Modeling the Dynamic Behavior of Hypermedia Applications

Paloma Díaz, *Member*, *IEEE*, Ignacio Aedo, and Fivos Panetsos

Abstract—Hypermedia applications can be defined as collections of interactive and multimedia documents that are organized as a hypertext net. The development of hypermedia applications poses specific problems, such as the need for modeling sophisticated navigational structures, interactive behaviors, and harmonic presentations involving the synchronization of contents. Moreover, the increasing popularity of Internet-based systems has put stress on the lack of mechanisms to formally specify security policies when designing hypermedia applications. Traditional design models and methodologies are not suitable for hypermedia applications and the up-to-now developed hypermedia-oriented models do not cover the whole set of design needs. In this context, we present Labyrinth, a hypermedia-oriented model providing formal elements to describe the static structure and dynamic behavior of this kind of nonlinear, multisensory, and interactive applications.

Index Terms—Hypermedia, design representation, formal model, multimedia, interactive behavior, security, synchronization, space-based relations, events, Labyrinth.

1 Introduction

Hypermedia applications can be defined as collections of multimedia documents that are organized into a hypertext net. While multimedia supplies a greater expressiveness, hypertext provides a geometry that allows information to be browsed and presented according to the needs and preferences of users. Moreover, multimedia also introduces the concept of interactivity which increases the system utility, offering the user a wide variety of opportunities to play an active role in the information transmission processes. Indeed, many educational hypermedia/multimedia environments include a number of different interactive activities oriented toward reinforcing the learning process of their students [1], [2], [3].

The development of hypermedia applications poses particular problems which do not appear in other software applications, such as the need for modeling sophisticated navigational structures, interactive behaviors, and presentations where multimedia contents are harmonized in several dimensions. Moreover, the increasing popularity of Internet-based systems has put stress on the lack of mechanisms to formalize security policies when designing hypermedia applications. Rational solutions to all these problems can be planned during the design phase if designers can rely on abstract models to formally specify the structure and behavior of the application under development. Since traditional design models and methodologies are not suitable for hypermedia applications [4], [5], a number of hypermedia-oriented models have been proposed to help designers to face their conceptualization tasks (such as [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]).

The majority of the hypermedia-oriented models developed to date are rather limited, since they are influenced by the characteristics of a specific authoring tool, programming language, database structure, or physical data representation. Even reference models like Trellis [8], Dexter [9], and Amsterdam [12] cannot be used to define some features of hypermedia applications [14]. For instance, Dexter and Amsterdam do not include elements to formalize security policies oriented toward protecting information from unauthorized or improper accesses (e.g., the typical access control implemented in most web servers). Besides, Dexter, Amsterdam, and Trellis do not provide mechanisms to explicitly define composite objects involving heritage mechanisms, despite being the basis of several hypermedia authoring tools (e.g., the concept of "background" in HyperCard or Toolbook). More recent models, such as OOHDM [15] or RMM [16], do not thoroughly delve into the synchronization problem although its relevance has been repeatedly stated [17], [12].

Labyrinth is a hypermedia-oriented model which covers the lacks detected in the aforementioned models and provides formal elements to describe the static structure and dynamic behavior of this kind of nonlinear, multisensory, and interactive applications [14]. Since the model is platform-independent, implementation issues, such as distribution of information and services, are not considered. To deal with such issues, a specific layer has to be implemented under the model specification. This model is divided into two tightly related parts: a static part, that includes elements to specify hypermedia applications and, a dynamic one, which is made up of a number of operations that can be performed over the elements of the static part. The present article deals with the specification of behaviors

- *P. Díaz and I. Aedo are with the Laboratorio DEI, Departamento de Informática, Universidad Carlos III de Madrid, Avda. de la Universidad 30, E-28911 Leganés, Spain. E-mail: {pdp@inf, aedo@ia}.uc3m.es.*
- *F. Panetsos is with the Departamento de Matemática Aplicada, Facultad de Biología, Universidad Complutense de Madrid, Avda. Complutense S/N, E-28040 Madrid, Spain. E-mail: fivos.panetsos@bio.ucm.es.*

Manuscript received 8 July 1998; revised 3 Mar. 1999; accepted 13 Oct. 1999. Recommended for acceptance by C.E. Landwehr.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 107118.

in hypermedia applications using the Labyrinth model, whose static part was thoroughly presented in [14].

We present a discussion about issues concerning the design of hypermedia systems and we analyze how different models, including Labyrinth, face such issues in Section 2. The main features of Labyrinth are described in Section 3, while its use is illustrated through several examples in Section 4. Section 5 includes the conclusions drawn from this work as well as some ongoing developments.

