and *sensors*. In this paper we argue that such dynamic characteristics are an important component of many hypertext systems, surveying existing implementations to examine the different realizations of dynamic operations, and that dynamic applications are also of importance within the hypertext domain.

Keywords: hypermedia, hypertext, sensors, temporal aspects, time, timers

1 Introduction

Information structured as a hypertext requires an interactive computer system for effective presentation. However, the degree of interactivity permitted in the information structured by the hypertext and in the hypertext itself varies from system to system. In some systems, the display of information is passive, i.e., unchanging except perhaps in direct response to a user's request. In others, the information displayed may incorporate dynamically-changing components (components that change, even in the absence of user input; for example, animations). In these systems, however, it is possible that the selection of *which* information is being shown will not change without user action. A further possibility for dynamic change in a hypertext is therefore that the selection of displayed information elements may also be changing, even in the absence of user activity.

In this paper, we will consider the characteristics of dynamic hypertexts and the differing capabilities of their implementations. Our goal will be to argue that inclusion of such dynamic characteristics is required if the full capabilities of hypertextual representation is to be achieved.

[*]Affiliated with the Department of Computer Science and the Institute for Advanced Computer Studies at the University of Maryland, College Park, MD. The portion of this work carried out at the University of Maryland is based upon work supported by the National Science Foundation under Grant No. CCR–8810312. It is also supported in part by a grant from the Northrop Research and Technology Center. Address correspondence to Department of Computer Science, University of Maryland, College Park, MD 20742.

[†]Affiliated with the Office Systems Engineering Group, Systems and Software Technology Division, National Computer Systems Laboratory. Contribution of the National Institute of Standards and Technology. Not subject to copyright. The mention of commercial products in this paper is to describe the research environment and does not imply recommendation or endorsement by the National Institute of Standards and Technology.

We note as clarification that by our use of the term "hypertext," we do not wish to imply that the document is limited only to textual material. Indeed, we place no limits on the permissible kinds of content. Documents may incorporate textual, graphical, video, audio, etc., material.

We will further specify what we mean by dynamic hypertext in the next section and discuss how such characteristics are included in existing systems in Section 3. Finally, in Section 4, we describe some useful applications that can be accomplished only with dynamic hypertext.

2 Hypertext and dynamic hypertext

Our notions of what makes up a hypertext were presented earlier in our description of a high-level reference model for hypertext [FS90a], and will be summarized here. Abstractly, the hypertext can be separated into *structure*, *content*, and *context* [FS89, Fur89]. The term "structure" may initially convey a bit too rigid an image of the hypertext. The structure defines the relationships among elements of the hypertext. A natural representation is a graph-like data structure, but it is also possible that it may be implemented by non-graph-based means, for example as the result of a computation.

We further separate the content of the hypertext from the structure—the structure therefore must contain "placeholders" that are associated with specific content elements. The relationships described by the structure must be presented to the hypertext's reader to permit traversal of the structure. The context is the representation of these relationships within the displayed content.

Viewed from the perspective of this framework, a hypertext system may incorporate dynamic behavior in any of the three parts: structure, content, and context. The most commonly-found dynamic behavior affects the content of the hypertext. In addition to the example of animation, mentioned earlier, dynamic displays may also be used in showing the results of continuing computations, may obtain information interactively from the hypertext's reader, etc.

Dynamic representation of context may also be useful. For example consider the representation of an anchor[1] that changes over time. The anchor may be represented by a highlighted region whose color, size, or location changes over time to draw more attention to itself. Alternatively, the anchor might be represented by a small animation.

An application that naturally benefits from dynamic structure is that of structured presentations. For an example, see Zellweger's Scripted Documents system [Zel87, Zel88, Zel89]. Dynamic structure may also be used in constraining aspects of a reader's browsing session—for example limiting the amount of time that a particular information element can be accessed or alternatively insuring that a particular piece of information is encountered during each time increment.

