Storyspace 3

Mark Bernstein
Eastgate Systems, Inc.
134 Main Street
Watertown MA 02472 USA
+1 (617) 924-0044
bernstein@eastgate.com

ABSTRACT

Storyspace was introduced in one of the first papers presented at the first ACM Workshop of Hypertext, and gave rise to a number of significant hypertexts, both fiction and nonfiction. A new implementation of Storyspace for contemporary computing environments is clearly desirable. This has been undertaken, with modest resources and in a short time frame. A number of surprising new facilities, many of them originally proposed in contrast or opposition to Storyspace, can be supported without altering or complicating the underlying Storyspace node and link model.

CCS Concepts

•Software and its engineering→Software creation and management → Designing Software •Applied Computing→Computers in other domains.

Keywords

Storyspace, hypertext, hypermedia, literature, fiction, education, design, implementation, support, history of computing, maps, links.

1.INTRODUCTION

We seldom discuss the design and implementation of hypertext systems anymore. This was once the core concern of this conference, but few new systems have a been described in these Proceedings in recent years [25][2][27].

Over time, the constraints and design forces on hypertext systems have changed. Yet *afternoon* is still the same, and we still want to read *afternoon* [29]. Our interests are not those of the bibliographer, the book collector, or the media archaeologist: we simply want to read about (or teach our students about) the fellow who wants to say that he may have seen his son die this morning. We prefer convenience, but are willing to take a certain amount of trouble. We prefer economy, but are willing accept a measure of expense. We want the experience to conform to the author's expectation (intention here is a suspect quality), but of course we need not mimic every incidental detail, shortcut and flaw.

Though Storyspace was perhaps the smallest of the first-generation hypertext systems, its implementation demanded substantial resources. Development was funded in part by a grant from the Markle Foundation, with support from Broderbund software, the University of North Carolina, Jackson (Michigan) Community College, as well as the Roger Schank's Artificial Intelligence lab at Yale, where Storyspace authors Michael Joyce and Jay David Bolter were visiting fellows in successive years. Writing of Storyspace 1, like other first-generation hypertext systems, was a substantial undertaking [3].

2.STORYSPACE

Resources on this scale are not readily available for developing hypertext systems today. This has long been the case; Janet Murray once remarked that Storyspace's shortcomings reflected the dearth of resources available to humanities computing. More recently, novelist Paul La Farge attributed the failure of previous efforts to create sustained narrative in hypertext fiction to Storyspace's impoverished visual design[40].

Still, we have learned a great deal about hypertext systems since 1987. Software development tools and methodologies have advanced substantially. What had once required many hands and many dollars had now to be reproduced as a part-time summer project by a single hand.

Storyspace 1 [7] [28] [30] is a stand-alone, monolithic hypertext writing and reading environment, one chiefly intended for reading and writing hypertext narrative. From the start, it offered a versatile visual map of node-link hypertexts organized with a hierarchical backbone, the capability of multiple perspectives and views, and directional dynamic links whose behavior could depend on the reader's trajectory through the hypertext. Storyspace hypertexts were widely reviewed, admired and reviled, and many continue to be taught and studied[9].

Over the years, Storyspace has been reimplemented for new computing environments. Storyspace 1 was written, in Pascal, for Macintosh System 5. Reimplementations by this author include Storyspace for Windows (in C), Storyspace 2 (in C++ for Metrowerks PowerPlant and OS X), and now Storyspace 3 (in Objective C++ for the Macintosh Cocoa framework using the Hereford foundation from Tinderbox).

3.IMPLEMENTATION

New hypertext systems are pleasant and interesting tools for study and research. Because they are new, they are small: changes can be made and tested quickly. Because they are small, changes have limited scope. Rapid iteration permits freer experimentation. Moreover, even where resources are ample, the temporal constraints of contemporary research impose important limitations on experiments that involve mature systems. The timescales of the summer internship, the master's thesis, and of the doctoral dissertation impose stern and unyielding restrictions. The more time we spend waiting for the compiler, the less time we have for creative research.

