As We May Link: A General Metamodel for Hypermedia Systems

Beat Signer and Moira C. Norrie

Institute for Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland
{signer,norrie}@inf.ethz.ch

Abstract. Many hypermedia models have been proposed, including those specifically developed to model navigational aspects of web sites. But few hypermedia systems have been implemented based on metamodelling principles familiar to the database community. Often there is no clear separation between conceptual and technical issues in the models and their implementations are not based on an explicit representation of a metamodel. This results in a loss of generality and uniformity across systems. Based on principles of metamodel-driven system development, we have implemented a platform that can support various categories of hypermedia systems through the generality and extensibility of the metamodel. We present our metamodel and show how it generalises concepts present in a range of hypermedia and link server systems.

1 Introduction

The vision presented by Vannevar Bush in his paper *As We May Think* [1] is often accredited as being the origin of hypermedia systems. Since then, many hypermedia models and systems have been developed, but they are all based on the same underlying model of information spaces as interlinked collections of resources. Variations abound according to the precise nature of the links and resources, how they can be authored and accessed and also the application domains considered. This has led to numerous categories of systems including open hypermedia, adaptive hypermedia, physical hypermedia and spatial hypermedia. Of course, the most famous of all hypermedia systems is the World Wide Web and the hypermedia community has actively investigated ways of extending the underlying technologies and tools to enable more advanced and flexible features to be supported. At the same time, hypermedia models have been adopted by the web engineering community as a basis for modelling navigation and adaptation in model-based approaches to web site development.

However, a study of the hypermedia literature reveals a lack of clear, conceptual models that are general and flexible enough to support the development of a wide range of hypermedia systems and applications. In some cases, conceptual and technical issues are combined into the same model, while other approaches integrate application-specific concepts into the core of the model. Further, in contrast to database systems, implementations are rarely metamodel-driven. This

means that a metamodel is not represented explicitly in the system resulting in, not only a loss of flexibility, but also the introduction of major restrictions in the model during the implementation process.

Our goal was to produce a general platform for the development of hypermedia systems based on principles of metamodel-driven engineering and extensibility. This meant defining a core link metamodel that is general enough to support features of many different systems and free from implementation issues. In addition, the core metamodel was designed with extensibility in mind so any type of resource and link could be supported. For example, in the implemented system, we currently support text and XHTML documents, images, videos, Flash movies, databases, RFIDs, interactive paper documents and program components as resources that can be linked together. In addition, each type of resource can have one or more selectors defined to enable links to and from elements within resources. We support both navigational and structural links as well as links with multiple targets, multiple sources and also links over links.

In this paper, we present our metamodel and show how it generalises existing hypermedia models in terms of supporting concepts of these models either directly in the core model or through extensibility. At the same time, we use this as an example to show the benefits that can be attained by using a conceptual metamodel as the basis for system engineering.

We start in Sect. 2 by describing the range of existing hypermedia models and systems in order to identify the requirements of a general metamodel and also highlight some of the problems of existing model definitions. Section 3 then presents the core of our metamodel in terms of link concepts. In Sect. 4, we show how a user model is integrated into the core metamodel. The concept of layers is introduced in Sect. 5 and we describe how this can be used to support nested links. Section 6 shows how the core model was extended to support structural links as well as navigational links. In Sect. 7, we discuss some key features of the implementation. Concluding remarks are given in Sect. 8.

2 Background

Over the last two decades a variety of hypermedia models for different domains and purposes have been proposed. We first review some of the best known in order to show the variety of features supported and part of the history of how these models evolved. We then discuss some general limitations of the proposed models and implementations in order to motivate our approach for a general metamodel supporting different hypermedia domains.

In an attempt to generalise concepts from different hypertext systems, the *Dexter* hypertext reference model [2] introduced three abstraction layers. The storage layer describes a network of interlinked nodes (components) whereas the within-component layer deals with the content and structure within those nodes. User interaction with hypermedia content is handled by the runtime layer. A limitation of the Dexter model is the fact that all data has to be encapsulated within the components and data not forming part of the hypermedia structure itself

cannot be addressed. Furthermore, the Dexter model does not specify in detail how anchors can be used to address parts of composite components. The *DeVise Hypermedia* (DHM) system [3] for cooperative hypermedia addressed some of these limitations by extending the Dexter model. Around the same time, the *Amsterdam Hypermedia Model* (AHM) [4] added concepts of time and context to the Dexter model to investigate ways of combining multimedia and hypertext concepts to support the linking of dynamic multimedia information. In addition to the original navigational hypermedia models, spatial [5] and taxonomic hypermedia [6] models were also investigated in the mid-90s.

Adaptive hypermedia systems enable the content and link structure to be adapted dynamically based on the user context by integrating a user concept into the model [5]. *AHAM* [7] is a reference model for adaptive hypermedia systems that extends the storage layer of the Dexter model with a user model.

