# As We Should Have Thought

*Peter J. Nürnberg, John J. Leggett, Erich R. Schneider*
Hypermedia Research Laboratory
Texas A&M University
College Station, TX 77843-3112 USA
Tel: 1-409-862-3217
E-mail:{pnuern, leggett, erich}@bush.cs.tamu.edu

**ABSTRACT**
The hypermedia field has long realized the need for first-class structural abstractions. However, we have failed to generalize the concept of ubiquitous structure management to problem domains other than navigation of information spaces. In this paper, we argue for the recognition of such a generalization, called structural computing, in which we assert the primacy of structure over data. We provide examples of four problem domains that are more naturally modeled with structure than data. We argue that support for structural computing must come in the form of new models, operating systems, and programming languages. We also assert that the experience gained by hypermedia researchers over the last decade may be naturally extended to form the basis of the new field of structural computing, and furthermore, the broadening of the applicability of our work is necessary for the continued vitality of our research community.

**KEYWORDS:** models of computation, hypermedia operating systems, hypermedia models, spatial hypertext, taxonomic hypertext, open hypermedia systems, hyperbases, structural computing

## 1 HYPERMEDIA IS DEAD

Linking is more than harmful [1, 10] — it is downright deadly. Two main problems exist with hypermedia research today and both can be tied directly to our current notion of linking. Firstly, linking implies a certain kind of structural paradigm, one in which the user (or occasionally a program) links information together for purposes of navigation. However, there are many other cases of computing in which structure plays a key role. We have even accepted some of these into our field. Yet, as a field, we have been unable to cast off the chains of the linking paradigm. This has often left us at a loss for common terminology to discuss these other uses of structure in computing. Witness the inability of many (even recently) proposed hypermedia models to naturally address the needs of spatial hypertext systems [6, 8] or the newer systems for literary [2, 12] and artistic work [4, 14].


Hypertext 97, Southampton UK


Secondly, linking implies the primacy of data, not structure. We still view hypertext functionality as something to be added onto/over "real" programs/data. The hypermedia field has long realized the necessity of *first-class* structural abstractions in our models, but we have yet to declare the *primacy* of structure over data in computing.

Now, ten years after the start of the hypertext conference series, we are faced with a question: should we evolve our field into a broader domain and allow our current notion of hypermedia to go the way of the dinosaur or try to sustain the species as "the" model of computing? In the rest of this paper, we attempt to make the case for expanding our horizons into the broader domain of structural computing.

## 2 STRUCTURAL COMPUTING

### 2.1 What We Should Have Realized

We should have realized that hypermedia is just a special case of a general philosophy of computing in which structure is more important than data. Structure should be the ubiquitous, atomic building block available to all systems at all times and from which all other abstractions (including data) are derived. Here, we call this philosophy of the primacy of structure "structural computing." What are the implications of completely adopting a structural computing paradigm?

Firstly, *models* must support the primacy of structure. It must not be the case that models build up idiosyncratic structural abstractions from data objects. Rather, models must support a set of generic structural abstractions that can be refined to suit the problem at hand.

Secondly, *(operating) systems* must support the primacy of structure. It must not be the case that structure is seen as an optional, additional abstraction for system construction. It must be *guaranteed* to be present and recognized as a resource to be managed.

Thirdly, *programming languages* must support the primacy of structure. It must not be the case that structural programming abstractions are reworked from scratch by programmers or only available as ad hoc extensions provided by various libraries. Core language definitions must provide generic structural capabilities.

In traditional systems, the computing environment provides basic *data* abstractions. Many models provide the data abstraction of "object," operating systems provide abstractions such as "file," and programming languages provide atomic data types such as "integer." How would these basic abstractions be changed in a structural computing environment? Models would provide *relationship* as the atomic building block. Operating systems would provide a generic structure store, in which abstractions such as file are modeled as structure with content. Programming languages could dispense with the notion of pointer, since all structure would automatically be separated from data and associated with content.

Without adopting structural computing completely and effecting the necessary changes to models, systems, and languages, we will relegate structurally-oriented systems (such as hypermedia systems) to ad hoc second-class oddities in a world of exclusively data-oriented systems.

