Mother – An Integrated Approach to Hypertext Domains

Claus Atzenbeck*
Hof University
Institute of Information Systems
Hof, Germany
claus.atzenbeck@iisys.de

Daniel Roßner
Hof University
Institute of Information Systems
Hof, Germany
daniel.rossner@iisys.de

Manolis Tzagarakis
University of Patras
Department of Economics
Rion, Greece
tzagara@upatras.gr

ABSTRACT

The idea to associate information with so-called *links* was developed by hypertext pioneers in the 1960s. In the 1990s the *Dexter Hypertext Reference Model* was developed with the goal to provide a general model for node-link hypertext systems. In the 1990s and 2000s there were important steps made for hypertext infrastructures, which led to *component-based open hypermedia systems* (CB-OHS).

In this paper we provide a detailed description of node-link structures. We argue that Dexter does not match the need of CB-OHS, as it supports a mix of multiple structure domains. Based on the implementation of link support in our system *Mother* we demonstrate how Dexter needs to be tailored accordingly. We further describe *Mother's* ability of node-link structures to interoperate with other available structure services and vice versa.

CCS CONCEPTS

• **Human-centered computing** → **Hypertext / hypermedia**; • **Software and its engineering** → *Software infrastructure*;

KEYWORDS

hypertext infrastructure; open hypermedia systems; CB-OHS; navigational hypertext; *Mother*; *Asgard*; *Midgard*; *Hel*

ACM Reference Format:

Claus Atzenbeck, Daniel Roßner, and Manolis Tzagarakis. 2018. Mother – An Integrated Approach to Hypertext Domains. In *HT '18: 29th ACM Conference on Hypertext and Social Media, July 9–12, 2018, Baltimore, MD, USA*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3209542.3209570>

1 INTRODUCTION

People have been referencing texts since they invented writing. The idea to associate information with so-called *links* was developed much later: A little more than 70 years ago, Vannevar Bush, who is referenced in the scientific community as one of the “fathers of hypertext”, proposed the idea of “associative trails” in his system *Memex* [6]. This idea was adopted and further developed by various computer scientists, e. g., Doug Engelbart [11] or Ted Nelson [28]. In the 1980s many academic and commercial systems became available, including *KMS* [1] or *NoteCards* [16]. Their main paradigm is based

on nodes interconnected via links. Because users can “navigate” from node to node, this type of hypertext is called *navigational hypertext*. In the late 1980s the *Dexter Hypertext Reference Model* [14] finally provided a general model of node-link hypertext systems.

Although other hypertext paradigms (such as spatial hypertext [20] or argumentation support [7]) have been introduced about the same time, the predominant association mechanism was and still is navigational hypertext. Its level of awareness further increased by the raising popularity of the World Wide Web in the 1990s.

At that time research on hypertext systems evolved even beyond today's WWW. Examples include *Hyper-G* [22] or *Microcosm* [18]. In an interview Andy van Dam describes the Web even as “the lowest common denominator of hypertext” [3]. By the end of the 1990s and early 2000s systems like *Construct* [43] or *Callimachus* [40] also supported non-navigational structures. Due to their open and component-based architecture they were called *Component-based Open Hypermedia Systems* (CB-OHS). These systems – as a consequence of supporting navigational and non-navigational structures within the same framework – raised the issue of interoperating structure services, i. e., combining different structure services.

Years later, our system (which we call *Mother*) with its structure-aware layer *Asgard* [4] became the first component-based open hypermedia system that was built primarily with the most challenging to handle hypertext structure type in mind: spatial structures. In this paper we describe *Mother's* support for navigational structures and its ability to interoperate with other structure services.