2 Design Issues for Hypermedia Systems

Models are often defined in information technology as abstract devices that can be used to formally represent the structural components and function of software applications. Depending on how detailed and easy to understand this representation derived from the model is, it can be used either to initiate the system implementation or, as a basic document, to analyze the application requirements with the target users. In any case, a model constitutes a cornerstone of the design stage of a software application, provided that it contains elements to gather both the static structure and the dynamic behavior of the applications.

In fact, each hypermedia development platform has an underlying model which defines the type of elements that can be embedded in the application and the processes that can be implemented. Thus, authoring tools (such as Hypercard, Authorware, or Director) impose a particular design discipline based on the metaphor they implement (stacks, flowcharts, and time baseline, respectively) and a specific way of defining behaviors constrained by the programming languages they support (HyperTalk, icon-based language, and Lingo, respectively). Likewise, mark-up languages (such as HyTime, HTML, or XML) offer elements to structure in a specific way hypermedia applications and define to some extent their hypermedia features (including links and multimedia contents). In this case, interactive behaviors are included by embedding scripts, by means of tags or SMSL (Standard Multimedia/Hypermedia Scripting Language [18]). An example of increasing the declarative document model provided by a mark-up language (SGML and HyTime) using scripts is the Metafile for Interactive Documents (MID) [19].

Anyway, the use of this kind of specific models is very restricted, as far as they produce specifications that are suitable for a particular environment but can not be easily translated to a different one. Thus, the design can not be reused if the development is moved on to a different implementation platform and the effort required to learn how to use the model is not repaid, unless the same platform is always used. For this reason, several models have been proposed in order to generate platform-independent specifications of hypertext and hypermedia applications.

A number of representation abilities can be required of such models as far as they are intended for gathering the semantics of hypermedia systems. Since models will be used to specify hypertexts, they have to provide elements to represent their structure and browsing semantics in both a declarative and a procedural way. The model should also take into account special features introduced by the use of multimedia, such as the representation of different kinds of information items, the harmonic and aesthetic combination of contents, and the definition of interactive behaviors. Moreover, models should gather the characteristics of existing hypermedia applications, which are mainly integrated into multiuser environments. Among others, models should provide conceptual solutions for three basic issues: the definition of groups of users, the management of personalised views of the application, and the specification of security policies aimed at ensuring information integrity and confidentiality.

In the following sections, each of these requirements are presented and we discuss the manner in which they are tackled by Labyrinth and similar models.

2.1 Specification of the Structure

Most hypermedia applications have an intrinsic structure made up of semantic relationships among their components which should not be confused with the navigation paths offered to their users. For instance, there is a clear structural relationship among the pages of an electronic book, independent of the supported navigation capabilities.

The structure of a hyperdocument can become blurred when information holders (nodes) and information pieces (contents) are considered and treated at the same level, confounding the logical structure of the application with the contents delivered to the user. This is the case with models like Dexter and Amsterdam that use the same element, called component, to represent both structure and content, though Amsterdam at least makes the contents (called data blocks) be the leaves of the hierarchy of components. There is a need to clearly distinguish between structure and information following the approach introduced by Tompa's model [20] and assumed in models like Trellis and Labyrinth. The latter uses two different elements to represent structure and information items: nodes and contents, each having a different composition. For instance, a content has a representation space, described in the next section, which is unnecessary for a node.

This separation between structure and content provides a number of benefits, including cleaner specifications (where elements that are conceptually different are defined through distinct elements of the model) and the ability to share contents by reference, as web pages do with images and other embedded contents. But, it does not provide means to represent the inherent structure of most hypermedia applications. To model complex structures, composition mechanisms are required. In particular, two abstraction mechanisms are quite common in hypermedia applications: aggregation and generalization [21]. Aggregation allows different elements to be referred to by means of a single composite element. For instance, a table of contents of an electronic book aggregates all of its chapters. Aggregated elements remain independent and do not share properties. Generalization defines a composite element whose components will inherit all its properties. For instance, each page of an electronic book can be generalized by means of a generic page where common presentation features are held. Elements sharing a number of characteristics and/or behaviors can be grouped by means of this abstraction mechanism. Both aggregation and generalization are used

in OOHDM and in the hypertext-oriented model of Richard and Rizk [22].

