Abstracted Navigational Actions for Improved Hypermedia Navigation and Maintenance

Wilfried Lemahieu

Department of Applied Economic Sciences
Katholieke Universiteit Leuven
Naamsestraat 69 B-3000 Leuven
Belgium
tel: + 32 16 32 68 86
fax: + 32 16 32 67 32
e-mail: wilfried.lemahieu@econ.kuleuven.ac.be

Abstract

This paper discusses the MESH framework, which proposes a fully object-oriented approach to hypermedia. Object-oriented abstractions are not only applied to the *conceptual data model*, but also to the *navigation paradigm*. This results in the concept of *context-based navigation*, which reduces the end user's disorientation problem by means of dynamically generated, context-sensitive guided tours. Moreover, maintainability is greatly improved, as both nodes and links are defined as instances of abstract classes. In this way, single links and entire guided tours are anchored on type level as *abstract navigational actions*, which are independent of the actual link instances.

1 Introduction

The hypermedia paradigm looks upon data as a network of *nodes*, interconnected by *links*. Whereas each node symbolizes a *concept*, a link not only stands for a *relation* between two items, but also explicitly assumes the semantics of a navigation path, hence the quintessential property of *navigational data access*. Their inherent flexibility and freedom of navigation raises hypermedia systems as utterly suitable to support user-driven exploration and learning. Therefore, hypermedia data retrieval embraces a notion of *location*. Data accessibility depends on a user's position in the network, denoted as the *current node*. Manipulation of this position gradually reveals links to related information [14]. Unfortunately, due to inadequacy of the underlying abstractions, most hypermedia technologies suffer from severely limited maintainability. Moreover, the explorative, non-linear nature of hypermedia navigation imposes a heavy processing load upon the end user, referred to as *cognitive overhead*. The stringent problem of cognitive overhead effecting into user disorientation and losing one's chain of thought is known as the 'lost in hyperspace' phenomenon [7].

The benefits of data modeling abstractions to both orientation and maintainability were already acknowledged in [6]. They yield richer domain knowledge specifications and more expressive querying. Typed nodes and links offer increased consistency in both node layout and link structure [9]. Higher-order information units and perceivable equivalencies (both on a conceptual and a layout level) greatly improve orientation [17]. Semantic constraints and consistency can be enforced [1]; [5], tool-based development is facilitated and reuse is encouraged [15]. The first conceptual hypermedia modeling approaches such as *HDM* [4] and *RMM* [8] were based on the entity-relationship paradigm. Object-oriented techniques were mainly applied in *hypermedia engines*, to model functional behavior of an application's *components*, e.g. *Hyperstorm* [2], *Microcosm* [3] and *Hyperform* [19]. Along with *EORM* [10] and *OOHDM* [16], *MESH* is the first approach where modeling of the *application domain* is fully accomplished through the object-oriented paradigm.

The *MESH* hypermedia framework as deployed in [11] proposes a structured approach to both data modeling and navigation, so as to overcome said maintainability and user disorientation problems. MESH is an acronym for *Maintainable*, *End user friendly*, *Structured Hypermedia*. MESH's data model is based on concepts and experiences in the related field of database modeling, taking into account the particularities inherent to the hypermedia approach to data storage and retrieval. Established object-oriented modeling abstractions are coupled to proprietary concepts to provide for a *formal hypermedia data model*. While uniform layout and link typing specifications are attributed and inherited in a *static* node typing hierarchy, both nodes and links can be submitted *dynamically* to multiple complementary classifications. The data model provides for a firm hyperbase structure and an abundance of meta-information that facilitates implementation of the *context-based navigation paradigm*. Here, conventional navigation along static links is complemented by run-time generated *guided tours*, which are derived dynamically from the context of a user's information requirements. This paper briefly overviews the data model and navigation paradigm[1]. Subsequently, the concept of *abstract navigational actions* is explained in detail. As a conclusion, the approach is evaluated with respect to both orientation and maintenance.