### 2.2 Examples of Structural Computing Problem Domains

In this section, we present four problem domains for which structural computing is shown to be useful. Two of these problem domains (argumentation support and spatial hypertext) come from the hypertext literature, while the other two (botanical taxonomy and linguistics) are not normally associated with hypertext. A characteristic of the first two domains is that they are accepted as part of the hypertext field, despite the fact that they are difficult to discuss if we constrain ourselves to the traditional terminology and abstractions of hypertext. What is it about these two domains that has caused us to accept them as examples of hypertext systems? We believe it is their emphasis on structure, whether static and constrained [17] or dynamic and virtual [7]. The second two problem domains are modeled in a natural and convenient way within the assumptions of ubiquitous support for first-class structure.

*Argumentation Support.* Argumentation systems [9, 15, 17] were an early example of a different kind of structural computing than the more traditional navigationally-oriented systems. Structure in argumentation systems is a special case of structure in general structural computing systems, because there are semantic constraints on how certain objects may be linked together. These links are not the "associative trails" of Bush, but the formal structures of Toulmin [15]. The structure built in argumentation systems is not as personal as in some other forms of hypertext; rather, information is linked in ways that reflect formal, communally defined relationships. Is the notion of link sufficient to discuss both personal associations and formal relationships, or does it carry with it too many preconceived notions? Clearly there are similarities between these types of structures. Without explicitly recognizing the fact that personal associations and formal relationships are both special cases of more general structure, we are forced to use ambiguously defined and often inappropriate terminology to discuss different systems and models. By recognizing navigational and argumentation support systems as peer specializations of structural computing we will be able to identify more precisely the terms and concepts common and specific to both.

*Spatial Hypertext.* Spatial hypertext has always pushed the limits of our notions of hypertext. Where is the structure? What exactly are the information nodes? What does it mean to "follow a link?" Structure in spatial hypertext systems is dynamic and implicit. It is defined by the placement of data objects in a space. This structure is not traversed explicitly for the purpose of navigating the information. Instead, it is traversed (by the system) for the purpose of finding higher-level compositions of atomic data objects and lower-level compositions. For example, in VIKI [8], this traversal consists in large part of a spatial parse that recognizes patterns of objects. The notions of link and navigation are insufficient to discuss spatial hypertext — more general notions of structure and traversal are needed. This is reflected in the fact that spatial hypertext systems are built from scratch, without using existing limited systems or models as building blocks. Here, recognition of spatial hypertext as a specialization of structural computing distinct from, but on par with, navigation systems allows us to avoid overloaded and inappropriate terminology such as node and link while recognizing common concepts such as general structure and behavior management.

*Botanical Taxonomy.* Taxonomic systems have (in general) not been thought of as hypertext systems. However, as discussed in [11], much of the taxonomic problem can be well-modeled and supported by the structure management facilities found in general hypertext systems. Taxonomic work starts with a set of samples, which are then grouped into taxa. Taxa may further be grouped into supertaxa. The resulting taxonomy is generally a tree or forest of taxon trees with samples at the leaves. Problems arise in taxonomic work when taxa or samples are differently parented and/or named by different researchers. Often no objective criteria for grouping taxa or samples exist, and since both the available sample set and the subjective grouping criteria change over time, there are no unique and "correct" taxonomies. Instead, researchers must be able to move between taxonomies, at

times using parts of several taxonomies to model their work accurately.

Computational support for taxonomic work demands recognition of structure. In taxonomic systems, it is imperative — in fact, it is the definition of the work — to name the structural elements produced. Different types of computation exist over taxonomic structure than those found in navigational systems. We have previously argued [11] that such taxonomic systems, especially when built on a hyperbase system, represent a kind of hypertext. This, however, automatically constrains the discussions about such taxonomic systems to the use of already defined terms invented to describe other systems with a fundamentally different focus. We now advocate the description of both taxonomic systems and navigational systems as special cases of structural computing systems. This allows us to compare and contrast the systems more effectively, by allowing the use of common terminology when appropriate and avoiding it otherwise.

*Linguistics.* The application of computing to linguistic work defines a broad field. Here, we concentrate on diachronic comparative linguistics, the results of which are trees or graphs of languages, in some ways similar to taxonomic structures, but even more complex. In this linguistic problem, the data with which one begins are samples of extant languages. These samples are grouped into dialects, languages, language families, etc., as in the taxonomic problem. However, different samples may also be combined in a process known as "reconstruction" to produce language definitions for extinct languages. Reconstructed past languages may also be grouped as present languages, forming new taxonomies. Finally, languages are implicitly grouped in temporal ancestor and descendent relationships through the process of reconstruction. Taxonomies of languages at any point in time, present or past, have all the same problematic characteristics of regular taxonomies, with the additional overhead of maintaining diachronic (temporal) relationships. Furthermore, unlike most taxonomies, it is common for languages in linguistic taxonomies to have multiple parents.