In Labyrinth, both abstractions are supported not only to define composite nodes but also composite contents, so that design solutions to complex problems can be specified, as will be shown in the examples of Section 4. On the basis of these composition mechanisms, we now introduce the concept of domain. The domain of an object o includes the object itself, the domains of all the objects o' aggregated by o, and the domains of all the objects o'' generalized by o, that is:

$$\begin{aligned}\text{domain}(o) = o \cup \text{domain}(o') \cup \text{domain}(o'')| \\ \forall o', o' \in \text{aggregatedBy}(o) \\ \forall o'', o'' \in \text{generalizedBy}(o).\end{aligned}$$

All Labyrinth operations concerning nodes or contents can also be applied to domains so that the operation will involve all the elements belonging to the domain.

Finally, structures can be specified in a declarative way but also in a procedural one as in [17]. For instance, the spatial structures generated by hypertext users described in [23] can be treated as virtual composites whose items and organization can only be known at runtime. Such virtual composites are modeled in Labyrinth through event-based specifications. In this case, the declarative definition of the composite is increased by means of an event, whose actions will determine the identity of its components.

2.2 Representation of Multimedia Information

The representation space of a multimedia content can be defined using different dimensions (e.g., frames, shots, scenes, and sequences can be relevant segmentation units in a movie [24]). Coordinates referring to such dimensions allow specific fragments of the contents to be univocally pointed at, whether to set the source or the target of a link (that is, an anchor) or with other purposes (e.g., to make a video start from a particular scene). No hypermedia model provides specific elements to define such spaces except for Labyrinth, which introduces the possibility of defining for each content as many representation spaces as needed and combining them to specify regions of that content. For instance, an anchor specification can state that it starts in the third word of the second paragraph of a text and its length is of three words.

Moreover, multimedia presentations should be aesthetically composed in two dimensions: space and time. Consequently, mechanisms to establish space-based relations (called alignment in this paper) and synchronizations among contents are needed. The need for synchronizing multimedia presentations, stated in [17], takes shape in the synchronization mechanism proposed in Amsterdam [12], but no method to specify space-based relationships is provided. To compose the multimedia presentation using the Labyrinth model, contents can be located into nodes establishing a pair of coordinates which belong to the node space and time axes, respectively. Another option to procedurally define the position of a content consists of using the operations to set alignments and synchronizations among contents. For example, two video files can be compelled to be presented without any spatial overlapping (space-based relation) and to end at the same time (synchronization). Conditional space- and time-based relationships can be defined by means of events associated to nodes, contents, and links. Indeed, the different types of synchronizations and alignments proposed as technical requirements in the MHEG standard [25] (script, condition, spatio-temporal, and system) are encompassed in the model.

Another relevant characteristic to be considered is the interactive nature intrinsically tied to multimedia applications. Users not only can interact with the system by selecting links, but they can also take part in the information delivery process through a number of activities which include control commands (such as stop, forward, rewind, pause, etc.), as well as more creative actions (such as dragging and dropping objects, writing, drawing, etc.). Models like OOHDM use events only for presentation purposes. This concept is enlarged by Labyrinth which makes use of events also to model interactive behaviors. An event describes a process carried out when some conditions are fulfilled (e.g., a message is displayed when two moving objects meet). The condition of a Labyrinth event encompasses the notion of event as "a nonpersistent occurrence in the world" (e.g., clicking the mouse), as well as the concept of state as "a thing that persists and we observe in the world" (e.g., an object is in a particular spatial position), both according to the definitions of [26]. The process executed when the event is triggered is expressed using the set of operations making up the dynamic part of the model. Thus, events offer a great potential to specify interactive behaviors. In addition, since events are independent elements that can be tied to several nodes, contents, or links, they can be reused in the same or other application. The modeling of events has also been encountered in the object-oriented model presented in [27], although it does not comprise a set of operations for the formal specification of interactive behaviors as Labyrinth does.

2.3 Specification of Browsing Semantics

Navigation using links is an essential function of hypermedia systems. Links are usually identified with hotwords and icons that users can click to move onto a different node. However, literature shows more complex links which can be embedded into any area or point of any kind of multimedia content and can have several sources or targets (n-ary or multiheaded links [7], [9], [12], [28]). The concept of anchor, first introduced by the Dexter model, has been proven to be a very powerful tool when representing the source and target of a link. However, to set an anchor into a multimedia content, a mechanism to point at a specific region of the content is required. Such a region is defined using coordinates of a representation space which depends on the nature of the involved content as it was stated in the previous section. Therefore, designers need a conceptual tool to define that space equivalent to the one provided in the Labyrinth contents.