We have informally defined dynamic operations to be those that proceed in the absence of user activity. A second dimension in the behavior of a hypertext system are the means provided for controlling operations. Two different means can be identified by which activities can be initiated and controlled: user-controlled and system-controlled. Of these, user-controlled events are static (i.e., take place only in response to a user's actions), while the system-controlled events are dynamic (i.e., take place independently of a user's actions, although they may be enabled by a user's action).

The class of dynamically-controlled operations on structure, content, and context may be considered as comprised of those system-initiated events that are controlled *synchronously* and those controlled

[1]*Anchor* is the commonly-used term for the terminal points of a link. It additionally conveys the sense that such a terminal point identifies an "interior" portion of a content element—a word or phrase, a feature in a diagram, or an individual sound in an audible passage. A *link* represents a relationship between elements of the content—i.e., as represented by the hypertext's structure.

asynchronously. "Synchronous" and "asynchronous" are defined here relative to the description of the hypertext. Analysis of the hypertext can theoretically determine when synchronously-controlled events will occur, relative to a given traversal of the hypertext by a reader.[2] Asynchronously-controlled events are invoked independently of the hypertext's description—for example in response to an externally-generated event that has been detected by a separate process. As a shorthand notation that reflects the differences in specification, we will say that synchronously-controlled events are controlled by *timers* and that asynchronously-controlled events are controlled by *sensors*. In other words, when an event is controlled by a timer, its actions are specified based on the clock—it may be invoked at a particular time, terminated after being active for a certain amount of time, etc. When an event is controlled by a sensor, its actions are dependent on externally-generated signals. Such events may be invoked, for example, when values produced by some external process exceed a specified threshold value, when some device is activated or deactivated, etc.

Timer-controlled and sensor-controlled events are not mutually exclusive, of course. Particular applications can certainly be thought of that depend on both.

3 Dynamic aspects of hypertext systems

In this section, we will consider what dynamic behaviors are incorporated into particular hypertext systems. For each system, we will consider the availability of dynamic effects within structure, content, and context, and then will consider the control mechanisms provided (i.e., whether the system provides the means to implement timers and sensors).

3.1 Hyperties

Hyperties [Shn87, Cog88] is a commercially-available hypertext system for the IBM-PC. The primary focus in the Hyperties design seems to be to provide a system with an intuitively-understandable and accessible user interface. Hyperties version 2.35's screen design is predictable and consistent, and perhaps intentionally not particularly extensible. (Apparently Hyperties version 3.0 will be far more extensible, possibly at the expense of this design consistency).

Hyperties only provides rudimentary support for dynamic content and no support for dynamic structure or dynamic context. In Hyperties 2.35, one is limited to swapping to an external videodisk sequence, replacing the Hyperties display with the display of the videodisk images. When the sequence terminates, the Hyperties browsing session resumes, overlaying Hyperties' buttons over the last frame of the sequence. Communication between Hyperties and the external world is purely one-directional—Hyperties can specify the bounds of the sequence to be controlled, but no information is passed back into Hyperties from the external process.

Hyperties does not permit specification either of timers or of sensors.

3.2 HyperCard

The philosophy behind the design of Apple's HyperCard [App87] is quite different from that of Hyperties. HyperCard's emphasis, as reflected by the features of HyperTalk [App88], the specification language for HyperCard, is on providing a general-purpose implementation base on which to build interactive

[2]We recognize that in practical terms, the analysis may be too computationally expensive to actually carry out.

applications. Indeed, HyperCard edges the boundary between Hypertext system and general-purpose programming environment.[3]

HyperCard's primitive objects are buttons (selectable objects) and fields (containing text). These are placed onto cards or onto backgrounds. The display seen by a user is formed by combining a card with a background. Cards are contained in stacks, and one stack is distinguished as the "home stack." HyperTalk is based on an event-driven paradigm, with messages passing along a hierarchy until an appropriate handler is encountered. In general, messages seeking a handler pass first to the buttons or fields, and then to the card, the background, the stack, the home stack, and then to HyperCard itself. It is also possible to send messages to objects not in the current hierarchy, in which case the hierarchy associated with the receiving object is used to handle the message.

HyperCard also permits invocation of externally-defined routines. The externally-defined routine can communicate with the HyperCard-defined world, as messages can be passed between the externally-defined routine and HyperTalk-defined routines.