As above, systems designed to support this problem domain require structure. As with regular taxonomies, the entire product of this work is the structure built from an ever-changing set of taxa and samples. Many results produced in the hypertext field apply directly to this domain. Again, however, it is unnecessarily limiting to call systems designed to support diachronic comparative linguistics "hypertext" systems. We advocate the description of these systems as another specific example of structural computing.

### 2.3 Conceptual Architectures for Structural Computing

What kind of conceptual architectures could support structural computing? Since structural computing is a generalization of hypermedia, in some sense, architectures supporting the former should be a generalization of the latter. Below, we discuss in broad terms the evolution of hypermedia system architectures with respect to generalizations and abstractions. The steps below are not intended to provide a rigorous accounting of the actual historical development of hypermedia systems, but rather describe in general terms the increasing generalization over time of representative hypermedia systems.

*Monolithic Systems.* Early hypermedia systems were monolithic (figure 1a). They included the interface to the user, a linking mechanism, and an interface to a basic file store in one program. In general, there was no distribution of any part of the system, except incidental distribution effected through distribution on the part of the interface (e.g., X) or underlying store (e.g., NFS). If behaviors (computations over structure) were distinguished at all, they were embedded in the system. The structure and data representations were combined into unified, application-dependent representations in the file store.

*Abstraction of Applications.* One of the first elements of the monolithic hypermedia system to be abstracted away was the user interface (figure 1b). This was manifested by the movement toward "open" hypermedia systems. A hypermedia system is open if it allows an open set of applications to participate in a common linking protocol provided by a link engine. As with the monolithic systems, however, all other aspects of the system (link engine, behaviors, interface to file stores) were centralized and embedded in one program. Openness at the application layer of the hypermedia system allowed natural explicit distribution at this layer as well. The degree to which applications should or should not be modified to participate in a common linking protocol is still an open issue [21], but the principle of this abstraction is well-established.

*Abstraction of Stores.* The next element of hypermedia systems to be targeted for abstraction was the interface to the stores (figure 1c). Systems began including an intermediate (and separate) layer which was interposed between the link engine (with embedded behaviors) and the file store, often in the form of a database. The inclusion of a database as the storage engine for hypermedia systems allowed transaction management, notification and access control, and other database provided functionality in link engine operations. The resultant storage layer was occasionally made open and distributed [3]. Oftentimes, after the storage layer had been abstracted away from the link engine, the link engine was renamed the 'link service,' since it now was a program with a single, well-defined purpose.

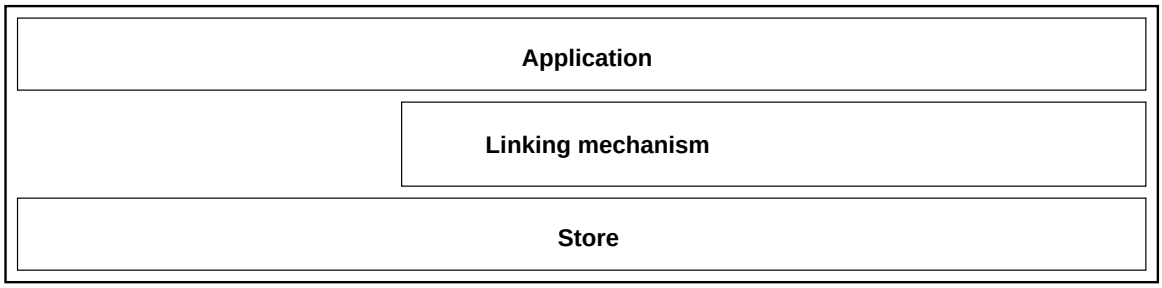


Figure 1a. Monolithic Systems.