2 The MESH hypermedia framework

2.1 The basic concepts: node types, layout templates and link types

On a conceptual level, a *node* is considered a black box, which communicates with the outside world by means of its *links*. External references are always made to the node *as a whole*. True to the object-oriented *information-hiding* concept, no direct calls can be made to its multimedia content. However, internally, a node may encode the intelligence to adapt its visualization to the *navigation context*, as discussed in section 4.1. Nodes are assorted in an inheritance hierarchy of *node types*. Each child node type should be compliant with its parent's definition, but may fine-tune inherited features and add new ones. These features comprise both node layout and node interrelations, abstracted in *layout templates* and *link types* respectively.

A *layout template* is associated with each level in the node typing hierarchy, every template being a refinement of its predecessor. Its exact specifications depend on the implementation environment, e.g. as to the Web it may be *HTML* or *XML* based. Node typing as a basis for layout design allows for uniform behavior, onscreen appearance and link anchors for nodes representing similar real world objects.

A *link* represents a one-to-one association between two nodes, with both a semantic and a navigational connotation. A directed link offers an access path from its *source* to its *destination node*. Links representing similar semantic relationships are assembled into *types*. Link types are attributed to node types and can be inherited and refined throughout the hierarchy.

Link type properties such as *domain*, *cardinalities* and *destination/inverse* allow for enforcing constraints on their instances. The *domain* is the node type (or aspect type, see the next section) to which the link type is attributed. The *cardinalities* determine the minimum and maximum number of link instances allowed for a given source node. The *inverse link type* is the *most specific* link type that encompasses all of the original link type's tuples, with reversed source node and destination node. As explained in detail in [11], a link type's *destination* is a derived property, defined as the *inverse link type's domain*. These properties can be overridden to provide for stronger restrictions upon inheritance. E.g. whereas an **artist** node can be linked to any **artwork** through a ***has-made*** link type, an instance of the child node type **painter** can only be linked to a **painting**, by means of the more specific child link type ***has-painted***.

2.2 Aspect descriptors and aspect types

The above model is based on a node typing strategy where node classification is total, disjoint and constant. The aspect construct allows for defining *additional* classification criteria, which are not necessarily subject to these restrictions. Apart from a single "most specific node type", they allow a node to take part in other secondary classifications that are allowed to change over time[2].

An *aspect descriptor* is defined as an attribute whose (discrete) values classify nodes of a given type into respective additional subclasses. In contrast to a node's "main" subtyping criterion, such aspect descriptor should not necessarily be *single-valued* or *constant over time*. Aspect descriptor properties

[1] For a more thorough discussion of the navigation paradigm and the data model, we refer to [12] and [13] respectively.
[2] We deliberately opted for a single inheritance structure. However, aspects can provide an elegant solution in many situations that would otherwise call for multiple inheritance. See [11] for further details.

denote whether the classification is *optional*/*mandatory*, *overlapping*/*disjoint* and *temporary*/ *permanent.* Each *aspect type* is associated with a single value of an aspect descriptor. An aspect type defines the properties that are attributed to the class of nodes that carry the corresponding aspect descriptor value. An aspect type's instances, *aspects*, implement these type-level specifications. Each aspect is inextricably associated with a single node, adding characteristics that describe a specific "aspect" of that node.

As opposed to node types, aspects are allowed to be volatile. Hence, dynamic classification can be accomplished by manipulating aspect descriptor values, thus adding or removing aspects at run-time. Aspect types attribute the same properties as nodes: *link types* and *layout*. However, their instances differ from nodes in that they are not directly referable. An aspect represents the *same real-world object* as its associated node and can only be visualized as a subordinate of the latter. E.g. to model an **artist** that can be skilled in multiple disciplines, a non-disjoint aspect descriptor *discipline* defines the **painter** and **sculptor** aspect types. Discipline-specific node properties are modeled in these aspect types, such that the **Michelangelo** node features the combined properties of its **Michelangelo.asPainter** and **Michelangelo.asSculptor** aspects.