Since HyperTalk is a general-purpose programming language, system-defined facilities can be used to implement dynamic structure, content, and context. Particular support is provided to implement both timers and sensors. Timers can be based on functions such as `seconds` and `ticks`. Messages such as `mouseDown`, `mouseStillDown`, and `mouseUp` can be used to create sensors that base processing on such user actions.

Handlers based on the `idle` message, which is generated repeatedly when no user activity is going on, can be used to implement actions that proceed in the apparent absence of external activity. However, implementation of time-based events, such as automated backups or automatic updates of displayed information, requires cooperation between the programmers of the events and the programmers of the hypertext document being examined—for example, messages such as `idle` could be trapped by a handler written by the document programmer, and would not then be passed along to the handler associated with the event.

3.3 HyTime and SMDL

The Standard Music Description Language (SMDL) is intended to represent musical documents to enable interchange [New90, Ber90, GNST90]. It is being defined in the notation of the SGML [ISO86] document type definition (in essence, as an attributed context-free grammar). A portion of SMDL, a notation for connecting together and synchronizing different simultaneously-active time-based events has been separated out and called "HyTime" (for "Hypermedia/Time-based Document Language") [GT90, GN90].

It is difficult to cast HyTime and SMDL in the framework of structure, content, and context. The questions about how the context is to be represented appear to be outside of the current scope of the SMDL activities. Structure and content are considered but the distinctions between structure and content are blurred. It appears that a work of music is being thought of as a number of separate but related threads. If an element of the hypertext's content is taken to be a major division of the work (say a movement), then the domain of HyTime is largely within the content, and consequently the content is dynamic. If the element is an individual component of a thread, then the HyTime operations are structural. What is clear, in any case, is that the musical domain can require specification of both dynamic content and dynamic structure.

HyTime defines a rich set of components for representing time. HyTime permits definition of a set of clocks, which may be divided into those that are based on "real time" and those that are based on "musical time." Clocks based on real time are directly related to actual time. However, those clocks based on

[3]We have discussed some of the characteristics of this boundary elsewhere in the context of our own Trellis system [FS90b].

musical time are based on a more abstract time, measured in virtual time units, that may have no inherent equivalent in real time units. Presumably even if a correspondence between real and musical time exists at a particular point in the score, this correspondence may be changed by changes to the presentation's tempo. Although the separate musical clocks can proceed in asynchrony, it is also possible to specify that they are to be synchronized at particular points in the score.

The primary focus of HyTime primitives is to provide timer-related mechanisms and not sensor-related mechanisms. The timers that can be specified can be quite complex, for example HyTime permits specifying that an event is to last until the beginning of another event ("duration until"). The overall emphasis of the SMDL effort suggests that timers are far more important to music than are sensors.

3.4 Trellis

Our own work has defined the Trellis model of hypertext [SF89, SF90]. A significant characteristic of the Trellis model, which is based on the Petri net, is its natural and integral representation of concurrency and synchronization. This permits specification of hypertexts in which multiple content elements are active at the same time, invoked and removed either independently or in a concurrent fashion, based on the author's specification.

Content in Trellis is arbitrary in form and can be dynamic. The basic model has been augmented with timing specifications that can automatically select links after a certain amount of time has passed and also can prohibit access to a link for a certain amount of time. Consequently, the Trellis model directly incorporates a form of dynamic structure.

The prototype implementations of the Trellis hypertext model, the αTrellis and χTrellis systems, are based on a client/server model. In the prototypes' architectures, the server contains an "engine" that implements the Trellis model. Zero or more clients communicate with this engine via remote procedure call invocations. Several different classes of clients have been developed to take advantage of particular elements of the operating environment. The most frequently used client, for example, permits text browsing, interactive process invocation, and link traversal. A graphics-based client shows associated images on a separate specialized display. Note that if two instances of the same class of client are associated with a particular server, the display seen by each client (and the functionality provided to the browser) is a duplicate of the other's. This permits an initial method for providing a distributed hypertext that can be browsed by simultaneous and cooperating browsers.[4] In the current prototypes, all clients are kept informed about the current state of the browsing session (i.e., which content elements are active and which links are selectable). Decisions about which portions of the browsing state to display are made by the clients, not the server.