2 PROBLEM ANALYSIS

Navigational hypertext uses a set of basic abstractions, including nodes, links, and anchors. A node is a container of information; links connect nodes, while anchors designate areas inside a node's content from which a link originates or lands, creating a graph of nodes. For users, links function simply as jump-addresses. Such navigation from node to node via anchors and links is the primary behavior exhibited by navigational hypertext. Besides functioning as addresses, links attempt also to capture a semantic relationship between connected nodes. Over the years a considerable number of systems emerged offering the necessary tools and models to support the authoring of navigational hypertext. Although the node-link paradigm in navigational hypertext sounds simple as a model, the developed systems offered nuanced versions in order to address the needs of their respective application domains.

While the first generation of systems (such as *NLS/Augment* [12] or *FRESS* [44]) demonstrated the concept of hypertext, it was the second generation of systems (such as *NoteCards* [16] or *KMS* [1]) that offered navigational hypertext in a way that is similar to contemporary systems in terms of model and architecture [15]. In *NoteCards* nodes came in the form of cards, which could be typed

*Corresponding author

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

HT '18, July 9–12, 2018, Baltimore, MD, USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5427-1/18/07.

<https://doi.org/10.1145/3209542.3209570>

to capture semantics while links were directional, typed, and anchored to the entire destination card. In addition, cards and links were programmable allowing the user to modify their behavior such as when and how to display the card's content. KMS offered similar abstractions as NoteCards, but did not embed links within the content of the nodes. Despite their emphasis on the node-link model these systems introduced additional abstractions that allowed the creation of hierarchical structures, for example, nodes of type *FileBox* in NoteCards or *Tree-items* in KMS.

Shortcomings and limitations of the second generation systems were identified rather early [15]. However, instead of concentrating at the identified limitations, the hypertext systems community moved into a different direction: to establish a common vocabulary with which the various existing systems could be conceived. The Dexter Hypertext Reference Model [17] originated from such concerns and aimed at providing a common formal conceptual model with which the diversity of existing navigational hypertext systems could be discussed, compared, and made to interoperate.

Dexter divides a hypertext system into three layers and specifies the roles of each layer. These layers are: (i) the *Storage layer*, responsible for the persistent storage of the objects that make up the (navigational) hypertexts; (ii) the *Within-Component layer*, responsible for the contents of the objects; and (iii) the *Runtime-layer*, responsible for handling the hypermedia objects (such as nodes, links, anchors, etc.) at runtime. Dexter defined also the interfaces between these layers: the anchors as the interface between the storage and the within-component layers and presentation specifications as the interface between the runtime and storage layers.

From a data model point of view, Dexter introduces a generic abstraction of component that can be specialized as nodes, link components (which contain presentation and anchor specifier representing links), or composite components which allows the creation of hierarchical structures. Links in the Dexter model can be single or bi-directional and *n*-ary, meaning that a single link component may connect a node to more than one destination nodes.

At around the same time, systems were developed to enable navigational hypermedia at a large and even global scale, overcoming the boundaries of the user's local computer. The WWW was one of the first attempts to offer navigational hypermedia at a global scale. WWW links are single-directional and embedded into the nodes' content. A global reference and naming scheme of nodes allows the WWW to create structures that span across systems. Despite the apparent simplicity of the model, the WWW model comes with a set of technologies that allow developers to extend and tailor the functionality of nodes and links as needed, e. g., HTML or JavaScript. From a navigational hypermedia point of view, the model has little to offer with heavy criticisms aimed towards its trivial model and limited structure awareness [30].

3 INFRASTRUCTURE DISCUSSION

In our recent paper [4] we propose a component-based open hypermedia system, which we call *Mother*. It consists of three layers: (i) *Midgard*, hosting all *applications* that count as user interfaces or have similar functionalities; (ii) *Asgard*, a collection of Mother's (partly intelligent) components that deal with *structures*; and (iii) *Hel*, all *knowledge-based* components.

Most previous CB-OHS were designed primarily targeting the needs of node-link structures. Mother's architecture "has been developed primarily with one of the most complex [...] hypertext structures in mind. This approach appreciates the specific demands of implicit structures rather than explicit ones" [4]. Such implicit structures are supported by a spatial structure service in Asgard. It includes several multiple, highly specialized parsers (spatial, visual, temporal) that are capable of analyzing spatial structures similar to how humans would interpret them [35].