Mature hypertext systems like Storyspace present a significant challenge to agile research because everything, naturally, depends on the hypertext and its nodes.

3.1.Recompiling The World

Three of the oldest classes in Storyspace 2 – the first object-oriented implementation – are Hypertext, Link, and Node. Over time, these classes naturally acquired new functionality and responsibilities. As Storyspace and its foundation gained new facilities, moreover, almost all these facilities relied on Node and Hypertext. A text pane obtains the text it is to view from the Node, the Map View obtains the title of each item from its Node, the pasteboard manager copies a representation of the Node to the clipboard when the user selects Copy from the Edit menu. Everyone needs to use Node, and Node needs to provide convenience functions to everyone.

As a result, changes to the interface of Node or Hypertext require recompiling everything else. This is a small penalty for a small system, but over time the penalty grows. Testing such large and monolithic classes is difficult, development slows, and the prospect of improving core classes grows unpleasant. This unpleasantness may sometimes be tolerable in an industrial environment, but it is inimical to experimentation. Besides, we have better uses for graduate students than waiting for the compiler.

The classic prescription for large and monolithic objects, of course, is to decompose them into a cluster of small objects, each with a single responsibility [34]. Unfortunately, such decompositions proved difficult to find – and because every step along the way again requires recompiling the world, the work is exceedingly slow and costly. The classic refactoring prescriptions – encapsulating instance variables, splitting the object along implementation seams, sprouting classes – are frustrated by the regularity and flexibility of the underlying attribute-value store. Each step in each refactoring is likely to require recompiling everything.

To restore the system's malleability, we introduced a family of new classes, NodeFacades and HypertextFacades, each of which owns only a single instance variable – the underlying Node or Hypertext. These facades provide small and focused slices of functionality; for example, NodeLinks provides access to links associated with a specific node, NodeIndexer provides an API to support tf-idf similarity searches, and NodeDeleter provides access to methods for deleting Nodes. Initially, these classes are simple facades, forwarding calls to Node or Hypertext. Client classes can now use one or more facades in place of using Hypertext or Node, and so the dependency graph is gradually decoupled. Because the Facade objects simply wrap a pointer, they can be created and thrown away at will. No changes need to be made to Node or Hypertext, and so this work can proceed quickly.

In time, some methods in the underlying classes were seldom or never used without the intermediacy of the Facade. Here, functionality could be refactored from the underlying class to the Facade, and the underlying class can now use the Facade. This refactoring is invariant with respect to the underlying class's interface, and so it once again avoids recompilation. The progressive refactoring can continue until the base classes are extensively hollowed out until they serve as Value Objects with a plethora of trivial helper methods.

I mention this refactoring because it is likely to impact any compiled hypertext system with significant age or complexity. It bears some resemblance to familiar idioms – pImpl, proxy, interface object – but it seems to have been seldom discussed in the monograph literature[23][31][22][24] and might prove broadly useful in hypertext research.

4.GUARD FIELDS

4.1.The Storyspace Guard Field

The distinctive feature of Storyspace is its dynamic link, a unidirectional link that can be activated or deactivated by a *guard field.* The guard field is a simply boolean expression whose terms may include the word clicked or the names of previously-visited notes enclosed in quotation marks. The guard field

("A" & (!"B")) | Anne

is satisfied if the reader has read the note "A" but not the note "B", or if the reader has clicked on the word "Anne". Guard fields proved invaluable for breaking cycles[5], a central anxiety of early hypertext research [18][11].

The original design of guard fields proved effective in terms of hypertext rhetoric was well of engineering. The notation is concise, a consideration that mattered greatly when storage and memory alike amounted to a few hundred kilobytes. The original formulation lends itself readily to parsing by recursive descent; fast and reliable guard field interpreters were never a source of concern. The underlying mechanism, which simply disables unwanted links, is easy for new writers to understand.