In Labyrinth, a link is defined between two sets of anchors, sources, and targets, respectively. Anchors can be declaratively specified in five different ways: referring to a whole node, referring to a node area or interval, referring to a whole content, referring to a region of a content's

TABLE 1
Hyperdocument: Static Definition and Dynamic Management

$A = \{A_i \mid i = 0,...n, n \in N \text{ where } A_i = (AnchorId_i, NodeId_i, ContentId_i, AnchorPos_i)\}$
$AnchorId_i \in \{\text{alphanumeric strings}\}$
$NodeId_i = -1 \cup \{NodeId_j \mid \exists N_j \in N, NodeId_j \in N_j, j = 0,1,...,n, n \in N\}$
$ContentId_i = -1 \cup \{ContentId_j \mid \exists C_j \in C, ContentId_j \in C_j, j = 0,1,...,n, n \in N\}$
$AnchorPos_i = \{Position_i, Extension_i\}$
$Position_i$ = Location of the initial point of the anchor in the node or content
$Extension_i$ = Extension of the anchor into the node or content

Operation name	Updates	Description of the operation
createAnchor	A --- A^P	A new anchor is added, whether in the basic hyperdocument (A) or in a personalized one (A^P).
deleteAnchor	A, L, al, el --- A^P, L^P, al^P, el^P	An existing anchor is removed from the basic hyperdocument (A) or from a personalized one (A^P). If it is the unique source or target of a link, then the link is also removed and, therefore, this operation may also modify L, al and el or L^P, al^P and el^P.
modifyAnchor	A --- A^P	It modifies the values of $NodeId_i$ and/or $ContentId_i$ of an anchor definition, whether in the basic hyperdocument (A) or in a personalized one (A^P).
modifyAnchorPosition	A --- A^P	It modifies the position and/or extension of an anchor, whether in the basic hyperdocument (A) or in a personalized one (A^P).
getLinksAnchor	none	It returns the list of the links associated to an anchor. L or L^P is accessed but not modified.

representation space, or referring to a content or content region only when it is presented in a specific context (node).

Link selection can be used to present additional information in the same node without moving to a different context. To include such a function, the Amsterdam model introduced the idea of context, where the elements that appear/disappear when a link is activated can be specified. Such contexts are modeled in Labyrinth using event-based specifications and, consequently, more powerful span links can be defined (e.g., the result of activating a link can depend on the user who selected it).

More complex behaviors can be achieved using virtual links [29] whose target or source depends on parameters that can only be known in runtime and, therefore, they are created when needed. For instance, sources of Microcosm links [30] are not embedded in the node specification but are added once a node has been activated. Other kinds of virtual links whose targets are dynamically calculated are hot [31], warm [31], and conditional links [32]. The set of sources and targets that make up a Labyrinth link can be dynamically calculated or modified using event-based specifications. The constraint imposed in Dexter about dangling links, that is, links whose source or target does not exist, has been removed in Labyrinth as in Hyperdisco [28] to represent both dynamically created anchors and references to external elements which do not belong to the application.

Links can also be used to represent semantically different relationships; they can be embedded into different types of information and their activation can produce a broad variety of results. For these reasons, the link definition has been complemented with attributes in models like [33]. Labyrinth distinguishes among the types of links (aggregation, generalization, referential, and version) and other attributes which can represent a great number of characteristics including the link direction (uni- or bidirectional), author, or presentation specifications.

2.4 Support for Personalized Views

Large hypermedia applications are not designed to meet the needs and specific interests of each individual user, especially in web environments usually addressed to a broad and anonymous audience. The general structural, presentation, and browsing features of the application might not satisfy the personal needs or preferences of some users who can be frustrated by a lot of useless information organized in a complex hyperstructure. Private views can be used to modify the user's hyperdocument view, hiding useless information and redefining the structure of the hyperdocument to a more familiar one. Moreover, hypermedia offers an ideal environment for collaborative work [34], where groups of users cooperate to reach a common goal. With this purpose, groups and even individual users can require private spaces to develop their work.