Open Hypermedia architectures address interoperability between hypermedia systems and the *Open Hypermedia Protocol* (OHP) was developed for the exchange of navigational link information. OHP was specified using DTDs resulting in a lack of detail due to the limited expressiveness of the chosen "specification language". The *Fundamental Open Hypertext Model* (FOHM) [8], an extension of OHP, attempts to provide a common data model for navigational, spatial and taxonomic hypermedia by providing operations for these three domains. However, a drawback of FOHM is its limitation to exactly those three domains, ignoring other existing hypermedia domains. The issue of limited extensibility in terms of structural abstractions necessary to support different hypermedia domains was addressed by *Component-based Open Hypermedia Systems* (CB-OHS) [9]. In many open hypermedia systems, the controlled sharing of information seems to be difficult since the majority of approaches do not consider user management and the issues of *data* and *link ownership* in their core model.

A distinguishing feature of open hypermedia systems is the fact that they use external link servers to deal with links between resources. Managing links separately from resources allows for greater flexibility in supporting features such as bidirectional links, multi-source and multi-target links and link groups. Importantly, it also enables the removal of the sharp distinction between the authors and users of links since users can create links between resources without having access rights to modify those resources. Well known link servers include *Chimera* [10], *Microcosm* [11] and *Hyper-G* along with its successor *Hyperwave* [12]. Similar issues of embedding links in resources as opposed to managing them separately arose in the context of the World Wide Web and the hypermedia community have contributed to the development of the *XML Linking Language* (XLink) [13] which allows links to be managed separately as well as providing more flexibility in terms of defining and accessing links. The XLink standard is based on the *Hypermedia/Time-based Structuring Language* (HyTime) [14]. As part of the Semantic Web initiative, the Annotea project [15] uses these ideas to allow users to create and share annotations of web resources.

More recently, physical hypermedia models for bridging the physical and digital worlds have been proposed. For example, *HyperReal* [16] is a mixed reality

model that introduces the concept of map components for managing geographical data. In addition, existing hypermedia solutions have also been challenged by new ideas such as the structural computing approach that treats structure as a first-class citizen and no longer puts the focus on the data [17].

As outlined, there is a wide variety of hypermedia models and systems. While there have been some attempts to provide reference models such as Dexter and FOHM, most hypermedia models and systems are isolated solutions for specific domains (e.g. navigational or spatial hypertext) or even specific applications. Although the Dexter model was instrumental in providing a common vocabulary, its specification is not detailed enough to enable information exchange between different systems based on the Dexter model or one of its extensions. Many models for hypermedia systems have claimed to be general and extensible and yet these have often disappeared only to be replaced by another hypermedia model. There is little or no support for evolution between these models with the result that applications and data are lost between implementations.

In our opinion, one of the causes for this situation is the lack of well-defined conceptual models on which implementations are based. Often models are presented as a mix of architectural, technical and conceptual features. As a result, the concepts become obfuscated and restrictions are introduced unnecessarily due to technicalities of the envisaged implementation.

Designing a system around a well-defined conceptual metamodel leads to increased generality and flexibility of both the model and the system. The use of metamodels as a basis for specifying and implementing hypermedia models is not widespread. In the field of web engineering where hypermedia models have been adapted to model navigation and adaptivity in web sites, metamodels are more commonly used and there have been efforts to define common metamodels (e.g. [18]). However, in this case, the metamodels tend to be focussed on the specific needs of web engineering.

Summarising, we feel that there is a need for a general framework to support the development of different categories of hypermedia systems and that this framework should be based on a general, extensible metamodel for hypermedia. The core of this metamodel has to be powerful enough to support the specification and modelling of different hypermedia domains in terms of a small set of fundamental link concepts. The development of the framework should be based on an implementation of the metamodel with the explicit representation of concepts of the metamodel in terms of metadata. While such a metamodel-driven approach to implementation is well-known to the conceptual modelling and database communities, along with its advantages in terms of flexibility and support for evolution, it is relatively rare to find it outside these communities and, in particular, in hypermedia systems. The result is that often model concepts are mapped to implementation-specific approximations that introduce restrictions and the model itself is hard-coded and static.

In the remainder of the paper, we present such a metamodel, the *resource-selector-link* (RSL) model, and describe how it was used to implement a general cross-media information platform called iServer [19]. We highlight how the RSL

model generalises concepts found in the range of hypermedia models mentioned above. Further we show how extensibility for domain-specific requirements is supported through a combination of concept specialisation in the metamodel and plug-in components in the architecture.