Yet the original formulation was not without disadvantages. The syntax was always hard to teach. The rest of Storyspace could be explained in one class session – a session in which, in the early years, many students had their first encounter with a computer – but guard fields needed a second session to themselves. While isolated guard fields are easy enough to test, moreover, the entire hypertext network becomes, with guard fields, a distributed program describing an elaborate finite state machine. The lack of visualization and debugging tools, and the distribution of the implicit state machine over thousands of links, makes editing a challenge; we know the conditions that the author imposed for this particular link, but may have no idea why those conditions were desirable or how following this link changes the state of other links. Some reasonable constraints – for example, that a link may be followed *n* times but not more – are difficult or impossible to express with guard fields. Other constraints that can be concisely stated in the story domain – *you cannot end Act I without establishing that there's a gun in the drawer* – must be enforced by multiple guard fields on many different links, and while these may not be particularly difficult for experienced writers to compose, understanding their purpose and intent can be a challenge to editors and critics who are asked to deduce the domain constraint from this distributed array of predicates.

4.2.Extending The Guard Field

The terseness of the original notation, so valuable in an era when memory capacity was scarce, has always baffled novices. Simple syntactic sugar can greatly clarify the notation:

Old: ("A" & (!"B")) | Anne

New: (visited(A) & unvisited(B)) | clicked(Anne)

The new notation is less esoteric and more readable. We can now cope with notes that have the same name; the guard field

visited(/Biographies/Washington)

is satisfied only after visiting the note named "Washington" that is located in the container, "Biographies." A common guard field

unvisited()

is satisfied when the link's destination has not previously been visited in this reading.

Since Storyspace 3 is built on the attribute-value store developed to support Tinderbox, we may easily extend guard fields to refer to generalized predicates. For example, if we use the note /Amy to keep track of the current state of the character named "Amy", then the guard field

$Cash(/Amy)>100

is satisfied if Amy has plenty of money, and the guard field

$Location(/Amy)!="Paris"

is satisfied if Amy is in London or Athens.

Storyspace 3 supports traditional guard field syntax as well by wrapping it in a new boolean function:

guard(*legacy guard-expression*)

Adding the wrapper when importing legacy documents is trivial, and in this way existing Storyspace hypertexts continue to operate as they always have while writers are offered a variety of new notational opportunities.

5. ASSERTIONS AND REQUIREMENTS

The Storyspace tradition of hypertext fiction has conducted a long dialogue with the separate tradition of instrumental *interactive fiction* growing originally from Crowther and Woods' *Adventure* [16] [35] [37]. As a rule, interactive fictions use links to vary what takes place in the narrative world, while hyperfiction more frequently uses links to vary the way underlying events are described: interactive fiction generally focuses on *story* while hyperfiction has predominantly focused on *plot* [10]. Storyspace accepts (and helped create) this framing in its use of guard fields that enable or disable individual links, thus determining whether a node or *writing space* can be seen now, or if access to it must be deferred. Exceptions to these inclinations abound, but the differing emphases on plot and story, *suzjet* and *fabula*, cannot be mistaken.

It is interesting to note in passing that the concerns of adaptive hypertext [20] are more closely allied with those of interactive fiction than with hyperfiction. In interactiuve fiction, we test whether the reader has acquired the Golden Key to decide whether or not they may pass to the second level of the adventure; in adaptive hypertext, we test whether the student has mastered arrays before they can proceed to study stacks and queues. In each case, we want to preclude access to a lexia until specified preconditions have been satisfied.

In conventional Storyspace hypertexts these preconditions must be checked on every link to the restricted writing space. This is certainly possible, but it is not always convenient, and the requirement is easy to overlook when revising the hypertext. Storyspace 3 augments guard fields with an additional predicate, **$Requirements**, for each writing space. If a note has requirements, they must be met before any incoming link can be traversed. Typical requirements are very much like guard fields: the requirement

unvisited(this)

asserts that this writing space can only appear once in any reading. We may also, as in guard fields, interrogate state variables the writer has chosen to use. A note with the requirement

$Cash(/Amy)>100 | $Cash(/May)>100

can be read if Amy or May have plenty of money.