Private views have been explicitly gathered in Neptune's contexts [35] and in the Trellis model. In the former, only nodes and links can be personalized and Trellis proposes a complex mechanism consisting of a special mark-up for a Petri net to define each personalization. In Labyrinth, the

TABLE 2
Users: Static Definition and Dynamic Management

$L = \{L_i \mid i = 0,...,n, n \in N$ where $L_i = (LinkId_i, LinkStart_i, LinkTarget_i)\}$ $LinkId_i \in \{$alphanumeric strings$\}$ $LinkStart_i = \{AnchorId_{ij} \mid \exists A_j \in A, AnchorId_{ij} \in A_j, j = 0,1,...,n, n \in N\}$ $LinkTarget_i = \{AnchorId_{ij} \mid \exists A_j \in A, AnchorId_{ij} \in A_j, j = 0,...,q, q \in N\}$		
Operation name	**Updates**	**Description of the operation**
createLink	L, al --- L^P, al^P	A new link is added, whether to the basic hyperdocument (L) or to a personalized one (L^P). To set the value of the mandatory attributes, al or al^P is modified.
deleteLink	A, L, al, el --- A^P, L^P, al^P, el^P	An existing link is removed, whether from the basic hyperdocument (L) or from a personalized one (L^P), as well as its anchors provided that no other components are associated to them. The list of events and attributes must also be updated. Therefore, A, al and el or A^P, al^P and el^P are modified.
activateLink	none	It activates a link. A or A^P is accessed but not modified.
getLinkTargets	none	It returns the list of targets of a given link. A or A^P is accessed but not modified.
getLinkSources	none	It returns the list of sources of a given link. A or A^P is accessed but not modified.
getILinksTo	none	It returns the list of links whose target is a specific node or content. A and L or A^P and L^P are accessed but not modified. If the element is a generalized node or content, then a recursive process retrieves the inherited links.
getLinksFrom	none	It returns the list of links whose source is a specific node or content. A and L or A^P and L^P are accessed but not modified. If the element is a generalized node or content, then a recursive process retrieves the inherited links.

concept of personalized hyperdocuments is a primary component of the definition of a hypermedia application, by means of which elements of a finer granularity than nodes and links can be personalized (e.g., events). Operations to personalize elements do not affect the basic document, similar to the PIE hypermedia system [29]. The model also includes operations to transfer personalized elements to the Basic Hyperdocument, where they become public. Finally, Labyrinth introduces the definition of users and groups of users, both of which can work in their own Personalized Hyperdocuments only visible to their owners.

2.5 Inclusion of Elements to Specify Security Policies

Hyperdocuments are usually implemented as collaborative environments that make use of intranets or web servers in order to provide information access to different users. In such environments, security becomes a main concern since both integrity and confidentiality of information have to be preserved. Security is not only an implementation issue. Security models can be used during the design stage to translate the rules that will govern information access into clearances assigned to user roles. An approach to gather security constraints consists of using a security model to specify the rules that will control the hyperdocument access, such as [36], and a hypermedia model to specify the remaining characteristics of the application (e.g., navigational structures, synchronizations, etc.). In contrast to this approach, which can be cumbersome from a designer's point of view, we propose the use of a unique model to specify all the characteristics of hypermedia applications, including those concerning security issues.

Two models, Trellis and the extended RMM, propose the definition of ad hoc hyperdocuments for each user to gather security constraints. Hyperdisco uses discretionary ACLs to grant users some privileges although such mechanisms can only be valid if a high degree of security is not required [37]. Other researches in this field are oriented toward more strict mandatory models, including the multilevel security policy proposed by Thuraisingham [38] which is based on an extremely basic hypertext-oriented model. Moreover, its security levels are defined on the basis of the typical privacy levels of a database or information system (unclassified, secret, and top secret) which can seem inappropriate for hypermedia systems where integrity is far more important than confidentiality.

In the Labyrinth model, the inclusion of users and the definition of access categories provide a formal basis for the specification of security policies using the principles of information flow security models, although considered from an information manipulation perspective. Security categories are used to classify the kind of actions a user can carry out in a hyperdocument or domain (browsing, personalizing, and editing, each one enlarging the privileges of the previous one). These security categories, or clearances, are used by an access control function to