3 Link Metamodel

Our general metamodel for hypermedia systems was defined using the semantic, object-oriented data model OM [20]. OM is a data model that integrates concepts from both entity relationship and object-oriented data models. The OM model is intended as a basis for efficient data management as well as semantic expressiveness, and a family of object-oriented database platforms have been realised based on this model including the OMS Java data management system [21]. Using OM together with OMS allowed us to directly implement the metamodel and we were able to exploit powerful features of the OM model such as multiple classification and ordered collections in the metamodelling process. For that reason, we choose to use the OM modelling notation here rather than a more commonly used alternative. However, it is important to note that, even if another implementation platform were used, it would prove beneficial to base the system design on our OM metamodel which provides rich classification structures over objects and associations together with a full operational model.

The OM model supports information modelling through a two-level structure of classification and typing, dealing with these on separate layers. *Typing* deals with representation and entities are represented by objects with attributes, methods and triggers defined for the corresponding object types. *Classification* deals with semantic roles and a particular classification is represented by a named *collection* of objects with a specified member type. In addition, OM provides a high-level *association* construct which enables associations between entities to be classified and manipulated directly.

The OM model differs from many conceptual models in that it is intended as an operational model for data management as well as for system design. Thus the OM model defines a full operational model over objects, collections and associations as well as constructs for their definition. The expressive features of the OM model enable us to capture the semantics of application domains in terms of a simple, but powerful set of constructs. Its support for the direct representation and manipulation of associations is particularly useful in supporting link management in systems that offer hypermedia functionality. For more details about the OM model and its additional features please refer to [20].

In this section, we focus on the core link functionality of our general model whereas other parts of the RSL metamodel are presented in Sect. 4 to Sect. 6. The schema of the core link model is shown in Fig. 1. The shaded rectangular shapes denote collections of objects (classification) where the name of the collection is given in the unshaded part and the name of the associated type in the shaded part. The type serves both as a constraint on membership in the collection and also as the default view of objects accessed through that collection. Thus, links

are represented by objects of type `link` grouped into the `Links` collection. The shaded ovals represent *associations between entities* of two collections.

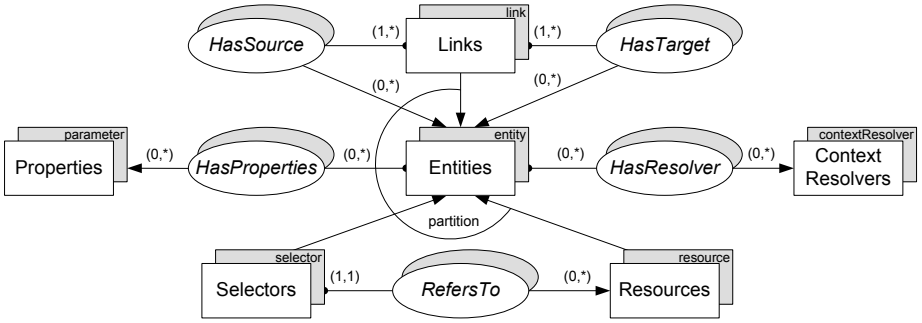

Fig. 1. Core link metamodel

The most general concept within our hypermedia metamodel is the generic notion of an `entity` (similar to components in the Dexter model). Note that all instances of entities are further classified and grouped by the collection `Entities`. While an entity represents only an abstract concept, there are three specific forms of entities described by three different subtypes: the `resource`, the `selector` and the `link` types.

The simplest type of entity is the `resource` type representing an entire information unit. While a `resource` is still an abstract concept, all types of media to be handled by the hypermedia system have to provide a specific extension of the `resource` type based on a plug-in mechanism. Our implementation of the hypermedia model, known as iServer, currently supports a variety of different resource types for web pages, movie and sound clips, images, Flash movies, physical objects marked with RFID tags as well as interactive paper. Note that we have a subtyping relationship between the `resource` and `entity` type but the specialisation of resources is also reflected in the model by designing `Resources` as a subcollection of the `Entities` collection.

Often we want to define links between not only entire resources but also specific parts of resources. For example, the anchor of a link within a web page addresses a specific part of an HTML document using the `href` anchor tag. Therefore, as a second subtype of the `entity` type, we introduce the concept of a `selector` which is a construct enabling parts of the related resource to be addressed (similar to reference objects in FOHM [8]). Again the specialisation of `Entities` is reflected by the `Selectors` subcollection. An association `RefersTo` represents the fact that a selector is always associated with exactly one resource, whereas each resource can have more than one referencing selector. The cardinality constraints specified at the source and target points of the associations indicate the possible level of participation of individual objects. Thus `(1,1)` at the source point of the `RefersTo` association indicates that each selector is

always associated with exactly one resource, whereas the `(0,*)` cardinality constraint at the target means that each resource can have zero or more referencing selectors. A `selector` is an abstract concept that has to be extended to support concrete types of media. For example, the selector to address parts of an XHTML document resource could be an XPointer expression, whereas the selector to specify parts of a movie clip could be a temporal selector with a start and an end time.