Note that if a writing space has no requirements, link behavior is unchanged from Storyspace 1. Since no writing spaces created with Storyspace 1 have any requirements, Storyspace 3 performs them without change and without a separate legacy or compatibility mode.

When a link is successfully traversed, Storyspace 3 records that the note has been visited and increments the note's counter, **$Visits.** In addition, the note may have an $OnVisit action that asserts changes to the document state. For example, the action

$Cash(/Amy)=$Cash(/Amy)-50

reduces Amy's cash balance, and

$Score(/ArrayQuiz)=$Score(/ArrayQuiz)+1

gives the student-reader credit for a correct answer.

6. EAGER LINKS

A number of early hypertext formalisms envisioned a multi-pane or multi-window collage of panes in which specified transitions might occurs as soon as their preconditions were met[42]. Tim Oren's GUIDES, for example, embodied animated characters who could, through gesture or expression, indicate willingness to discuss a topic raised in the text [39], and the generalization of this formalism to encompass arguments among the guides themselves was readily foreseen. "Conversations With Friends" [4] distinguishes between *eager* links, which would lead a character to speak up immediately when their preconditions were satisfied, and *timid* links, which would simply lead the character to seek attention.

Storyspace 3 extends the $Requirements mechanism by providing *shark* links. If a note's requirements are satisfied, Storyspace additionally checks to see if any shark links lead away from the note. If an outbound shark link exists and if it can be followed – if its guard field and its destination's $Requirements are satisfied – then the shark link is followed immediately.

Just as $Requirements simplify guard fields by allowing the writer to refactor terms shared by all a note's inbound links, shark links provide convenient exception handling. Suppose that a character is to board a steamship, and that it is necessary that we actually see them purchasing a ticket. If they already have purchased a ticket, they may proceed on board. If the reader's trajectory has no yet encompassed a scene in which the character obtains a ticket, a shark link may interpolate here a trip to the ticket office. The same effect could be obtained with multiple guarded links, but at the cost of added complexity.

7. SCULPTURAL HYPERTEXT

Sculptural hypertext [36][6] was originally proposed as a radical, exotic alternative to familiar note-and-link hypertext. Storyspace 3 incorporates a flexible sculptural hypertext mechanism within the familiar formalism of Storyspace.

Storyspace hypertexts offer both text links – links anchored to text spans – and *plain links,* which are notionally anchored to the writing space as a whole. Plain links for each note are kept in an ordered list. If a reader clicks on a word not otherwise linked, or if she presses the [Return] key, Storyspace follows the highest-priority plain link which has a satisfied guard field. Only if there are no satisfied plain links does Storyspace require an explicit selection.

In Storyspace 3, we can go even further. If the reader has not clicked on a text link, and if no basic links are found, we next examine the value of the *current deck,* a list of string tokens. If the current deck is empty (as it is in all Storyspace 1 documents), Storyspace 3 waits for an explicit selection. If the current deck is not empty, however, Storyspace 3 gathers a pool of all notes for which

- the note's $Deck has a term in common with the *current deck*
- the note's $Requirements are satisfied.
- the note is unvisited or, if no eligible note is unvisited, the note has not been visited more than any other eligible note.

If more than one such note is found, one note is chosen at random from the eligible set, and that note becomes the destination.

Though sculptural hypertext has not yet proven to be of great interest to hypertext research, it has become a staple of literary games – particularly through Failbetter's *Fallen London* [1] and more broadly through narrativist games like Morningstar's *Fiasco* or Czege's *My Life With Master* (see [33]).

8. GENERALIZED STRETCHTEXT

When hypertext systems were more often discussed and their design more energetically debated, stretchtext – epitomized by Peter J. Brown's Guide [13] – was generally viewed as inherently in opposition to node-link hypertext like Storyspace. Despite the enthusiasm in early hypertext research for formalism [26], the formal properties of complex stretchtext networks were never thoroughly elucidated, and a late effort to reconcile stretchtext with more familiar paradigms [10] attracted scant notice.