For this paper our focus is on sophisticated support for explicit structure types and structure interoperability. In particular, we aim for support of navigational structures. The Dexter model subsumes most hypertext systems at that time and even goes beyond the Web as we know it today. However, we need to bear in mind that Dexter was developed years before the first CB-OHS; thus, it is not a surprise that the model does not reflect a component-based view on systems. Because of that, we had to analyze Dexter and identify how to tear it apart and reassemble it in such a way to let it fulfill the requirements of a CB-OHS – in particular of Mother.

In the 1980s many systems combined various structure domains in a *single, monolithic model*. For example, KMS distinguishes between tree (i. e., hierarchical) and annotation (i. e., node-link) structures, whereas tree structure were dominant in usage at that time [1]. On the contrary, CB-OHS are built to support *arbitrary* structure domains within a *single architecture*, each provided by its own structure service. Such services may interoperate with each other.

In the following we take the Dexter node-link model and use only those parts that let us design a pure navigational service. Any other structure abstraction is put into its appropriate structure domain.

An important class in Dexter is named *Component* [15], referring to hypertext nodes. Links are components with endpoints. This makes links first-class objects in Dexter. At that point Dexter already provides full support of navigational hypertext. However, it also offers composites for hierarchical structures. Those differ in their behavior compared to node-link structures, e. g., removing a composite would recursively remove also the containing items.

As tree structures are supported by its own hierarchical structure service in Mother, we removed the notion of composites from our node-link support. Similar to Dexter, in Mother we also consider links as components that have a number of endpoints associated. However, components cannot contain other components, which would be a hierarchical relationship and thus not within the scope of the navigational domain. As components are not organized hierarchically by our node-link model, we also ignore Dexter's parentID attribute in Mother's component class.

Endpoints in Dexter have types that provide information about the link direction; those are SOURCE, DESTINATION, or BIDIRECT. The latter would be used for bidirectional links. Dexter also defines the direction NONE [17], however, we support dropping this type as argued in [15]. In Mother's link model an endpoint type is named *direction* (instead of *type* as in Dexter) in order not to be confused with *typed links* (see, e. g., [37]); those would have semantics/metadata attached to links. For such cases, Mother provides a specific metadata service, which is capable of adding arbitrary key-value pairs to any component, including links. In Mother metadata is considered a separate structure domain. Discussions about metadata services can also be found in previous publications (e. g., [27, 42]).

Dexter also includes presentation specifications (i. e., information about how components are to be displayed) and content in its model. In Dexter the data itself stays inside the within-component layer is not managed by the model itself. In Mother we removed both presentation specification and content from the node-link model. Those are handled by Midgard applications (see also discussion in [4]). Content (i. e., data from this point of view) lies outside Mother; the system keeps its focus on structure instead.

In summary, we use a subset of Dexter's model abstractions, removing any part that is not specific to navigational structures, including (i) composites; (ii) presentation specification; and (iii) content. Instead of overloading Mother's navigational hypertext model beyond the requirements specific to its structure domain, we offer support for hierarchical and metadata structures via specialized services. Content stays outside the system and is consequently handled by Mother's Midgard layer applications. This keeps the individual structure models simple (similar to the Web's simplistic node-link model), while granting sophisticated and flexible services.

Most of the CB-OHS mentioned in this article own a separate structure-aware node-link service. In Mother the appropriate layer for this would be Asgard. However, reducing Dexter also transformed our idea of a navigational structure services. The reason is as follows: As explained above, Mother is not content-aware. Instead, Midgard applications handle the content. The knowledge layer Hel holds all knowledge as well as linkbases, of which Mother supports an arbitrary number. This is beneficial, for example, if one wants to use multiple linkbases with different access rights as it eases implementation and lowers security risks. In Mother links are separate objects, however, the pointer to the respective resources are represented as strings, for which we suggest to use URIs [5]. Since Mother provides full node-link support at Midgard und Hel layers, there is nothing left for Asgard to take care of.