After providing a mechanism to allow entities to be referenced by a link, we now provide a specification for the links themselves. A link within our hypermedia metamodel is always directed and leads from one or more *sources* to one or more *targets*. A source may either be an entire `resource` or parts of a resource addressed by a `selector`. This is reflected in the model by making the collection `Entities` the target collection of both the `HasSource` and `HasTarget` associations. Furthermore, the `(1,*)` cardinality constraint at the source point of both associations indicates that each link must have at least one and possibly many sources and targets. In this way, we support multi-target links as well as links with multiple sources. The `(0,*)` at the target point of the `HasSource` and `HasTarget` associations specifies that there is no limit on the number of links for which an entity may be the source or target. Note that by ensuring that each link has at least one source and target entity, we prevent any occurrence of dangling links as proposed in the Dexter model and guarantee that the system is always in a consistent state where links can be resolved. For cases where the source or target entity is not available at link creation time, special placeholder elements could be used and replaced at a later time.

While there are many existing hypermedia models dealing with multi-target links, we found that links with multiple source anchors are not supported by most systems. However, from our experience of integrating information across different digital and physical information spaces, we can say that the concept of multi-source links is very powerful. For example, if the same information is published on different output channels (e.g. a web page and an interactive paper document) the semantics of a single link is maintained by associating it with two different sources for the two different types of media triggering the link resolution. Also note that since the underlying OM model provides bidirectional associations as a higher-level construct, all the associations used within the cross-media link model are also bidirectional. This enables us to, not only get all the link targets for a specific link source, but also to find the corresponding link sources given a specific target object.

By also modelling `Links` as a subcollection of `Entities`, we gain the flexibility to create links whose sources or targets are defined by other links. This means that we can annotate any link with supplementary information. While other systems also support the annotation of links with additional information, our approach of using the metamodel's link functionality for annotating links entails the advantage that links can not only be annotated with textual information but with any arbitrary entity. This means that we can use resources, parts of resources or even other links to annotate a link. For example, we could

have a web page with links to additional information and these links could then be annotated by other users with textual comments or links to different web resources etc. A final remark about the three core concepts (resource, selector and link) is that a `partition` is specified over the `Resources`, `Selectors` and `Links` subcollections to denote that each entity belongs to exactly one of these three categories.

To provide some additional flexibility for future extensions, each entity can be associated with a set of properties which are stored as a set of string tuples in an entity's property attribute. These properties, represented by key/value pairs, are not predefined by a system implementing the model. They can be defined individually to customise an entity's behaviour for specific application domains. For example, one could define a link property `onActivate` which would represent the action to be taken when a link is activated. Possible values could be `openInline` to open the link target within the current resource or `openNew` to display the link target in a separate view. This is similar to concepts in XLink [13] where the `actuate` attribute is used to define the traversal behaviour and the `show` attribute defines where a link should be shown (e.g. in the same or in a new window). However, we try to be as flexible as possible by not predefining a set of properties but rather introducing an abstract property set which can be extended for specific domains. Another example is to provide a flexible "typing" of links by introducing a property with the name `type` and assigning the appropriate values to it as proposed in the Dexter model. For instance, we could introduce a special type for links which represent annotations and treat them in a specific way. As an alternative, we could also introduce domain- or application-specific subcollections of `Links` as a means of classifying links. This combination of being able to associate properties to links and also classifying them provides a very flexible and powerful way of representing link taxonomies.

Finally, our core model provides functionality for the context-dependent handling of entities. Each entity can be associated with a set of context resolvers which are then used to compute an entity's visibility. A `contextResolver` basically returns a boolean value representing an entity's accessibility based on data managed by the hypermedia model as well as any other available contextual information. If multiple context resolvers are associated with a single entity, the entity will only be visible if all context resolvers return positive feedback. While the context resolver is an abstract concept, various domain- and application-specific context resolver extensions can be registered with the system.

By introducing the concept of context-dependent information at the very core of our model (i.e. at the entity level), we gain the flexibility of having context-dependent resources, selectors and links operating independently of each other. For example, a link with multiple targets may be accessible in a given context while, for the same context, some of its target entities may be inaccessible. The implementation of adaptive hypermedia functionality mentioned in the previous section is just one of the domains that can be supported by the context resolver concept. Entities can be easily tagged with different properties which will then be used in the decision process of specific context resolvers. A built-in context

resolver for handling access rights has to be provided by all systems implementing our hypermedia metamodel and is presented in the next section as part of the user management component.