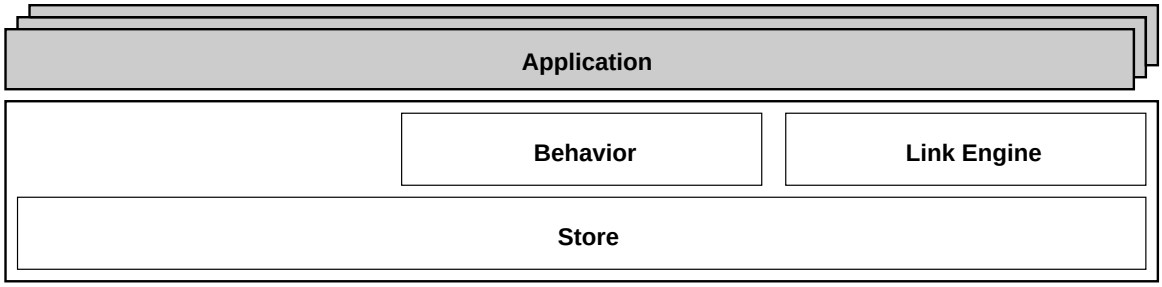


Figure 1b. Abstraction of Applications.

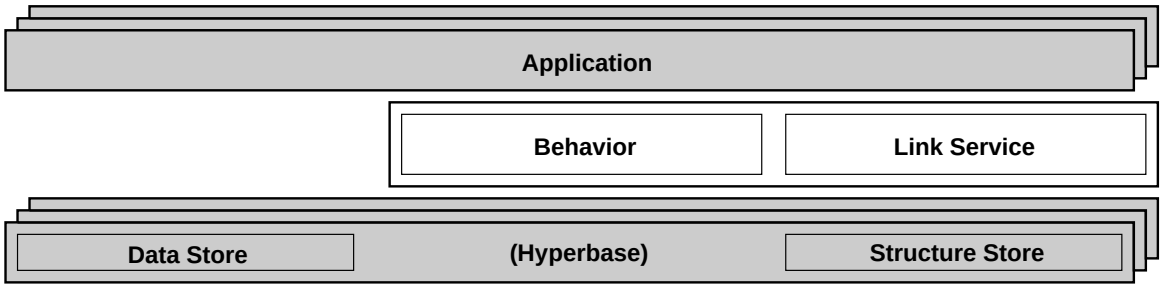


Figure 1c. Abstraction of Stores.

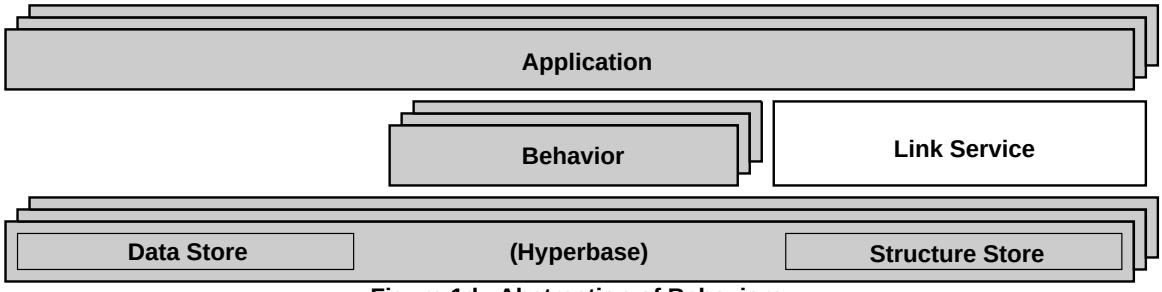


Figure 1d. Abstraction of Behaviors.

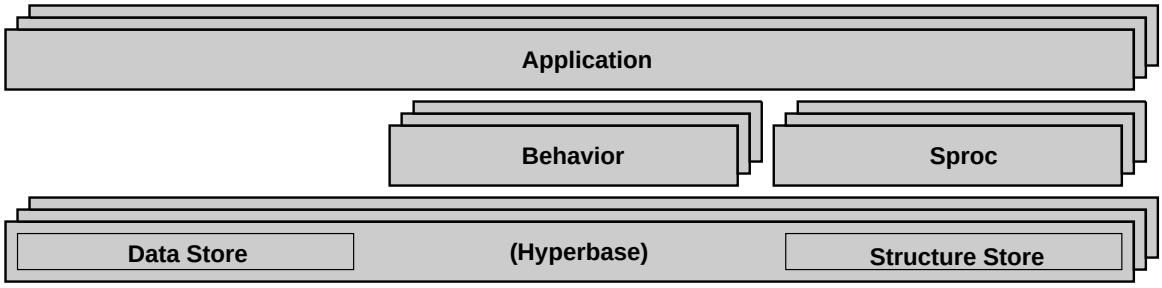


Figure 1e. Abstraction of Structure Processors.