2.3 Link typing and subtyping

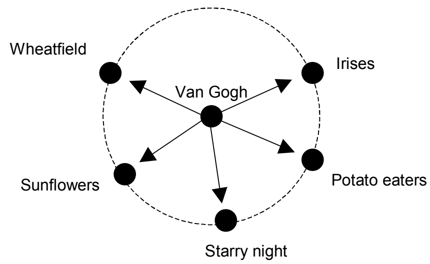
In common data modeling literature, subtyping is invariably applied to *objects*, never to *object interrelations*. If additional classification of a relationship type is called for, it is *instantiated* to become an object type, which can of course be the subject of specialization. However, as for a hypermedia environment, node types and link types are two separate components of the data model with very different purposes. It would not be useful to instantiate a link type into a node type, since such nodes would have *no content* to go along with them and thus each instance would become an 'empty' stop during navigation. Therefore, in MESH, specialization semantics can be enforced not only upon node types, but also upon the *link types.*

A link instance can be seen as a source node - destination node tuple (n_s, n_d). Tuples for which this association represents a similar semantic meaning are grouped into link types. A sub link type will model a type whose set of instances constitutes a subset of its parent's, and which models a relation that is more specific than the one modeled by the parent. A link type's domain, cardinalities and inverse can be overridden upon specialization. MESH presents a formal overriding mechanism, wherein particular care is taken so as not to violate the parent's constraints, particularly in case of a non-disjoint classification. For further details we refer to [11]. Link types are deemed extremely important, as they not only enforce semantic constraints but also *interface* between nodes, such that these can be coded and updated independently of one another. Moreover, they provide the basis for *context-sensitive node visualization*, as discussed in section 4.1.

2.4 Guided tours derived from the current context

In conventional hypermedia applications, the *current node* is the only variable that determines which information is accessible at a given moment; navigation is only possible to nodes that are linked to this current node. Its value changes with each navigation step as it represents the immediate focus of the user's attention. MESH introduces the *current context* as a second, longer-term variable that 'glues' the various visited nodes together and provides a background about which common theme is being explored. The current context is defined as the combination of a *context node* and a *context link type*. The context node represents the subject around which the user's broader information requirements 'circle'. The nature of the relationship involved is depicted by the context link type.

MESH builds upon its data model and the context notion to reconcile navigational freedom with the ease of linear navigation, by offering guided tours to a disoriented end user, chaining together all nodes pertaining to a common subject with *forward*/*backward* links. In contrast to the traditional guided tour notion [18], such guided tour is not static, but is adapted dynamically to the *navigation context*. In addition, a node is able to tune its *visualization* to the context in which it is accessed, hence providing the user with the most relevant subset of its embedded multimedia objects. E.g. the typical hypermedia links (represented as arrows) between **Van Gogh** and each of his **paintings** can be complemented by a *guided tour* (represented as dotted lines) along these **paintings** (see figure 1).



Current context: **Van Gogh.*has-painted***

Figure 1: a guided tour, as derived from the current context

A guided tour derives from the current context. Therefore, MESH discriminates between *direct* and *indirect* links. A direct link represents a lasting relation between two nodes. Direct links are typed and reflect the underlying conceptual data model. Because they are permanent and context-independent, they are stored explicitly into the hyperbase and are always valid. E.g. the node **Sunflowers** is directly linked to the **Van Gogh** node. An *indirect* link between two nodes indicates that they share relevancy to a common third node. The latter denotes the *context* within which the indirect link is valid. As indirect links not only reflect the data model, but also depend on a run-time variable, the *current context*, they cannot be stored within the hyperbase. They are to be created *dynamically* at run-time, as inferred from a particular context. E.g. an indirect link between **Sunflowers** and **Wheatfield** is only relevant when exploring information related to **Van Gogh**.