4 User Model

In order to support both personalisation and the sharing of links and resources, we need a notion of *data ownership* combined with different levels of access rights. While most early hypermedia systems did not deal with an explicit representation of users as part of the model, some adaptive hypermedia models (e.g. AHAM [7]) introduced the concept of user models in the core of the system. However, while those user models typically deal with the aggregation and storage of user access patterns, our user model only provides functionality for managing data ownership and access rights at the entity level. The richer user models investigated by the adaptive hypermedia community could be integrated as a domain-specific extension of our metamodel. Note that even more recent link models such as the XLink standard do not provide the concept of data ownership nor do they deal with the definition of link access rights. By defining the access rights at the entity level, we can define individual permissions for links, resources and selectors. The representation of the fundamental user management component in our model is illustrated in Fig. 2.

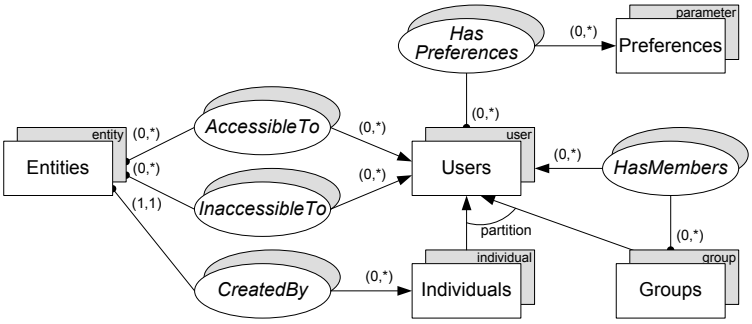

Fig. 2. User management

A `user` can either be an `individual` or a `group`. Users can be classified in different groups represented by the collection `Groups`, where a `group` itself can be part of other groups. Each `entity` is created by exactly one individual user having full control over its content. Note that the collaborative authoring of links and resources is possible due to the fact that the creator can define read and write access rights for other groups of users or individuals. The two associations `AccessibleTo` and `InaccessibleTo` are introduced to define access rights in a flexible way. The set of individuals having access to a specific entity is defined by the groups and individuals associated by `AccessibleTo` minus the groups

and individuals defined through the `InaccessibleTo` association. In addition, there exists a constraint that access rights defined for an individual always have priority over access rights that have been defined for a group. More formally, a group $\mathscr{G}$ is defined by some subgroups $\mathscr{G}_i$ and some individuals I_k that it contains, i.e. $\mathscr{G} = \{\mathscr{G}_1, \ldots, \mathscr{G}_m, I_1, \ldots, I_n\}$. The expanded set of individuals who are members of a group is given by

$$\mathscr{E}(\mathscr{G}) = \bigcup_{g \in \mathscr{G}} \mathscr{E}(g) \quad \text{with} \quad \mathscr{E}(I) \equiv \{I\}.$$

For a specific entity e, let $\mathscr{G}_a(e)$ denote the set of groups explicitly specified as having access to e and $\mathscr{G}_x(e)$ those explicitly denied access. Correspondingly, let $\mathscr{I}_a(e)$ denote the set of individuals explicitly specified as having access to e and $\mathscr{I}_x(e)$ those explicitly denied access. Then $\mathscr{A}(e)$, the set of individuals having access to entity e is defined as

$$\mathscr{A}(e) = \mathscr{I}_a(e) \cup (\mathscr{E}(\mathscr{G}_a(e)) \setminus \mathscr{E}(\mathscr{G}_x(e)) \setminus \mathscr{I}_x(e)).$$

This allows us to define complex access rights for an individual entity of the form "the entity should be visible to everybody except one specific group of users and two particular individuals". The activation of a link may depend on the user and even the user role. An author of a cross-media application based on our hypermedia metamodel may not only define different selectors for different users but also link the same selector to different information resources based on the user profile. Note that a specific context resolver can be used for ensuring entity access control based on the presented user model.

5 Layers

We have already introduced the concept of a selector to address parts of a resource as a link source or target entity. However, so far we have not explained how we deal with the case that the parts of a resource defined by different selectors overlap. For example, we could have one selector which specifies a paragraph within an XHTML document, while another selector specifies a word within that paragraph. The overlapping selectors can create a link resolution problem in terms of not knowing which link to activate when the word is selected. This is the problem of supporting so-called nested links. In the case of HTML, this problem does not arise as overlapping anchors are not allowed, but this is also quite restrictive and therefore a number of hypermedia models support some form of overlapping anchors (strictly nested and/or partly overlapping). But, even if nested link anchors are supported, it is often the case that the link resolution behaviour in the case of overlapping anchors cannot be specified. For example, the XLink specification allows for nested and overlapping link anchors but does not provide any functionality to control their behaviour. To become more flexible in defining the semantics of nested link source and target anchors, we introduce the concept of layers shown in Fig. 3.