The main goal in our Dexter-based design was to cut out complexity while keeping the service as generic as possible. In order to achieve this, we distinguish between *linkbase* and *linkbase API*. The latter provides an interface to the linkbase to communicate with and – more important – propagates URIs to identify links and endpoints. Mother provides support for multiple linkbases.

The main task of a linkbase is to manage queries for existing links or add new ones. We chose a relative data structure over a graph based, because the corresponding nodes are identified by a unique string-based ID, optionally including anchor information. Nodes themselves are not part of the link data, therefore, the only item remaining to store is a list of links, kept inside a table with specific indexes to speeds up access routines. It is important to note, that the linkbase itself does not restrict the semantics of IDs and anchors. It only requires them to be UTF-8 encoded strings. Their meaning is maintained by an authority outside the scope of Mother. The linkbase is responsible for the creation of link IDs, guaranteed to be unique in the scope of a single linkbase instance only. Simply speaking, the *linkbase* defines how links are stored and makes very few assumptions about what links are to users or software. On the other hand, the *linkbase API* defines what a link is, how the parts of a link can be interpreted, and how the communication with the linkbase works (for altering, adding, or retrieving links).

We evaluated different communication protocols. A link server should be easy to integrate in any kind of existing system, and be

ready for future improvements, e. g., authentication or encryption. In 1996, [8] came up with a draft proposal for the standard Open Hypermedia Protocol (OHP), which emerged from many discussions within the Open Hypermedia Systems Working Group (OHSWG) about interoperability of distinct but similar hypertext systems [10].

From that on, OHP improved significantly. As a result, OHP was split into many different interfaces. The OHP Navigational Interface (OHP-Nav) has been one of them [24]. It describes a common hypertext data model for navigational hypertext systems including a set of operations [9]. However, the development stopped in the early 2000s with an important unresolved issue: Should the components talk to each other through a programming API (e. g., CORBA) or rather via an on-the-wire communication model [25]? This and the fragmented documentation led us to the decision to keep the „good“ parts (e. g., the data model) and use those along with HTTP. Because there are many other relevant protocols, we cannot provide an exhaustive comparison in the scope of this paper. With respect to the benefits of using HTTP for inter-application communication [13], the main advantage is its similarity in operations compared to OHP: The OHP operations *Create*, *Get*, *Update*, and *Delete* [9] perfectly map to the HTTP verbs *PUSH*, *GET*, *PUT*, and *DELETE* and hence make HTTP a suitable protocol for our purpose.

The chosen communication model combined with the OHP data model is not sufficient for a full featured API. Additionally, Mother's architecture has to be taken into consideration. In 2000, Fielding presented REST as an „architectural style for distributed hypermedia systems“ [13]. We adopt the idea by using URIs for identifying endpoints and links in a global context. Because the linkbase only provides local IDs, the API offers them as global and unique URIs.

With such unique IDs it is easy to use HTTP verbs to manipulate or add links. Confirming to REST, we do not specify the media type of the answer. It is up to the linkbase API to answer in a way the client understands (e. g., XML, HAL+JSON) and vice versa. For users it is irrelevant to know how links are stored; they just need the URI and have support for the required media types. Furthermore, REST demands „Hypermedia As The Engine Of Application State“ (HATEOAS), which decouples the server's functionality and the client's understanding of it. This means, that the client only needs to know the entry point. This design allows the API (provided that there is a smart client on the other side) to change over time. Adding or removing new functionalities become possible. The overall design described above leads to a flexible and simple to implement solution for both servers and clients.

For demonstration purposes we implemented a simple browser add-on for Chrome and Firefox, offering rich linking features to the user. The version of the software observes the URI of a visited site and queries for existing links, which use this URI as an endpoint. If this is the case, the user gets an unobtrusive notification within the add-on section of his browser. From there he can follow or change the links. Furthermore he can add new links, which are by design visible to everybody who has access to the same linkbase. The next iteration of this add-on should be able to inject links directly into the visited resource. Unfortunately, it is difficult to define endpoints appropriately for dynamic Web pages, as their content may change frequently. It may be necessary to provide unchangeable snapshots of such resources or offer the possibility to compute semantical anchors instead of syntactical ones [cf. 41].