The Trellis model directly incorporates timers. Sensors are implemented through application of the client/server architecture. A process implementing a sensor will behave as an independent browser, using a program-callable interface to communicate its state changes to the server (and consequently to the other browsers) by selecting links. Note that as the sensor is a client, it has access to the current state of the hypertext and can base its decisions on that information, if appropriate.

[4]It is entirely possible and expected that simultaneously-active clients will result in cases in which concurrent actions cause conflict. Such conflicts are handled by the server. Implementation of this resolution is made particularly easy by the organization of the server as a collection of abstract data types.

3.5 Scripted documents

Zellweger's scripted document system [Zel87, Zel88, Zel89] permits overlaying *active paths* through a set of documents and within a single document. While the term "document" is used in a traditional way, an arbitrary action can be associated with the subsequent presentation of the document; i.e., an action can invoke an arbitrary computer application, including those that produce dynamically-changing displays. Consequently, this provides a form of dynamic content.

Paths are described by a *script*, which consists of a sequence of entries describing the steps along the path. Each entry in the script is described by a template, which includes fields permitting invocation of an arbitrary action and specification of the sequencing behavior on subsequent playback of the path. The sequencing behavior, in other words the process by which the playback steps along the entries in the path, can be application-based (i.e., waiting until the application terminates or time-based sequencing (i.e., continuing after a specified period of time), and reader confirmation can also be required before stepping.

Paths, therefore, provide direct support for dynamic structure and can be used to implement dynamic context. However, the primary support is for timer-controlled events rather than sensor-controlled events. While timer behavior is specified directly in the script entry, it appears that only a limited form of sensor is provided. The script may invoke any program, but it appears that the communication from the program to the script is limited to notification of completion. The strength of scripted documents, therefore, is in the organization of material from potentially heterogeneous sources for automated or semi-automated presentation in a structured fashion.

3.6 PROXHY

Kacmar has implemented PROXHY (for "PRocess-based Object-oriented eXtensible HYpertext system") [Kac90], which is based on a general hypertext architecture defined by Leggett. PROXHY is designed around an integrated process message-passing model and object-oriented class-instance inheritance model. The general model distinguishes anchors, links, and information elements within the hypertext. In PROXHY, the information elements can be applications, for example animations, and the anchors can invoke a program (i.e., both content and anchors can be dynamic). In our categorization, then, this provides direct support for dynamic content and dynamic context. It is unclear whether PROXHY supports dynamic structure, but even if not it appears that it could be added. In other words, dynamic structure appears compatible with the capabilities of the PROXHY link.

The focus in the PROXHY description is presenting the architecture of the general hypertext and of the PROXHY implementation, as well as describing the manner in which the objects that implement PROXHY communicate. Issues of timers and sensors are not directly addressed, but again inclusion of timers and sensors appears compatible with the architectures.

3.7 The Dexter model

The Dexter hypertext model [HS90] is not an abstraction of one specific implementation or family of implementations. It is of particular importance, however, to mention because it is an abstraction derived from the collective experience of many researchers implementing over a dozen (fairly diverse) systems. In this model, the notion of a *presenter* encapsulates dynamic content. A presenter is a function associated with a node in a hypertext. This function is applied to its content element to produce information in consumable form during browsing. A presentation function can be as simple as dumping text to a window, or it can be a dynamic program such as an editor, an animator, or some visualization tool. Timers are not specifically mentioned in the Dexter model. Sensors can be accomplished as in Trellis, with programs acting as virtual users interacting with the base engine of a document.

4 Hypertext application of timers and sensors

In this section, we will consider a number of timer and sensor dependent applications that one might wish to include in a hypertext system. While each of the applications can be presented as a special-purpose feature of a given hypertext implementation, we suggest that a general implementation of timers and sensors will permit the direct expression of these activities.