TABLE 3a
Nodes: Static Definition and Dynamic Management

$N = \{N_i \mid i = 0,...,n, n \in \mathbf{N}$, where $N_i = (NodeId_i, NodeCategory_i)\}$ $NodeId_i \in$ {alphanumeric strings} $NodeCategory_i \in$ {browsing, personalizing , editing}		
Operation name	**Updates**	**Description of the operation**
createNode	N, al, ac --- N^P, al^P	It adds a new node, whether in the basic hyperdocument (N) or in a personalized one (N^P). In the first case, the users that will be able to access the node have to be specified in the ac function. In either case, some mandatory attributes have to be tied to the node by including new values into al or al^P.
deleteNode	N, A, L, lo, al, el, ac --- N^P, A^P, L^P, lo^P, al^P, el^P	It removes a node from the basic hyperdocument (N) or from a personalized one (N^P). The function tuples where the node appears are also deleted and, therefore, lo, al, el and ac or lo^P, al^P and el^P are updated. Anchors and links embedded into the node must also be updated and so A and L or A^P and L^P are modified. Since contents, attributes and events are independent, a cascade deletion is not done.
duplicateNode	N, A, L, lo, al, el, ac --- N^P, A^P, L^P, lo^P, al^P, el^P	It duplicates a node along with its elements (location of contents, anchors, attributes and events), except from the incoming links, whether in the basic hyperdocument (N) or in a personalized one (N^P). The same components as in "deleteNode" and, for the same reasons, are modified.
setNodeCategory	N --- N^P	It assigns a security category to a node, whether in the basic hyperdocument (N) or in a personalized one belonging to a group of users (N^P). Since individual personalizations are only accessed by their owners, there is no benefit in allowing their ac function to be modified.
generalizeNode	A, L, al --- A^P, L^P, al^P	It creates a relation of generalization between nodes N_i and N_j in the basic hyperdocument or in a personalized one, making the elements associated to N_i (location of contents, anchors, events and attributes) be inherited by N_j. With this purpose, a special type of structural link (called generalization link) is set between them and, therefore, A, L and al or A^P, L^P and al^P are updated.
aggregateNode	A, L, al --- A^P, L^P, al^P	It creates a relation of aggregation between nodes N_i and N_j in the basic hyperdocument or in a personalized one, making N_j to be referred to by means of the identifier of N_i. With this purpose, a special type of structural link (called aggregation link) is set between them and, therefore, A, L and al or A^P, L^P and al^P are updated.
decomposeNode	A, L, al --- A^P, L^P, al^P	The result of a generalization or aggregation is removed, whether from the basic hyperdocument or from a personalized one. That is, the corresponding generalization or aggregation link is deleted. Since heritage is perfomed in runtime, these operations will not give place to a cascade deletion of the elements tied to the generalized nodes.
createNodeVersion	N, A, L, lo, al, el, ac --- N^P, A^P, L^P, lo^P, al^P, el^P	It creates a new version of a node, whether in the basic hyperdocument or in a personalized one. That is, a duplicated node is created and connected to the original node through a version link. The same components as in "duplicateNode" are updated.

constrain the manipulation ability of users according to their role in a particular context (node, content, or domain). Moreover, objects (nodes, contents, and domains) have a security category which determines the most permissive kind of operation they can undergo. When a user initiates an operation, a security check matches the user category against the security category of the involved objects to decide whether to proceed or not.

TABLE 3b
Nodes: Static Definition and Dynamic Management (cont.)

deleteNodeVersion	N, A, L, lo, al, el, ac --- N^P, A^P, L^P, lo^P, al^P, el^P	One version of a node is removed, whether from the basic hyperdocument or from a personalized one. That is, a node is deleted and also the link connecting it to the previous and later version, that are now connected by a new link version. The same components as in "createNodeVersion" are updated.
moveNode	HD^B, HD^P	A node is moved from the basic hyperdocument (HD^B) to a personalized one (HD^P) or vice versa. Intuitively, this operation is equivalent to the following process: the original node is duplicated and it is included into the other hyperdocument, keeping the same identifier. If the original node is being moved from HD^P to HD^B, then it is deleted. Otherwise, the original node also remains in HD^B so other users can access it.
getGeneralizedNodes	none	It returns the list of the nodes that generalize a given node. N, A and L or N^P, A^P and L^P are accessed but not modified.
getAggregatedNodes	none	It returns the list of the nodes that aggregate a given node. N, A and L or N^P, A^P and L^P are accessed but not modified.
getNodeComponents	none	It returns the list of the nodes that are generalized or aggregated by a given node. N, A and L or N^P, A^P and L^P are accessed but not modified.

3 The Labyrinth Model

This section describes the Labyrinth components from both perspectives, static and dynamic. The dynamic part, that offers a number of operations concerning the usage and management of hypermedia applications, is studied in depth since the static part is thoroughly presented in [14].

The use of the Labyrinth model to gather the characteristics of hypermedia applications can require the combination of a declarative model of hyperdocument with event-based specifications. The declarative model can be used to represent those features of hyperdocuments which are expected to remain stable in the sense that they do not depend on any external factor and they have a meaningful existence by themselves. However, several hypermedia applications, such as dynamically created web pages or adaptive systems, involve structures, contents, or interfaces that are created at runtime, that is, what Halasz called virtual objects [29]. In such cases, the declarative model has to be enlarged with event-based specifications where both the conditions constraining the object creation and the process building the object are established.