4 SCENARIO

In this section we aim at providing a case scenario for various services. Some is still work in progress. Imagine two researchers, Sarah and Otto, working jointly on a scientific publication. So far they mainly used (i) e-mail to discuss early ideas; (ii) Web browsers to search for related articles; (iii) pen and paper to scribble ideas and get a first glimpse of potential relationships; and (iv) word processors to write hierarchical outlines. There are gaps between different media used; e. g. Otto's used Web search engine or word processor is unaware of the paper on which Sarah scribbles first ideas or their e-mail discussions. There is only little support for structuring across different media or application domains.

Spatial hypertext. Now imagine Sarah and Otto using Mother to discuss and structure their ideas related to the publication. Furthermore, they use the system to also query information they want to further use. In order to accomplish that, Sarah and Otto use Mother's spatial hypertext application Mindspace. While structuring ideas on Mindspace's UI, the system uses intelligent methods to propose other relevant information and puts that on the space.

Spatial hypertext + hierarchies. Some spatial hypertext applications (like Viki [21] or VKB [36]) offer collections. Those can be thought as spaces in which subspaces are located. In Mother the spacial structure services does not have a notion of hierarchies. When Otto starts organizing spaces hierarchically, the hierarchy itself is handled by its own, highly specialized service.

Spatial hypertext + links. Otto finds an interesting object on the 2D space in Mindspace which was acquired from the knowledge base. The information represented by the object was automatically assembled by text mining methods harvesting the Web. Furthermore, it contains links to the original sources. Otto traverses those links in order to read and understand the original information. Furthermore, Otto adds links to objects in Mindspace manually. Those are stored in the linkbases.

Links + Web pages. Otto browses many relevant Web pages. Mother's link functionality lets him add unidirectional or bidirectional links to/from those even if he does not have write access on the Web pages. Since links are first-class objects in Mother, Otto can even see which resources link to the current Web page. With this information Otto navigates to relevant Web pages much quicker.

Metadata + spatial hypertext + links + Web pages. Otto grants Sarah access to his information in Mindspace. Sarah does not want to change the spatial structure, thus she annotates some objects instead. For that she uses Mother's metadata service. She also annotates some of Otto's links that she retrieved from the linkbase. For example, she adds types like "contradicting" or "supporting" to links in order to provide an understanding of the kind of relation to the linked Web page. Similar to that, Sarah also annotates Web pages she reads. Since both Sarah and Otto have access to the same resources within Mother, any of Sarah's annotations is also displayed to Otto, regardless of the Midgard tool used.

Hierarchy + Web pages + metadata. While browsing the Web for relevant information, Otto realizes that it would be beneficial to organize Websites hierarchically. For that his Web browser uses a plugin that connects to Mother's hierarchy structure service. Furthermore, Otto and Sarah use the metadata service to annotate the various hierarchy levels to indicate priorities or required resources.

5 CONCLUSION AND FUTURE WORK

In line with our previous work [4], we discussed in this paper steps toward Dexter-based navigational hypertext support in Mother. We argued why and how we reduce Dexter to match the needs of a CB-OHS. We also state that – in contradiction to other CB-OHSs – the node-link service became so much reduced that there was no reason to keep it as a separate Asgard component. Instead, node-link support can be fully handled by Hel.

We also mentioned support for spatial structures (discussed in detail in [4]), hierarchical structures, and metadata. Furthermore, this paper uses a case scenario to describe how different services can be reused in various combinations. This opens a higher level of inter-domain interoperability and lets users add or combine arbitrary structure services supported by the respective Midgard applications. We argue that this high level of freedom is not possible with super-structures, as proposed in the past (e. g., Dexter or FOHM [26]), but rather with highly specialized services of which each supports only a single structure domain, following the UNIX philosophy: "Make each program do one thing well" [23].