Structured presentations

As noted in the discussion of Zellweger's scripted documents, structured presentations are a natural application of timers. Such presentations can be relatively self-directed (e.g., a demonstration that plays continuously) or relatively interactive (e.g., a demonstration that changes as requested by the reader). Similarly, structured presentations may be usefully imbedded within larger presentations, perhaps even as implementation of simple animations.

Reflecting external events

The direct use of sensors is to permit reflection within a hypertext of events that occur outside of the hypertext. As a simple example, the sensor can be used to implement a trigger, perhaps giving notification to the reader when some event occurs by causing the system to "browse" the document concurrently with the reader thereby giving visual or audible cues. As a further example, in a software-engineering-oriented environment, sensors can be used to cause the hypertext system to parallel the activities being carried out, thereby providing appropriate informative displays automatically at each step of the process. It is, of course, a limitation to only consider display of information. Coupled with dynamic content, it may be possible to actually implement sophisticated shell-script-like controllers with sensors that invoke processors when required by the creation or modification of data.

Time-limited events

HyTime and scripted documents have provided excellent examples of use of timers to *limit* events. Two situations are immediate: those in which the timer is used to guarantee that some event takes a certain amount of time, and those in which the timer is used to insure that the event completes within a specified amount of time. Such capabilities are also useful in controlling interfaces with hardware devices—for example giving a CD-ROM Disk sufficient time to spin up, or insuring that the system is ready when the next item of data is presented. In more sophisticated application, such limits might also be used to alter the hypertext display, perhaps to skip over parts of the hypertext if some predefined limit was about to be exceeded.

Periodic events

Timers can be used to invoke periodic events; for example a regular automatic backup of an actively-changing hypertext. Note that if the period is very small (perhaps even instantaneous) then the event will appear to occur automatically when some preconditions hold.

Active help

Lerner [Ler89] has identified *active help* in a user interface as help that is presented when it appears that the reader has become confused, perhaps as reflected by long periods of inactivity. Clearly this is another form of periodic event that can be implemented with timers, but one that is tied to specified user actions, perhaps most appropriately detected by sensors.

5 Conclusions

In this paper, we have defined dynamic hypertext and have discussed the dynamic aspects of existing hypertext systems. We believe that an important distinction in dynamic operations is whether they are tied to the hypertext's representation (called timers), or independent of the representation (called sensors). Inclusion of timers and sensors in a hypertext implementation increases the flexibility of the resulting system without requiring additional special-purpose functions. The resulting hypertexts may be used to implement interactive applications that begin to approach those commonly expressed in a more conventional programming language. Indeed, such hypertexts may be of use in situations where there are no human readers at all; i.e., to control machine-based processes.

References

[App87] Apple Computer, Inc. *HyperCard User's Guide.* Apple Computer, Inc., 1987.

[App88] Apple Computer, Inc. *HyperCard Script Language Guide: The HyperTalk Language.* Addison-Wesley, 1988.

[Ber90] Steven V. Bertsche. X3V1.8M SD-6 HyperMedia/time-based document (HyTime) and Standard Music Description Language (SMDL): User needs and functional specification. Technical Report SD-6, ANSI Project X3.542-D, April 1990. Second Draft.

[Cog88] Cognetics Corporation. *Hyperties Author's Guide*, August 1988. Corresponds to version 2.3 of the software.

[FS89] Richard Furuta and P. David Stotts. Object structures in paper documents and hypertexts. *BIGRE*, (63-64):147–151, May 1989.

[FS90a] Richard Furuta and P. David Stotts. The Trellis hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 83–93. National Institute of Standards and Technology, February 1990. NIST Special Publication 500-178. Workshop held January 16–18, 1990.

[FS90b] Richard Furuta and P. David Stotts. Generalizing hypertext. *T.S.I.—Technique et Science Informatiques*, 1990. To appear.

[Fur89] Richard Furuta. An object-based taxonomy for abstract structure in document models. *The Computer Journal*, 32(6):494–504, 1989.

[GN90] Charles F. Goldfarb and Steven R. Newcomb. X3V1.8M journal of development; ANSI project X3.542-D; Hypermedia/Time-based structuring language (HyTime). Technical Report SD-7, ANSI Project X3.542-D, April 1990. Sixth Draft.