3.1 The Basic and Personalized Hyperdocument (HD^B and HD^P)

The Labyrinth model defines a hypermedia application as the union of a **Basic Hyperdocument** and a **Personalized Hyperdocument** (see Table 1). A hyperdocument is a fully connected hypermedia application made up of nodes connected through links, in such a way that each node is connected to a "hub" node, whether directly or not [39]. The "hub" node plays an important role in the hyperdocument definition since it represents the hyperdocument itself and contains information concerning the entire application.

The Basic Hyperdocument (HD^B) includes the public components of the application while the Personalized Hyperdocument (HD^P) consists of a number of private views (HD^Pi) only accessed by their owners, whether individual users or groups. The HD^B consists of sets of Users (U), Nodes (N), Contents (C), Anchors (A), Links (L), attriButes (B), and Events (E) and the functions of location (lo), attributes list (al), events list (el), and access list (ac), that will be described below. Each HD^Pi is composed of the user identifier of the personalization owner ($UserId$) and adaptations of items defined in the basic hyperdocument, except for the access list (ac) that need not be personalized according to the definition of Personalized Hyperdocument (it can only be accessed by its owners and, therefore, no access rights need to be specified). Thus, N^P, C^P, A^P, L^P, B^P, and E^P are either composed of new elements or modifications of the corresponding components of HD^B. Similarly, lo^p, al^p, and el^p are also composed of new or modified functions.

The dynamic part of the model includes operations that allow hyperdocuments to be created and removed, as summarized in Table 1. Each operation of the dynamic model includes a procedure that, following the security model explained in [40], is executed to decide whether the operation can or cannot be done. Such a procedure analyzes information about the subject (user) and the objects involved in the different actions to be carried out. Subjects require a clearance not lower than the security category of the objects to be granted permission. If this security check does not succeed, the operation is not performed. Each operation is treated as a transaction which is only executed if all its actions are considered safe.

Each of the elements of the basic and personalized hyperdocuments are described below.

3.2 The Set of Users (U)

Users of hypermedia applications, whether individuals or groups, are included in this set whose static definition is shown in Table 2. Each user (U_i) is made up of a $UserId$

TABLE 4a
Contents: Static Definition and Dynamic Management

$C = \{C_i \mid i = 0,...,n, n \in \mathbf{N}$ where $C_i = (ContentId_i, ContentCategory_i, ContentType_i)\}$
$ContentId_i \in$ {alphanumeric strings}
$ContentCategory_i \in$ {browsing, personalizing , editing}
$ContentType_i = \{Type_i, Units_i\}$
$Type_i \in$ {text, video, program, picture, ...}
$Units_i = \{x_{ij} \mid j = 1,...,n, n \in \mathbf{N}\}$, $x_{ij} \in$ {pixels, characters, frames,...}

Operation name	Updates	Description of the operation
createContent	C, al, ac --- C^P, al^P	It adds a new content, whether in the basic hyperdocument (C) or in a personalized one (C^P). In the first case, the users that will be able to access the content have to be specified in the ac function. Anyway, some mandatory attributes have to be tied to the content by including new values into al or al^P.
deleteContent	C, A, L, lo, al, el, ac --- C^P, A^P, L^P, lo^P, al^P, el^P	It removes a content from the basic hyperdocument (C) or from a personalized one (C^P). The function tuples where the content appears are also removed and, therefore, lo, al, el and ac or lo^P, al^P and el^P are updated. The anchors and links embedded into the content must also be updated, so that A and L or A^P and L^P are modified. Since nodes, attributes and events are independent a cascade deletion is not done.
duplicateContent	C, A, L, lo, al, el, ac --- C^P, A^P, L^P, lo^P, al^P, el^P	It duplicates a content along with its elements (anchors, attributes and events), except from the incoming links, whether in the basic hyperdocument (C) or in a personalized one (C^P). The same components as in "deleteContent" are updated.
setContentCategory	C --- C^P	It assigns a security category to a content, whether in the basic hyperdocument (C) or in a personalized one belonging to a group of users (C^P). Since individual personalizations are only accessed by their owners, there is not worth allowing their ac function to be modified.
generalizeContent	A, L, al --- A^P, L^P, al^P	A relation of generalization between contents C_i and C_j is established in the basic hyperdocument or in a personalized one, making the elements tied to C_i be inherited by C_j (that is anchors, events and attributes). With this purpose, a special type of structural link (called generalization link) is set between them and, consequently, A, L and al or A^P, L^P and al^P are updated.
aggregateContent	A, L, al --- A^P, L^P, al^P	A relation of aggregation between contents C_i and C_j is established in the basic hyperdocument or in a personalized one, making C_j to be referred to by means of the identifier of C_i. With this purpose, a special type of structural link (called aggregation link) is set between them and, therefore, A, L and al or A^P, L^P and al^P are updated.
decomposeContent	A, L, al --- A^P, L^P, al^P	The result of a generalization or aggregation is removed, whether from the basic hyperdocument (C) or from a personalized one (C^P). That is, the corresponding generalization or aggregation link is deleted. Since heritage is perfomed in runtime, these operations will have no side effects on the elements tied to the generalized contents.