After a long quiescence, however, interest in stretchtext has increased among the vernacular literary games and IF communities[21]. *Pry,* a novella by Danny Cannizzaro and Samantha Gorman, is a stretchtext fiction that reflects concepts originally proposed in Fluid[43], and a pattern library of Stretchtext idioms found in TWINE fiction is in preparation [14].

Though Storyspace 3 strives to avoid modes, combining all its extensions in a single formalism, one modality cannot be avoided. The Storyspace reader clicks to follow links, but the writer and editor must be allowed to click to select and revise text. Storyspace 3 leverages this long-extant and seemingly-inescapable modal behavior to support generalized stretchtext through macro expansion.

When writing, we may insert placeholders that can be interpreted by the performance engine. For example, the placeholder

^include(/sayings/Cicero)

will be replaced, in the reader's view, by the text of the note Cicero in the container "sayings". Similarly,

^replace(anchor,note,*action*)

will embed a link with the specified anchor, If the link is clicked, the anchor text will be replaced by the contents of the designated note, and an optional action may be performed in order to record a change of state.

9.REPRESENTATION

9.1.Document Representation

The central issue confronting the original Storyspace document format was speed of loading and saving documents. Storyspace originally ran on 6mHz 68000 processors equipped with a slow 800K floppy disk drive. Data transfer alone required most of the resources of the computer, and hence it was vital that additional processing be minimized. Even then, performance was barely adequate; Stuart Moulthrop's *Victory Garden* [38], a hypertext of 986 lexia, 2804 links, and 96,000 words, originally required five minutes to load[1]. In addition, the limit imposed by 800K disk capacity place a premium on compactness.

These design forces led Storyspace 1 to adopt a file format which was little more than a flattened representation of structs as they appeared in memory. These were segregated into file chapters, beginning with an introductory header struct that listed offsets to each chapter – the collection of node descriptors, the collection of link descriptors, the text heap, and so forth. Pointers were replaced by fixed element IDs but little additional processing was required in order to read or write the document file.

This file representation proved satisfactory for many years, and continues to be supported. Its weakness, however, lay in its fragility. If a file was damaged by software error or media defect, recovery was not much easier than manually reconstructing the memory image of the running program. In particular, if any of the offsets recorded in the header were incorrect, the entire file would be rendered unreadable.[7]

The passage of time and operation of Moore's Law transformed the design forces that impact the document's external representation. Processors are orders of magnitude faster, and even laptop and tablet computers make additional processors available for compute-bound tasks. We no longer labor to squeeze a novel onto a floppy disk when even a mobile phone can easily store a library of tens of thousands of books. Storyspace 3 thus follows Tinderbox in adopting Tinderbox's XML representation for its files[8]. XML is not notably compact – *Victory Garden,* for example, grows from 800K to 40 Mb – but this file size (and its 2s load time) are negligible concerns.

9.2.Internal Representation

Performance concerns also mandated that Storyspace 1 represent its nodes and links as static frames with fixed offsets so that access to any facet of any node required no more than simple pointer arithmetic. In consequence, a number of Storyspace facilities were constrained to use fixed buffers; note titles, for example, were originally limited to 32 bytes. (In addition to the constraints imposed by performance, it should be kept in mind

[1] These performance concerns were by no means unique to Storyspace. Students using Intermedia for their coursework habitually brought a book to the computer lab, the better to pass the delays imposed by database latency[41].

that no general-purpose language besides LISP at that time possessed what would today be regarded as even a rudimentary string library.)

In addition, implicit considerations of the personal computing environment led to plausible design assumptions that, with the passage of time, became obsolete and even risible. The Macintosh screen had a fixed height of 342 pixels, sufficient to display perhaps 30 lines of rendered type in a space of just under five inches (12cm). Scrolling a few screenfuls of text was a reasonable compromise, though some early writers preferred to avoid scrolling entirely, but the notionally-infinite plane in which all Macintosh images were rendered was limited in practice to 32,767 pixels. That amounted to nearly 100 screens of text, which seemed both amply in principle and approaching the capacity of contemporary scroll bars to control. In time, both screens and documents became larger; ultimately, the size of the graphic plane became a real constraint on the Storyspace outline view, which could only display a few thousand writing spaces. This was of little concern, of course, when 1000-node hypertexts were at the outer bound of feasibility, but the constraint persisted into the 21st century. Here, too, a reimplementation removes difficulties that the quirks of early systems arbitrarily imposed.