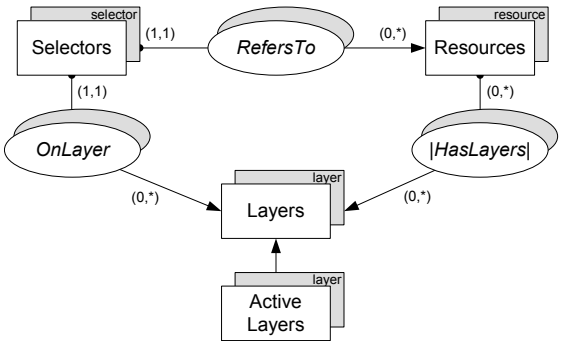

Fig. 3. Layers

Each selector is associated with exactly one layer and we do not allow overlapping selectors on the same layer, thereby forcing overlapping link source selectors to be defined on separate layers. In the case that a concrete selection would return several links by activating multiple overlapping selectors, by definition, the link bound to the selector on the uppermost layer will be selected.

The OM model supports collections of four different behaviours—sets, bags, rankings and sequences—to cater for collections with and without multiple occurrences of elements and with or without an explicit ordering. This also applies to associations. A selector's associated resource defines the set of available layers over the association |`HasLayers`| and the vertical bars indicate that this is a totally ordered association (ranking) that provides an explicit ordering of the layers. Furthermore, it is possible to activate and deactivate specific layers by adding them to or removing them from the `ActiveLayers` collection. The order defined by this association is used to choose the appropriate layer in the case that a selection addresses multiple overlapping selectors. Note that a selector can only be associated with a layer defined by its related resource over the |`HasLayers`| association.

Specific layers may be activated, deactivated and dynamically reordered enabling us to generate context-dependent links by resolving a particular selection to different selectors depending on the current set of active layers and the order of layering defined by the associated resource. An application may also control the navigational behaviour by switching the active layer set as a result of a user repeatedly providing the same selection.

6 Structural Links

As explained earlier, links are already first class objects in our model. By using links to describe structural components as well as navigational relationships between different resources, we place structure on the same level as resources and navigational links. Note that we do not give priority to structure over data as sometimes proposed by structural computing [17] but rather consider them to be on the same level.

In Fig. 4, we present an extension of our metamodel that distinguishes between structural and navigational links between resources. The collection `Links` introduced in Sect. 3 is partitioned into `Navigational Links` and `Structural Links`. By modelling structural links as a subcollection of regular links, they can be used to define a structure over arbitrary entities (e.g. resources, selectors and even links).

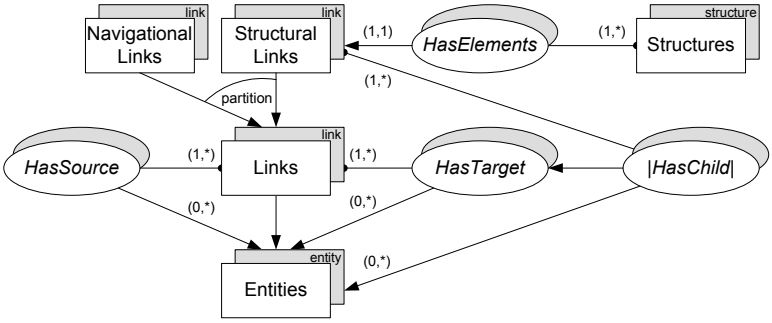

Fig. 4. Navigational and structural links

All structures are handled by the `Structures` collection and a single structure is related to its structural links by the `HasElement` association. It is necessary to have such an explicit grouping of structure elements since parts of structures might be reused by other structures as shown later in this section. Furthermore, structural links are a specialisation of general links since we have to introduce an order for the substructure relationship. For example, if we want to model the structure of a document with different chapters and sections within a chapter, we have to know the order of the chapter within the entire document as well as the order of the sections within the chapters. The order over such substructure relationships is introduced by the ordered |`HasChild`| subassociation of the `HasTarget` association. Therefore, the `Structures` and |`HasChild`| components provide us with information about all of the components belonging to a specific structure as well as their structural relationships.

A first type of structure that can be defined based on the new concepts is a structural relationship of different resources. An example of such a *structure over data* is a regular document containing chapters, sections, paragraphs etc. It is up to an application to define different domain-specific structures. Note that since we have a clear separation of data and structure, it is possible to reuse the same resource in different structures by transclusion as suggested by Nelson [22]. The context-dependent link resolution, discussed earlier as an example to support adaptive hypermedia, is also available for structural links. Therefore, the structure of a specific resource may change based on the user accessing it or any other context resolver-based adaptation.

Structural links can not only be defined directly over resources but we can also define a *structure over structures*. It is important to know that each structural

link within a structure defines a substructure containing the structural link's source elements and all of its children (recursively). This implies that we can superimpose any structure on top of existing structural components and, of course, the structural composition of data and substructures can also be combined. Note that to address a substructure we do not define a structural link to the `structure` element but rather to the corresponding structural link defining the substructure. An example of such a reuse of a substructure component could be a chapter of a document (e.g. a technical specification in an appendix) that is structurally referenced by different documents.