[GNST90] Charles F. Goldfarb, Steven R. Newcomb, Donald Sloan, and Alan D. Talbot. X3V1.8M journal of development; ANSI project X3.542-D; Standard Music Description Language (SMDL). Technical Report SD-8, ANSI Project X3.542-D, April 1990. Sixth Draft.

[GT90] Charles F. Goldfarb and Alan D. Talbot. X3V1.8M/SD-7 journal of development; Standard Music Description Language (SMDL), Part two: Hypermedia/Time-based document subset (HyTime). In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 180–188. National Institute of Standards and Technology, February 1990. NIST Special Publication 500-178. Workshop held January 16–18, 1990.

[HS90] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 95–133. National Institute of Standards and Technology, February 1990. NIST Special Publication 500-178. Workshop held January 16–18, 1990.

[ISO86] ISO. *Text and Office Systems—Standard Generalized Markup Language*, October 1986. Document Number: ISO 8879–1986(E).

[Kac90] Charles John Kacmar. *PROXHY: A Process-Oriented Extensible Hypertext Architecture*. Ph.D. dissertation, Texas A&M University, 1990.

[Ler89] Barbara Staudt Lerner. *Automated Customization of User Interfaces*. Ph.D. dissertation, Carnegie-Mellon University Computer Science Department, Pittsburgh, PA, September 1989. Also issued as Technical Report CMU-CS-89-178.

[New90] Steven R. Newcomb. Explanatory cover material for section 7.2 of X3V1.8M/SD-7, fifth draft. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, page 179. National Institute of Standards and Technology, February 1990. NIST Special Publication 500-178. Workshop held January 16–18, 1990.

[SF89] P. David Stotts and Richard Furuta. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1):3–29, January 1989.

[SF90] P. David Stotts and Richard Furuta. Temporal hyperprogramming. *Journal of Visual Languages and Computing*, 1990. To appear.

[Shn87] Ben Shneiderman. User interface design for the Hyperties electronic encyclopedia. In *Proceedings of Hypertext '87*, pages 189–194, November 1987. Published by the Association for Computing Machinery, 1989.

[Zel87] Polle T. Zellweger. Directed paths through collections of multi-media documents. In *Hypertext '87*, November 1987. Position paper.

[Zel88] Polle T. Zellweger. Active paths through multimedia documents. In J. C. van Vliet, editor, *Document Manipulation and Typography*, pages 19–34. Cambridge University Press, April 1988.

Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), April 20–22, 1988.

[Zel89] Polle T. Zellweger. Scripted documents: A hypertext path mechanism. In *Hypertext '89 Proceedings*, pages 1–26. ACM, New York, November 1989.

NIST-114A (REV. 3-89) **U.S. DEPARTMENT OF COMMERCE** NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY	1. PUBLICATION OR REPORT NUMBER NISTIR 4404
BIBLIOGRAPHIC DATA SHEET	2. PERFORMING ORGANIZATION REPORT NUMBER
	3. PUBLICATION DATE August 1990

4. TITLE AND SUBTITLE

Dynamic Characteristics of Hypertext

5. AUTHOR(S)

Richard Furuta and P. David Stotts

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899	7. CONTRACT/GRANT NUMBER
	8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

NIST
National Science Foundation/CCR-8810312
Northrop REsearch and Technology Center

10. SUPPLEMENTARY NOTES

☐ DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

We consider a dynamic hypertext as one that changes even in the absence of user activity. Hypertexts are comprised of structure, content, and context, and dynamic operations may affect any of them. Dynamic operations are synchronous (related to the hypertext's description) or asynchronous. We call the former timers and sensors. In this paper we argue that such dynamic characteristics are an important component of many hypertext systems, surveying existing implementations to examine the different realizations of dynamic operations, and that dynamic applications are also of importance within the hypertext domain.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

hypermedia, hypertext, sensors, temporal aspects, time, timers

13. AVAILABILITY	14. NUMBER OF PRINTED PAGES 13
[X] UNLIMITED	
[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).	
[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.	15. PRICE A02
[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.	

ELECTRONIC FORM