Modern processors and programming environments present Storyspace with far different constraints. Hypertext nodes are represented as attribute-value stores with prototype inheritance. Almost no caching or performance optimization is required to obtain adequate performance.

10.CONCLUSION

A single summer's development campaign by a single developer – a developer who could not in this time be relieved of other commitments – sufficed to reimplement Storyspace for OS X *El Capitan.* Much has been written about preservation and archeology of digital literature (see [32] [17]), and this effort concretely improves the accessibility of a number of hypertexts about which much has been written. Additional benefits of the reimplementation include greatly improved typography and enactment and improved accessibility for readers with visual or motor impairment.

A variety of new facilities have been added to Storyspace, providing support for interactive fiction, sculptural hypertext, and generalized stretchtext without introducing additional operating modes or affecting the simplicity of the underlying link model [19].

The success of a writing tool depends on the success of the work written with its aid. That success was clear, in the end, for the original Storyspace. Though this is the 27th ACM Hypertext Conference, we arguably write fewer significant hypertexts today than we wrote at the time of the tenth. "Where again," I might ask, "are the hypertexts?" Hypertext broadly, and hypertext fiction specifically, were for some years the target of a reaction against the commercial internet[15], against corporate publishing [12], or against postmodernism. Those battles have been lost and won: perhaps it is time we once more picked up our virtual pens.

11.ACKNOWLEDGMENTS

I am grateful to many correspondents for assistance on topics addressed in this paper, including Jay David Bolter, Michael Joyce, Matt Kirschenbaum, George P. Landow, Stacey Mason, Janet Murray, and Emily Short. Brent Simmons and Lee Hasiuk provided welcome technical advice during implementation. Mark Anderson provided many useful corrections and suggestions for *Getting Started With Hypertext Narrative,* on which this paper is largely based.

12.REFERENCES

[1] Alexis Kennedy, E. "Fallen London".

[2] Atzenbeck, C., Bernstein, M., Al-Shafey, M. A., and Mason, S. 2013. "TouchStory: Combining Hyperfiction and Multitouch Proceedings of the 24th ACM Conference on Hypertext and Social Media". *HT '13*. 189-195.

[3] Barret, B. 2012 *Memory Machines: the evolution of hypertext*. Anthem Press.

[4] Bernstein, M. 1995 *Conversations With Friends: Hypertexts With Characters*. In Hypermedia Design, S. Fraïse, F. Garzotto, T. Isakowitz, J. Nanard, and M. Nanard, eds. Springer.

[5] Bernstein, M. 1998. "Patterns of Hypertext". *Hypertext '98*. 21-29.

[6] Bernstein, M. 2001. "Card Shark and Thespis: exotic tools for hypertext narrative". *Hypertext 2001: Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*. 41-50.

[7] Bernstein, M. 2002. "Storyspace 1". *Proceedings of the 13th ACM Hypertext Conference*. 172-181.

[8] Bernstein, M. 2003. "Collage, composites, construction". *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*. 122-123.

[9] Bernstein, M. 2010. "Criticism". *Proceedings of the 21st ACM conference on Hypertext and hypermedia*. 235-244.

[10] Bernstein, M. 2009. "On Hypertext Narrative". *ACM Hypertext 2009*.

[11] Bernstein, M., Joyce, M., and Levine, D. B. 1992. "Contours of Constructive Hypertext". *European Conference on Hypermedia Technology*. 161.

[12] Birkerts, S. 1994 *The Gutenberg Elegies*. Faber and Faber.

[13] Brown, P. 1991. "Higher Level Hypertext Facilities: Procedures With Arguments". *Hypermedia*. 3, 2, 91-100.

[14] Dias, B. 2016. "A Garden of Devices in Dynamic Prose". *in preparation*.