There are a number of open issues related to the development and also a number of open research questions. For example, we need to make objects handled by the spatial hypertext application Mindspace globally addressable in order to allow links pointing to them.

Regarding links it is still an open issue how to synchronize link anchors (which are stored within Mother's link base) and dynamic node content. The Midgard application (e. g., the browser plugin) needs to resolve the new anchor position. This is in particular difficult in cases when the Midgard application does not have unique control over node contents.

There is also the need to improve Mother's node-link support. For example, we do not consider security issues yet, e. g., access rights. Furthermore, we could extend our list of anchor direction markers with HIDDEN, as proposed by [15]. This would let users create a "blind" endpoint that exists but would not appear in any operation.

Regarding the node-link support in Mother it is still an open issue of whether we need to implement a navigational structure service within Asgard. It is likely that we want to offer computational services over networks or introduce node-link specific behavior at some point. Both would be good reasons for creating an Asgard service.

Mother is a CB-OHS, consequently prioritizing structure rather than data. This makes it different to the data-driven Web. Data – managed by Midgard applications – is outside the scope of Mother. Even though Mother shows some advantages over other CB-OHS, it is still of the same kind. In the late 1990s a different paradigm found its way to the scientific community: *structural computing*, which claims that structure is more important than data [32]. Structural computing always has supported structures independent of its kind. It is the search for first-class structural abstractions as generalization of structure. Related discussions during the early 2000s created various positions and levels of understanding, including the question of whether structural computing is an evolution or a revolution [19], implications of structural computing to hypertext infrastructures [34], and various discussions regarding the role of

data, structure, and behavior within structural computing with the goal to create a coherent and global theory for it [2, 29, 33, 39].

A huge future step for Mother is extending its capabilities toward structural computing requirements. This could include, for example, introducing structural primitives (as proposed for Callimachus [38]) or supporting the EAD model (introduced by [31]).

Structural computing has the potential to lift hypertext systems to a higher and more sophisticated level of structure domain support and interoperability. Even though our CB-OHS Mother has already a very strong focus on structure, we believe that turning it into a structural computing environment would create huge benefits. This is where we want to go in a long run.

ACKNOWLEDGMENTS

This work is part of the ODIN project, sub-project HEIMDALL, funded by the German Federal Ministry of Education and Research (grant ID 03PSWKPD).