A *guided tour* is defined as a path of *indirect* links along all nodes relevant to the current context. These nodes are directly linked to the context node (through instances of the context link type) and indirectly to their predecessor and successor in the tour. As they are chained into a linear structure, a logical order should be devised in which the subsequent tour nodes can be presented to the user. The most obvious criterion is in alphabetical order of a *node descriptor* field. More powerful alternatives are discussed in [Lemahieu, 1999]. E.g. the context **Van Gogh**.***has-painted*** yields a guided tour among the nodes {**Irises**, **Potato eaters**, **Starry night**, **Sunflowers, Wheatfield,** …} with **Van Gogh** as the *context node* and ***has-painted*** as the *context link type*.

2.5 A general implementation architecture

in MESH, the *information content* and *navigation structure* of the nodes are separated and stored independently. The resulting system consists of three types of components: the *nodes*, the *linkbase/repository* and the *hyperbase engine*. In [11], a platform-independent implementation framework was provided, but all subsequent prototyping is explicitly targeted at a *Web* environment.

A node can be defined as a static page or a dynamic object, using e.g. *HTML* or *XML*. Its internal content is shielded from the outside world by the indirection of link types playing the role of a node's *interface*. Optionally, it can be endowed with the intelligence to tune its reaction to the *context* in which it is accessed, by integrating the node type's set of attributed link types as a parameter in its layout template's presentation routines (see also section 4.1).

Since a node is not specified as a necessarily searchable object, linkage information cannot be embedded in a node's body. Links, as well as meta data about node types, link types, aspect descriptors and aspects are captured within a searchable *linkbase/repository* to provide the necessary information pertaining to the underlying hypermedia model, both at design time and at run-time. This repository is implemented in a relational database environment. Only here, references to physical node addresses are stored, these are never to be embedded in a node's body. All external references are to be made through location independent *node ID*'s.

The *hyperbase engine* is conceived as a server-side application that accepts navigational commands from the current node, retrieves the correct destination node, keeps track of the current context and provides facilities for generating maps and overviews. Since all relevant linkage and meta information is stored in the relational DBMS, the hyperbase engine can access this information by means of simple, pre-defined and parameterized *database queries*, i.e. without the need for searching through *node content*.

3 Abstract navigational actions

3.1 Introduction

Navigational actions in MESH can be classified according to two dimensions. First, there is moving *forward* and *backward* within the current tour, along indirect links. Second, and orthogonal to this, there is the option of moving *up* or *down* along direct links, closer to or further away from the session's starting point. Additionally, one can distinguish between actions that change the current context and actions that only influence the current node. However, any of these movements can be specified as an *abstract navigational action*, i.e. independently of the actual link instance(s) involved. This feature will prove to be very advantageous to both navigation and maintenance.

3.2 Moving forward/backward within the current tour

Moving forward or backward in a guided tour along indirect links, results in the node following/preceding the current node being accessed to become the new current node. The current context remains unaffected (see figure 2). Such action can be specified (and anchored) as a simple "next" or "previous" command. The hyperbase engine calculates the correct destination node, based on the current node and the current context. Hence, the action can be specified unambiguously without referring to the actual link instance.

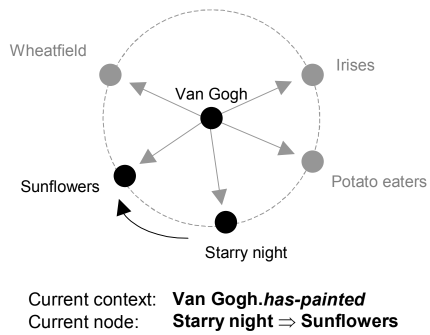


Figure 2: moving forward within the current tour

3.3 Moving up/down

Moving down implies an action of 'digging deeper' into the subject matter, moving away from the starting point. This is accomplished through selection of a direct link type from the current node. In the case of a *unique* destination node, the result is the latter node being accessed. In the case of a *set* of destination nodes, the outcome is a new "nested" tour being started.