[15] Carr, N. G. 2010 *The shallows : what the Internet is doing to our brains*. W.W. Norton.

[16] Crowther, W. and Woods, D. 1976. "Adventure".

[17] Grigar, D. and Moulthrop, S. A. "Pathfinders".

[18] DeYoung, L. 1990. "Linking Considered Harmful". *ECHT'90 - Hypertext: Concepts, Systems and Applications*. 238-249.

[19] E. James Whitehead, J. 2002. "Uniform Comparison of Data Models Using Containment Modeling". *Hypertext '02*. 182-191.

[20] E. Knutov, P. De Bra, and Pechenizkiy, M. 2009. "AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques". *New Review of Hypermedia and Multimedia*. 15, 5-38.

[21] Short, E. 2016. "Set, check, or gate? A problem in personality stats".

[22] Feathers, M. C. 2005 *Working effectively with legacy code*. Prentice Hall Professional Technical Reference.

[23] Fowler, M. 1999 *Refactoring*. Addison-Wesley.

[24] Freeman, S. and Pryce, N. 2010 *Growing object-oriented software, guided by tests*. Addison Wesley.

[25] Gaffney, C., Conlan, O., and Wade, V. 2014. "The AMAS Authoring Tool 2.0: A UX Evaluation Proceedings of the 25th ACM Conference on Hypertext and Social Media". *HT '14*. 224-230.

[26] Halasz, F. and Schwartz, M. 1994. "The Dexter Hypertext

Reference Model". *Communications of the ACM*. 37, 2, 30-39.

[27] Hargood, C., Davies, R., Millard, D. E., Taylor, M. R., and Brooker, S. 2012. "Exploring (the Poetics of) Strange (and Fractal) Hypertexts Proceedings of the 23rd ACM Conference on Hypertext and Social Media". *HT '12*. 181-186.

[28] Bolter, J. D. and Joyce, M. 1987. "Hypertext and Creative Writing". *Hypertext '87*. 41-50.

[29] Joyce, M. 1990. "afternoon, a story".

[30] Joyce, M. 1988. "Siren Shapes: Exploratory and Constructive Hypertext". *Academic Computing*. 11 ff.

[31] Kerievsky, J. 2005 *Refactoring to patterns*. Addison-Wesley.

[32] Kirschenbaum, M. G. 2008 *Mechanisms : new media and the forensic imagination*. MIT Press.

[33] Bernstein, M. 2016 *Getting Started With Hypertext Narrative*. Eastgate Systems, Inc.

[34] Martin, R. C. 2009 *Clean code : a handbook of agile software craftsmanship*. Prentice Hall.

[35] Merrit Kopas, E. 2015 *Videogames For Humans*. Instar Books.

[36] Millard, D. E., Hargood, C., Jewell, M. O., and Weal, M. J. 2013. "Canyons, Deltas and Plains: Towards a Unified Sculptural Model of Location-based Hypertext Proceedings of the 24th ACM Conference on Hypertext and Social Media". *HT '13*. 109-118.

[37] Montfort, N. 2003 *Twisty little passages : an approach to interactive fiction*. MIT Press.

[38] Moulthrop, S. 1991 *Victory Garden*. Eastgate Systems, Inc.

[39] Oren, T., Solomon, G., Kreitman, K., and Don, A. 1990 *Guides: Characterizing the Interface*. In The Art of Human Computer Interface Design, B. Laurel, ed. Addison Wesley.

[40] Farge, P. L. 2008. "Luminous Airplanes".

[41] Smith, K. E. and Zdonik, S. B. 1987. "Intermedia: A case Study of the Differences Between Relational and Object-Oriented Database Systems". *OOPSLA 87*. 22(12), 12, 452-165.

[42] Stotts, P. D. and Furuta, R. 1989. "Petri-net based hypertext: Document structure with browsing semantics". *ACM Transactions on Office Information Systems*. 7, 1, 3-29.

[43] Zellweger, P. T., Mangen, A., and Newman, P. 2002. "Reading and Writing Fluid Hypertexts". *Hypertext 2002*. 45-54.