The interest in hyperbases represented the abstraction of the structure store from the data store in the storage layers of hypermedia systems [13, 16, 19]. Hyperbases provided a separate interface to the storage layer that exclusively managed structure. This abstraction usually did not result in an open or distributed structure store layer in systems.

Instead, it represented the promotion of structure to a 'first-class' entity in the storage layer (which itself may or may not have been open and/or distributed).

*Abstraction of Behaviors.* Currently, there is interest in the abstraction and generalization of behavior (computations over structure). While some earlier systems allowed for the assignment of one of a closed set of behaviors to specific links, only recently have systems been reported that allow an open set of behaviors over structure [5, 18, 20]. Oftentimes, these behaviors are not distributed, and [11] discussed reasons why behaviors may be more efficiently implemented as threads of control inside a link service. In any event, newer systems (in general) should now allow an open set of behaviors (figure 1d).

*Abstraction of Link Services.* One principle underlies all the abstractions discussed above: abstraction of "non-hypermedia essential" properties from the link service. All of the above steps can be generalized to abstracting some functionality from the previous generation of hypermedia systems, and consequently allowing this functionality to become open and/or distributed. Indeed, in terms of the general conceptual architecture presented here, all non-link service elements have been abstracted and opened/distributed. More accurately, all layers have been opened except for the link service itself.

What does it mean to open this layer of hypermedia systems? What are generalized link services? We propose the term "structure processor" for generalized link service (figure 1e) and assert that open structure processors implement in a natural way structural computing.

The term structure processor (or Sproc) is appropriate for this new abstraction, because all Sprocs have in common the fact that they process structural abstractions. Sprocs may be thought of as clients of generic structure served by the structure store, and servers of specialized structure to applications or other processes that desire it.

The examples given in Section 2.2 all have in common an elevation of structure to first-class status in the conceptions of their respective problems. The conceptions of structure all have some basic elements in common. These are the elements that are provided by structure stores in hyperbases. However, they all specialize these general structural abstractions in what can be considered structure processors. To be sure, different problem domains may require different behaviors, but in systems with open behaviors, this is not problematic. Programmers building systems to support such problem domains should be able to consider the structure facilities of hyperbases as given. This maximizes the reuse of generic structural abstractions. However, general structural computing systems must allow for an open set of processes that refine these generic abstractions in order to admit all possible structural computing systems.

## 3 FUTURE RETHINK/REWORK

If linking is harmful and hypermedia is dead, where does this leave the work currently being done in hypermedia? Clearly, there is interesting and important work being done on navigationally-oriented hypermedia systems. However, just as clearly, we can identify interesting and important work related to hypermedia by virtue of the fact that it, too, is a special case of structural computing. We as hypermedia researchers have much to bring to other structural computing problem domains because of our rich history in dealing with a kind of first-class structure. For real progress to occur in these other domains, though, we must end the privileging of hypermedia over these other domains. As long as we are forced to consider other structural computing domains as derivations of hypermedia, we will be using inappropriate and overloaded concepts and terminologies in our modeling and discussions.

The call for the recognition of structural computing as an encompassing field for hypermedia and its relatives does not imply the need to discard hypermedia systems, concepts, or terminologies. It does, however, require two things of our field. The first is a rigorous definition of structural computing. What are the atomic elements and operations over general structure? How are these best implemented? The second is a set of mappings from this rigorous definition to specific domains. The first such mapping should probably be to the most well-established structural computing domain, navigational hypertext. However, we assert that there are a vast array of fields whose problems may be conveniently and efficiently addressed in a structural computing paradigm.

Viewed in light of these mappings, it would be helpful to *rethink* old ideas in extant structural computing fields (navigational hypertext, spatial hypertext, etc.) to define clearly that which is specific to these fields and that which applies generally to structural computing. We also should consider how we can provide support for general structural computing systems. The hypermedia field has a long history of designing and implementing infrastructure for hypermedia systems. We should now *rework* the results of our experience and apply it to the building of both general systems infrastructure and domain-specific structural computing systems. This reworking applies to models, operating environments, and programming languages.