Last but not least, we can also specify a *structure over links*. This enables us to put different navigational links in relation to each other. A simple example would be the structure of an ordered list of navigational links defined by a single structural link. Such an ordered list of navigational links could be used to represent trails and tours, which are two well-known concepts available in many hypermedia models and systems. While we have indicated some examples for *structure over data*, *structure over structures* and *structure over links*, it is beyond the scope of this paper to discuss the potential applications of structural links in detail. We also point out that we are still investigating the implications of this powerful concept for different domains.

7 Implementation

The RSL metamodel presented in this paper is implemented using the OMS Java data management system [21]. The resulting iServer platform [19] for cross-media information management has been used over the last five years in a variety of projects for linking digital and physical information resources and, specifically, for the implementation of the iPaper framework [23] for interactive paper.

Since the metamodel was implemented on the OMS Java data management system, the iServer platform provides a Java API for accessing data as well as metadata. We have also implemented a Web Service interface providing the same functionality as the Java API to offer a more general language independent programming interface for the iServer platform. Other extensions include a distributed version of the platform where different distributed iServer instances can exchange link information based on peer-to-peer technology.

Different media-specific plug-ins for the resource and selector concepts have been developed including plug-ins for XHTML documents, movie and sound clips, still images, Flash movies, physical hypermedia based on RFID tagged physical objects and interactive paper. For a given resource type, there may be varying types of selectors based on the requirements of specific applications. In Table 1, we suggest time spans to be used as a candidate for movie selectors. However, a specific application might need to link movies based on spatial information within the movie whereas others might need to define links based on a combination of temporal and spatial information. The iServer architecture therefore supports the definition of different selectors for a single resource type.

Medium	Resource	Selector
paper	document page	shape
web page	XHTML document	XPointer
movie	mpeg file, avi file etc.	time span
movie	mpeg file, avi file etc.	shape
sound	mp3 file, wav file etc.	time span
image	gif file, jpeg file etc.	shape
database	database workspace	query
physical object	RFID space	RFID tag

Table 1. iServer plug-ins

To illustrate the flexibility of the iServer model and framework, we would like to provide some more details about the iWeb resource plug-in for linking XHTML documents. Similar to other link server approaches proposed by the hypertext community, our iWeb plug-in uses iServer as an external link repository for web pages. In contrast to most existing link servers, the iServer-based approach results in a flexible cross-media solution capable of integrating arbitrary digital or physical resources.

The definition of a selector for XHTML documents was straightforward since we could build on work already done in the context of XLink [13]. XLink uses the XML Pointer Language (XPointer) to address a specific part of an XML document. By using XPointer expressions as XHTML selectors within the iServer framework, we can define any part of an XHTML document as a link source or link target. However, as explained earlier, we obtain some additional features not available in the XLink language such as the well-defined semantics for multi-layered link resolution or the link ownership and access right control.

To integrate the link metadata stored in iServer with existing XHTML pages, we implemented an extension for the Firefox web browser. When a new web page is requested by the user, it is first downloaded from the server and immediately visualised in the browser. In a second step, the iWeb browser extension starts to parse the web page and augment it with supplemental link information that is acquired by contacting the iServer Web Service based on specific JavaScript functionality. As soon as the integrated web page has been rendered by the browser extension, the page gets redisplayed in the browser's main window.

The general resource and selector concepts together with the multi-layer functionality have proven to be powerful enough to support the integration of digital and physical objects. Different authoring tools haven been developed for creating and browsing link information managed by the iServer platform. For example, the iWeb Firefox extension can be used to augment arbitrary XHTML web documents in a similar way to the Annotea [15] project. Another authoring tool developed based on QuickTime technology enables parts of movie clips to be linked based on temporal selectors. An active component mechanism implemented on top of the iServer platform supports links to pieces of program code that can used as link targets and are executed when a link is activated [23].

A wide variety of applications have been developed using the iServer platform. These include several applications which use the iPaper plug-in to support interaction between paper and digital resources including an interactive guide for the Edinburgh festivals [24], PaperPoint [25], a paper-based interface for PowerPoint presentations and Print-n-Link [26], a system to support the reading of scientific publications. In addition, iServer has been used to support a number of interactive media installations designed by artists.

Although the system has been extended significantly over the last few years, this always happened as an evolutionary rather than a revolutionary process. This means that, even if the core of the model was extended, all applications evolved with the changes of the model due to our database-driven approach and even our first applications are still operational. For example, the recent extension of the framework to support structural links did not affect the operation of existing applications or the data managed by these applications. However, any new or existing application can now make use of the new structural links functionality. For instance, an application for the publishing of interactive paper documents now uses the structural link functionality to define a domain-specific document model.