REFERENCES

- [1] Robert M. Akscyn, Donald L. McCracken, and Elise A. Yoder. 1988. KMS: a distributed hypermedia system for managing knowledge in organizations. *Commun. ACM* 31, 7 (1988), 820–835.
- [2] Kenneth M. Anderson, Susanne A. Sherba, and William V. Lephien. 2003. Structure and behavior awareness in themis. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*. ACM Press, 138–147.
- [3] Claus Atzenbeck and Mark Bernstein. 2018. Interview with Andy van Dam. *ACM SIGWEB Newsletter Winter* (January 2018).
- [4] Claus Atzenbeck, Thomas Schedel, Manolis Tzagarakis, Daniel Roßner, and Lucas Mages. 2017. Revisiting Hypertext Infrastructure. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media 28th ACM Conference on Hypertext and Social Media*. ACM Press, 35–44.
- [5] T. Berners-Lee, R. Fielding, U. C. Irvine, and L. Masinter. 1998. *Uniform Resource Identifiers (URI): Generic Syntax*. RFC 2396. Network Working Group.
- [6] Vannevar Bush. 1945. As we may think. *The Atlantic Monthly* 176, 1 (7 1945), 101–108.
- [7] Jeff Conklin and Michael L. Begeman. 1987. gIBIS: a hypertext tool for team design deliberation. In *Proceedings of the ACM Conference on Hypertext*. ACM Press, 247–251.
- [8] Hugh Davis, Andy Lewis, and Antoine Rizk. 1996. *OHP: A Draft Proposal for a Standard Open Hypermedia Protocol (Levels 0 and 1: Revision 1.2 – 13th March 1996)*. Technical Report. University of Southampton.
- [9] Hugh Davis, Siegfried Reich, and David Millard. 1997. A Proposal for a Common Navigational Hypertext Protocol. In *Proceedings of the 3.5 Open Hypermedia System Working Group Meeting*.
- [10] H. C. Davis, D. E. Millard, S. Reich, N. Bouvin, K. Grønbaek, P. J. Nürnberg, L. Sloth, U. K. Wiil, and K. Anderson. 1999. Interoperability between hypermedia systems: the standardisation work of the OHSWG. In *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*. ACM Press, 201–202.
- [11] Douglas C. Engelbart. 1962. *Augmenting Human Intellect: A Conceptual Framework*. Summary Report AFOSR-3233. Stanford Research Institute.
- [12] Douglas C. Engelbart. 1984. Authorship Provisions in AUGMENT. In *Proceedings of the COMPCON Conference, San Francisco, CA, Feb 27–Mar 1, 1984*. 465–472.
- [13] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. University of California.
- [14] Kaj Grønbaek and Randall H. Trigg. 1992. Design issues for a Dexter-based hypermedia system. In *Proceedings of the ACM Conference on Hypertext*. ACM Press, 191–200.
- [15] Kaj Grønbaek and Randall H. Trigg. 1996. Toward a Dexter-based model for open hypermedia: unifying embedded references and link objects. In *Proceedings of the 7th ACM Conference on Hypertext*. ACM Press, 149–160.
- [16] Frank G. Halasz, Thomas P. Moran, and Randall H. Trigg. 1987. NoteCards in a nutshell. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface (CHI '87)*. ACM Press, 45–52.
- [17] Frank G. Halasz and Mayer Schwartz. 1994. The Dexter Hypertext Reference Model. *Commun. ACM* 37, 2 (2 1994), 30–39.
- [18] Wendy Hall, Hugh Davis, and Gerard Hutchings. 1996. *Rethinking Hypermedia. The Microcosm Approach*. Kluwer.
- [19] David L. Hicks and Uffe K. Wiil. 2003. Searching for revolution in structural computing. *Journal of Network and Computer Applications* 26, 1 (2003), 27–45.
- [20] Catherine C. Marshall, Frank G. Halasz, Russell A. Rogers, and William C. Janssen. 1991. Aquanet: a hypertext tool to hold your knowledge in place. In *Proceedings of the 3rd ACM Conference on Hypertext*. ACM Press, 261–275.
- [21] Catherine C. Marshall, Frank M. Shipman, and James H. Coombs. 1994. VIKI: Spatial hypertext supporting emergent structure. In *Proceedings of the 1994 ACM European Conference on Hypermedia Technology*. ACM Press, 13–23.
- [22] Hermann Maurer. 1996. *Hyper-G Now HyperWave. The Next Generation Web Solution*. Addison-Wesley.
- [23] M. D. McIlroy, E. N. Pinson, and B. A. Tague. 1978. UNIX Time-Sharing System: Foreword. *The Bell System Technical Journal* 57, 6 (1978).
- [24] David Millard, Siegfried Reich, and Hugh Davis. 1998. Reworking OHP: the Road to OHP-Nav. In *Proceedings of the 4th Workshop on Open Hypermedia Systems at the ACM Hypertext '98 Conference*. 48–53.