In conclusion, hypermedia *as a field unto itself* may or may not be dead. However, it can *evolve* into the much broader, richer, and more important field of structural computing, of which hypermedia will always be the most venerable example. The question before us as hypermedia researchers is whether or not to claim this broader field as our own.

**ACKNOWLEDGEMENTS**

This research was supported in part by the Texas Advanced Research Program under Grant No. 999903-230.

**REFERENCES**

1. DeYoung, L. 1990. Linking considered harmful. *Proceedings of ECHT 90*, (Versailles, France, Nov) 238-249.

2. Greco, D. 1996. Hypertext with consequences: recovering a politics of hypertext. *Proceedings of HT 96* (Washington, DC, Mar) 85-92.

3. Kacmar, C., and Leggett, J. 1991. PROXHY: A process-oriented extensible hypertext architecture. *ACM Transactions on Information Systems 9*(4) Oct, 339-419.

4. Kendall, R. 1996. Hypertextual dynamics in A L*ife Set for Two*. *Proceedings of HT 96* (Washington, DC, Mar) 74-84.

5. Leggett, J. J., and Schnase, J. L. 1994. Viewing Dexter with open eyes. *Communications of the ACM 37*(2) Feb, 76-86.

6. Marshall, C., Halasz, F., Rogers, R., Jansen, W. 1991. Aquanet: a hypertext tool to hold your knowledge in place. *Proceedings of HT 91* (San Antonio, TX, Dec) 261-275.

7. Marshall, C., and Shipman, F. 1993. Searching for the missing link: discovering implicit structure in spatial hypertext. *Proceedings of HT 93* (Seattle, WA, Nov) 217-230.

8. Marshall, C., and Shipman, F. 1995. Spatial hypertext: designing for change. *CACM 38*(3), Aug, 88-97.

9. McCall, R., Bennett, P., D'Oronzio, P., Ostwald, J., Shipman, F., and Wallace, N. 1990. PHIDIAS: integrating CAD graphics into dynamic hypertext. *Proceedings of ECHT 90*, (Versailles, France, Nov) 152-165.

10. Meyrowitz, N. 1991. Hypertext—does it reduce choleterol, too? *From Memex to Hypertext* (Kahn et al., ed.) Academic Press, Boston.

11. Nürnberg, P., Leggett, J., Schneider, E., and Schnase, J. 1996. Hypermedia operating systems: a new paradigm for computing. *Proceedings of HT 96* (Washington, DC, Mar) 194-202.

12. Rosenberg, J. 1996. Content-oriented integration in hypermedia systems. *Proceedings of HT 96*, (Washington, DC, Mar) 11-21.

13. Schnase, J., Leggett, J., Hicks, D., Nürnberg, P., and Sànchez, J. 1993. Design and implementation of the HB1 hyperbase management system. *EP-ODD 6*(1), Mar, 35-63.

14. Sawhney, N. Balcom, D., and Smith, I. 1996. HyperCafe: narrative and aesthetic properties of hypervideo. *Proceedings of HT 96* (Washington, DC, Mar) 1-10.

15. Schuler, W., and Smith, J. 1990. Author's Argumentation Assistant (AAA): a hypertext-based authoring tool for argumentative texts. *Proceedings of ECHT 90*, (Versailles, France, Nov) 137-151.

16. Schütt, H. and Streitz, N. 1990. HyperBase: a hypermedia engine based on a relational database management system. *Proceedings of ECHT 90*, (Versailles, France, Nov) 95-108.

17. Smolensky, P. Bell, B., Fox, B., King, R., and Lewis, C. 1987. Constraint-based hypertext for argumentation. *Proceedings of HT 87* (Chapel Hill, NC, Nov) 215-245.

18. Wiil, U., and Leggett, J. 1992. Hyperform: Using extensibility to develop dynamic, open and distributed hypertext systems. *Proceedings of ECHT 92* (Milan, Italy, Nov), 251-261.

19. Wiil, U. 1993. Experiences with HyperBase: a hypertext database supporting collaborative work. *ACM SIGMOD Record 22*(4), Dec, 19.

20. Wiil, U. 1995. Hyperform: rapid prototyping of hypermedia services. *CACM 38*(8) Aug 1995, 109-111.

21. Wiil, U. and Demeyer, S. 1996. Proceedings of the 2nd workshop on open hypermedia systems. University of California at Irvine technical report ICS-96-10.