In traditional hypermedia navigation, selection of a link instance l from a given source node n_s results in its unique destination node n_d being accessed: $n_s.l := \{n_d \mid l = (n_s, n_d)\}$. E.g. selection of the link (**Sunflowers, National Gallery**) from the current node **Sunflowers**, induces an access to the node **National Gallery**. This can be symbolized as **Sunflowers**.(Sunflowers, National Gallery) := {**National Gallery**}.

However, MESH aggregating single *link instances* into *link types*, yields the opportunity of anchoring and consequently selecting a *complete link type* from a given source node. Selection of a link type L from a source node n_s yields a set of all destination nodes n_d of tuples representing link instances of L with n_s as the source node, i.e. all nodes that are linked to the current node by the selected link type: $n_s.L := \{n_d \mid (n_s, n_d) \in L\}$. Depending on maximum cardinality of the link type, the resulting set may be a singleton or may contain multiple destination nodes.

E.g. selection of the *unique* link type ***exhibited-in*** from the current node **Sunflowers**, induces an access to the node **National Gallery**. The latter can be symbolized as **Sunflowers.*exhibited-in*** := {**National Gallery**}. The result is the same as with traditional hypermedia, only now, the action is specified by the *link type* instead of the actual link instance.

Selection of the *non-unique* link type ***reviews*** from the same current node **Sunflowers** generates a *collection* of nodes to-be-accessed: **Sunflowers**.***reviews*** := {**review#1**, **review#2**, **review#3**, …}. The result of such action is a *context change*: a new context emanates, resulting in new indirect links. A new guided tour is generated, nested within the former, according to this new context (figure 3). The new tour is completely specified by its context, i.e. the combination of the source node and a *link type*. Again, such action can be identified unambiguously without referring to the actual link instances.

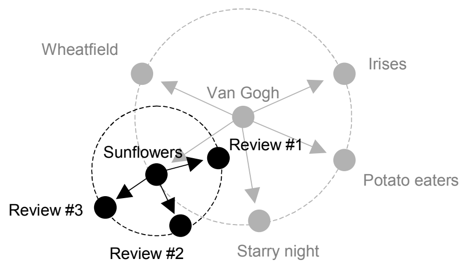


Figure 3: moving down, resulting in a context change

Hence contexts, and consequently guided tours, can exist in layers. As such, it is possible to 'delve' into a subject and have multiple *open* tours, nested within one another, where the context node of one tour is the current node of the tour it is nested in. Navigation along indirect links is invariably carried out within the "deepest", i.e. most recently started tour. Continuing a tour on a higher level is only possible if all tours on a lower level have been either completed or disbanded. The latter is accomplished by *moving up*, which reverses the latest *move down* action. If the latter involved a context change, the move up action results in the reestablishment of the previous context and the cancellation of the tour generated through this most recent link type selection. The previous context's context node and indirect links are restored. The most recent context node (**Sunflowers** in the example) again becomes the current node. Obviously, this action doesn't require the specification of a link instance either and can be identified by a simple command.

3.4 Abstract navigational actions applied to a complete tour

The way in which navigational actions are specified in MESH, allows for casting abstract navigational actions to a whole *class* of nodes, regardless of the actual instance they are applied to. In this way, selections of link *types* that exist at a sufficiently high level of abstraction can be imposed upon every single node belonging to a tour. E.g. in the context of **Van Gogh.*has-painted***, a ***reviews*** link type selection can be issued once on *tour* level, with additional (nested) tours **Irises.*reviews***, **Potato_eaters.*reviews***, **Starry_night.*reviews*** etc. being generated automatically for each node participating in the **Van Gogh.*has-painted*** tour. If these tours in their turn include navigational actions on type level, a complex navigation pattern results, which can be several levels deep. Again, *forward* and *backward* links always apply to the current tour, i.e. to the open tour at the 'deepest' level. In addition, the abstract navigational actions and tour definitions sustain the generation of very compact tree-shaped overviews and maps of complete navigation sessions. In this respect, the *move up* and *move down* actions indeed correspond to moving up or down in the graph. The represented information can also be *bookmarked*, i.e. bookmarks not just refer to a single node but to a complete navigational situation, which can be resumed at a later time[3].