- [25] David E. Millard, Hugh C. Davis, and Luc Moreau. 2000. Standardizing Hypertext: Where Next for OHP?. In *Proceedings of the 6th International Workshop and 2nd International Workshop on Open Hypertext Systems and Structural Computing*. Springer, 3–12.
- [26] Dave E. Millard, Luc Moreau, Hugh C. Davis, and Siegfried Reich. 2000. FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia*. ACM Press, 93–102.
- [27] Marc Nanard, Jocelyne Nanard, and Peter King. 2003. IUHM: a hypermedia-based model for integrating open services, data and metadata. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*. ACM Press, 128–137.
- [28] Theodor Holm Nelson. 1999. The Unfinished Revolution and Xanadu. *Comput. Surveys* 31, 4es (Dec. 1999).
- [29] Peter J. Nürnberg. 2000. Repositioning Structural Computing. In *Open Hypermedia Systems and Structural Computing. 6th International Workshop, OHS-6. 2nd International Workshop, SC-2. San Antonio, TX, June 3, 2000 (Lecture notes in computer science)*, Siegfried Reich and Kenneth M. Anderson (Eds.), Vol. 1903. Springer, 179–183.
- [30] Peter J. Nürnberg and Helen Ashman. 1999. What was the question? Reconciling open hypermedia and World Wide Web research. In *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*. ACM Press, 83–90.
- [31] Peter J. Nürnberg, Kim C. Kristoffersen, David L. Hicks, and Uffe K. Wiil. 2005. EAD Revisited: First Experiences. In *Proceedings of the 2005 Symposia on Metainformatics (MIS '05)*. ACM.
- [32] Peter J. Nürnberg, John J. Leggett, and Erich R. Schneider. 1997. As we should have thought. In *Proceedings of the 8th ACM Conference on Hypertext*. ACM Press, 96–101.
- [33] Peter J. Nürnberg, Uffe K. Wiil, and David L. Hicks. 2004. A Grand Unified Theory for Structural Computing. In *Proceedings of the International Metainformatics Symposium 2003 (Lecture Notes in Computer Science)*, David L. Hicks (Ed.), Vol. 3002. Springer, 1–16.
- [34] Peter J. Nürnberg, Uffe K. Wiil, and David L. Hicks. 2004. Rethinking structural computing infrastructures. In *Proceedings of the 15th ACM Conference on Hypertext and Hypermedia*. ACM Press, 239–246.
- [35] Thomas Schedel and Claus Atzenbeck. 2016. Spatio-Temporal Parsing in Spatial Hypermedia. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. ACM, 149–157.
- [36] Frank M. Shipman, Robert Airhart, Haowei Hsieh, Preetam Maloor, J. Michael Moore, and Divya Shah. 2001. Visual and spatial communication and task organization using the Visual Knowledge Builder. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*. ACM Press, 260–269.
- [37] Randall H. Trigg. 1983. *A Network-Based Approach to Text Handling for the Online Scientific Community*. Ph.D. Dissertation. University of Maryland.
- [38] Manolis Tzagarakis, Dimitris Avramidis, Maria Kyriakopoulou, Monica M. C. Schraefel, Michalis Vaitis, and Dimitris Christodoulakis. 2003. Structuring primitives in the Callimachus component-based open hypermedia system. *Journal of Network and Computer Applications* 26, 1 (2003), 139–162.
- [39] Manolis Tzagarakis, Michail Vaitis, and Nikos Karousos. 2006. Designing domain-specific behaviours in structural computing. *New Review of Hypermedia and Multimedia* 2 (12 2006), 113–142.
- [40] Manolis Tzagarakis, Michalis Vaitis, Athanasios Papadopoulos, and Dimitris Christodoulakis. 1999. The Callimachus approach to distributed hypermedia. In *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*. ACM Press, 47–48.
- [41] W3C. 2017. *HTML 5.3*. Public Working Draft 1. W3C.
- [42] Uffe K. Wiil, David L. Hicks, and Peter J. Nürnberg. 2001. Multiple open services: a New Approach to Service Provision in Open Hypermedia Systems. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*. ACM Press, 83–92.
- [43] Uffe K. Wiil, Samir Tata, and David L. Hicks. 2003. Cooperation services in the Construct structural computing environment. *Journal of Network and Computer Applications* 26, 1 (2003), 115–137.
- [44] Nicole Yankelovich, Norman Meyrowitz, and Andries van Dam. 1985. Reading and Writing the Electronic Book. *Computer* 18, 10 (1985), 15–30.