8 Conclusions

We have presented RSL, a general metamodel for hypermedia systems dealing with data, structure and navigation information based on a core set of link concepts. Our conceptual modelling approach led naturally to a very general and flexible link model that integrates various concepts of existing hypermedia models. In addition, the RSL model caters for cross-media linking and provides extensibility for the introduction of new resource types. To show the flexibility of the model, we also described the iServer framework for cross-media information spaces which is based on RSL and supports a rich variety of applications.

References

1. Bush, V.: As We May Think. Atlantic Monthly **176**(1) (July 1945) 101–108
2. Halasz, F.G., Schwartz, M.: The Dexter Hypertext Reference Model. Communications of the ACM **37**(2) (1994)
3. Grønbæk, K., Trigg, R.H.: Design Issues for a Dexter-based Hypermedia System. Communications of the ACM **37**(2) (February 1994) 40–49
4. Hardman, L., Bulterman, D.C.A., van Rossum, G.: The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. Communications of the ACM **37**(2) (February 1994) 50–62
5. Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction **6**(2–3) (July 1996) 87–129
6. Parunak, H.: Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning. In: Proc. of Hypertext '91, San Antonio, USA (December 1991) 233–242
7. Bra, P.D., Houben, G.J., Wu, H.: AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia. In: Proc. of Hypertext '99, Darmstadt, Germany (February 1999)

8. Millard, D., Moreau, L., Davis, H., Reich, S.: FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains. In: Proc. of Hypertext 2000. (May 2000)
9. Nürnberg, P.J., Leggett, J.J., Wiil, U.K.: An Agenda for Open Hypermedia Research. In: Proc. of Hypertext '98, Pittsburgh, USA (June 1998) 198–206
10. Anderson, K.M., Taylor, R.N., E. J. Whitehead, J.: Chimera: Hypermedia for Heterogeneous Software Development Environments. ACM Transactions on Information Systems **18**(3) (2000) 211–245
11. Hall, W., Davis, H.C., Hutchings, G.: Rethinking Hypermedia: The Microcosm Approach. Kluwer Academic Publishers (May 1996)
12. Maurer, H.: Hyperwave: The Next Generation Web Solution. Addison Wesley (1996)
13. DeRose, S.J.: XML Linking. ACM Computing Surveys **31**(4) (December 1999)
14. Newcomb, S.R., Kipp, N.A., Newcomb, V.T.: The "HyTime": Hypermedia/Time-based Document Structuring Language. Communications of the ACM **34**(11) (November 1991) 67–83
15. Kahan, J., Koivunen, M.R., Prud'Hommeaux, E., Swick, R.R.: Annotea: An Open RDF Infrastructure for Shared Web Annotations. In: Proc. of WWW10, 10th International World Wide Web Conference, Hong Kong (May 2001)
16. Romero, L., Correia, N.: HyperReal: A Hypermedia Model for Mixed Reality. In: Proc. of Hypertext 2003, Nottingham, UK (August 2003) 2–9
17. Nürnberg, P.J., schraefel, m.c.: Relationships Among Structural Computing and Other Fields. Journal of Network and Computer Applications **26**(1) (2003) 11–26
18. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. In: Proc. of ICWE 2003, 3rd International Conference on Web Engineering. (July 2003) 497–506
19. Signer, B.: Fundamental Concepts for Interactive Paper and Cross-Media Information Management. PhD thesis, ETH Zurich, Switzerland (2006)
20. Norrie, M.C.: An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In: Proc. of ER '93, 12th International Conference on the Entity-Relationship Approach, Arlington, USA (December 1993) 390–401
21. Kobler, A., Norrie, M.C.: OMS Java: A Persistent Object Management Framework. In: Java and Databases. Hermes Penton Science (May 2002) 46–62
22. Nelson, T.: Literary Machines. Mindful Press (1982)
23. Norrie, M.C., Signer, B., Weibel, N.: General Framework for the Rapid Development of Interactive Paper Applications. In: Proc. of CoPADD 2006, 1st International Workshop on Collaborating over Paper and Digital Documents, Banff, Canada (November 2006) 9–12
24. Belotti, R., Decurtins, C., Norrie, M.C., Signer, B., Vukelja, L.: Experimental Platform for Mobile Information Systems. In: Proc. of MobiCom 2005, 11th Annual International Conference on Mobile Computing and Networking, Cologne, Germany (August 2005) 258–269
25. Signer, B., Norrie, M.C.: PaperPoint: A Paper-Based Presentation and Interactive Paper Prototyping Tool. In: Proc. of TEI 2007, First International Conference on Tangible and Embedded Interaction, Baton Rouge, USA (February 2007) 57–64
26. Norrie, M.C., Signer, B., Weibel, N.: Print-n-Link: Weaving the Paper Web. In: Proc. of DocEng 2006, ACM Symposium on Document Engineering, Amsterdam, The Netherlands (October 2006) 34–43