4 Conclusions

4.1 Improved maintainability

MESH's object-oriented data modeling abstractions allow for hypermedia maintenance capabilities equaling their database counterpart; with unique object identifiers, monitoring of integrity, consistency and completeness checking, efficient querying and a clean separation between authoring *content* and *physical hyperbase maintenance*. MESH formulates specific rules for inheriting and overriding layout and link type properties, taking into account the added complexity of plural (possibly overlapping and/or temporal) node classifications. Links are treated as first-class objects, with link types being able to be subject to multiple specializations themselves, not necessarily in parallel with node subtyping.

[3] See [11] for further details.

MESH greatly improves node independence and maintainability by anchoring *link types* instead of *link instances*. Abstract navigational actions, both within the current tour and orthogonal to the current tour, can be unambiguously specified by the combination of the source node and a link type. The latter defines the context within which the action takes place, or the new context induced by the action. An *anchor* is defined as the association between a user interface event (e.g. clicking an underlined word, a button, a hot spot in a map etc.) and such action. Upon stimulation, it causes the current node to *close* and pass the corresponding link type as a return value to the hyperbase engine. The latter calculates the correct destination node. A unique link type is mapped to a unique destination node. A non-unique link type is mapped to a *guided tour*, of which the first participating node is accessed. Hence, anchors remain independent of actual link instances and can be defined once at the level of node type (or aspect type) *layout templates*, instead of for each individual node or aspect instance.

Maintenance of the individual link instances does not affect the node's internal properties. Links can be added or removed without affecting the anchor and, consequently, the node's content. Guided tours do not require any maintenance nor design effort, as the author is not even engaged in their realization: they are calculated at runtime by means of linkbase queries by the hyperbase engine, according to the user's actions. This approach does not only facilitate development to a great extent, but also improves the user's grasp upon the underlying hypermedia structure by providing similar anchors to similar links. As such, cognitive overhead and the risk of disorientation are reduced. Moreover, by acknowledging a node type's set of attributed link types as a factor in its visualization, a node can provide an appropriate reaction to each context in which it may be accessed. Consequently, the user can be directed to the most relevant section(s) of the node's information content in this particular context. Sub link types, modeling a more specific relationship between two nodes than their parent, potentially provoke a more specific reaction by their instances' destination nodes. More details on this *context-sensitive visualization* principle can be found in [11].

A last advantage of MESH's emphasis on abstraction is the fact that the majority of the nodes' properties can be defined on an aggregate level. Authoring is greatly facilitated because layout as well as attributed link types and anchors can be laid down at node *type* (or aspect type) level. By means of inheritance and overriding, abstract specifications can be refined on more concrete levels of the typing hierarchy. In this respect, delegation to aspect types allows for specialization according to different criteria, such that a cohesive set of properties can be encapsulated into a single entity. Needless to say that specification of properties on an abstract level will also improve hyperbase consistency, which in its turn reduces cognitive overhead and, consequently, end user disorientation.

4.2 Facilitated orientation

Indeed, apart from the obvious benefit of a well-maintained hyperbase, the abstractions should permit a user to better *understand* the information presented in the hypermedia system. The use of higher-order information units, i.e. node and link types, allows for consistent layout and user interface features. Reflecting similarities between nodes and the representation of collections of nodes as (source node, link type) combinations, induces a stronger sense of structure and is to reduce cognitive overhead.

Obviously, navigation is facilitated by the context-sensitive navigation paradigm providing run-time generated guided tours and adapting node visualization to the context in which a node is accessed. Moreover, through the specification of navigational actions *on tour level*, complex navigation patterns can be applied to all nodes in a tour without additional effort. The abundance of meta-information as node, aspect and link types allows for enriching *maps* and *overviews* with concepts of varying granularity. A final benefit is the ability to *bookmark a complete navigational situation* in an utterly compact manner, with the possibility of it being resumed later on, from the exact point where it was left.

References

[1] Ashman H., Garrido A. and Oinas-Kukkonen H., Hand-made and Computed Links, Precomputed and Dynamic Links, *Proceedings of Hypertext - Information Retrieval - Multimedia (HIM '97)*, Dortmund (Sep. 1997)

[2] Bapat A., Wäsch J., Aberer K. and Haake J., An Extensible Object-Oriented Hypermedia Engine, *Proceedings of the seventh ACM Conference on Hypertext (Hypertext '96)*, Washington D.C. (Mar. 1996)

[3] Davis H., Hall W., Heath I., Hill G. and Wilkins R., MICROCOSM: An Open Hypermedia Environment for Information Integration, *Computer Science Technical Report CSTR 92-15* (1992)

[4] Garzotto F., Paolini P., and Schwabe D., HDM - A Model-Based Approach to Hypertext Application Design, *ACM Trans. Inf. Syst. Vol. 11*, No. 1 (Jan. 1993)

[5] Garzotto F., Mainetti L. and Paolini P., Hypermedia Design, Analysis, and Evaluation Issues, *Commun. ACM Vol. 38*, No. 8 (Aug. 1995)

[6] Halasz F., Reflections on NoteCards: Seven Issues for Next Generation Hypermedia Systems, *Commun. ACM Vol. 31*, No. 7 (Jul. 1988)

[7] Hammond N., Learning with Hypertext: Problems, principles and Prospects, *HYPERTEXT a psychological perspective*, C. McKnight, A. Dillon and J. Richardson Eds., *Ellis Horwood*, New York (1993)

[8] Isakowitz T., Kamis A. and Koufaris M., The Extended RMM Methodology for Web Publishing, *Working Paper IS-98-18, Center for Research on Information Systems*, 1998 (Currently under review at ACM Trans. Inf. Syst.)

[9] Knopik T. and Bapat A., The Role of Node and Link Types in Open Hypermedia Systems, *Proceedings of the sixth ACM European Conference on Hypermedia Technology (ECHT '94)*, Edinburgh (Sep. 1994)

[10] Lange D., An Object-Oriented design method for hypermedia information systems, *Proceedings of the twenty-seventh Hawaii International Conference on System Sciences (HICSS-27)*, Hawaii (Jan. 1994)

[11] Lemahieu W., Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling, *Doctoral dissertation (unpublished)*, Leuven (Jul. 1999)

[12] Lemahieu W., A Context-Based Navigation Paradigm for Accessing Web Data, *Proceedings of the ACM Symposium on Applied Computing (SAC 2000)*, Como, Italy (Mar. 2000)

[13] Lemahieu W., *MESH*: A Model-Based Approach to Hypermedia Design, in: *Chen, Q. (ed.) Human Computer Interaction: Issues and Challenges*, Idea Group Publishing, Hershey, PA (Jan. 2001)

[14] Lucarella D., A Model For Hypertext-Based Information Retrieval, *Proceedings of the European Conference on Hypertext*, Versailles (Nov. 1990)

[15] Nanard J. and Nanard M., Hypertext Design Environments and the Hypertext Design Process, *Commun. ACM Vol. 38*, No. 8 (Aug. 1995)

[16] Schwabe D. and Rossi G., Developing Hypermedia Applications using OOHDM, *Proceedings of the ninth ACM Conference on Hypertext (Hypertext '98)*, Pittsburgh (Jun. 1998)

[17] Thüring M., Hannemann J. and Haake J.: Hypermedia and Cognition: Designing for comprehension, *Commun. ACM Vol. 38*, No. 8 (Aug. 1995)

[18] Trigg R., Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment, *ACM Trans. Office Inf. Syst. Vol. 6*, No. 4 (Oct. 1988)

[19] Wiil U. and Leggett J., Hyperform: a hypermedia system development environment, *ACM Trans. Inf. Syst. Vol. 15*, No. 1 (Jan. 1997)