which identifies the user, a *UserType*, which states if the user is an individual or a group, and a list of user identifiers (*UserList*) whose meanings depend on the *UserType*: It includes the list of groups an individual user belongs to or the list of members of a group.

The user definition can be increased with attributes referring to its characteristics, properties, or other relevant information. Such attributes are defined and their values are modified by means of the appropriate operations concerning attributes (see Sections 3.7 and 3.10).

TABLE 4b
Contents: Static Definition and Dynamic Management (cont.)

createContentVersion	C, A, L, lo, al, el, ac --- C^P, A^P, L^P, lo^P, al^P, el^P	It creates a new version of a content, whether in the basic hyperdocument (C) or in a personalized one (C^P). A copy is created and connected to the original content through a version link. The same components as in "duplicateContent" are updated.
deleteContentVersion	C, A, L, lo, al, el, ac --- C^P, A^P, L^P, lo^P, al^P, el^P	One version of a content is removed, whether in the basic hyperdocument (C) or in a personalized one (C^P). The content is removed as well as the link associating it to the previous and later version, that are now connected by means of a new link version. The same components as in "createContentVersion" are updated.
moveContent	HD^B, HD^P	A content is moved from the basic hyperdocument (HD^B) to a personalized one (HD^P) or vice versa. Intuitively, this operation is equivalent to the following process: the original content is duplicated and it is included into the other hyperdocument, keeping the same identifier. If the original content is being moved from HD^P to HD^B, then it is deleted. Otherwise, the original content remains in HD^B so other users can access it.
addUnits	C, A --- C^P, A^P	It adds new units to the represention space of a content, whether in the basic hyperdocument (C) or in a personalized one (C^P). Adding units can affect the anchors placed in the content and, therefore, A or A^Pmight be modified.
deleteUnits	C, A --- C^P, A^P	It removes a specific unit from the represention space of a content, whether in the basic hyperdocument (C) or in a personalized one (C^P). Deleting units affects the anchors placed in the content and, therefore, A or A^Pmight be modified. The last unit of a content can not be deleted.
getGeneralizedContents	none	It returns the list of the contents that generalize a given content. C, A and L or C^P, A^P and L^P are accessed but not modified.
getAggregatedContents	none	It returns the list of the contents that aggregate a given content. C, A and L or C^P, A^P and L^P are accessed but not modified.
getContentComponents	none	It returns the list of the contents that are generalized by or aggregated in a given content. C, A and L or C^P, A^P and L^P are accessed but not modified.

The dynamic model includes operations to create, modify, and delete users as well as to obtain information about their components (see Table 2). Concerning the security procedure, operations compromising the general policy of the application (e.g., creation of users or modification of their abilities) should be allowed only to a special user responsible for the security of the system. This security manager can be implemented as a unique user or as a number of different users. Our recommendation is to keep this security manager within a very restricted group of users that should not include all authors of a hyperdocument since most hypermedia systems are running in high risk environments (e.g., web servers) permanently threatened by malicious users. As exposed in [37], the responsibility of managing a security policy can only be put at the user's side when few and not drastic attacks are expected.

With the inclusion of the set U in the model, the three following issues can be addressed during the design stage:

- A users' structure can be specified defining groups of users that collaborate in a particular information domain.
- A security policy can be formally specified by means of the access function ac (see Section 3.12).
- Designers can define private hyperdocuments which belong to particular users, whether group or individual users. Personalized Hyperdocuments belonging to a group of users can be modified by all the members of the group since access control processes are not applied in private views.

Since users can belong to more than one group, several choices can be offered at runtime, including the ability to decide whether to work as an individual user or as member of a particular group [41].

3.3 The Set of Nodes (N)

A node is an abstract information holder that can contain any type and number of information items. The node represents the smallest information retrieval unit of the

TABLE 5
Anchors: Static Definition and Dynamic Management

$A = \{A_i \mid i = 0,...n, n \in N \text{ where } A_i = (AnchorId_i, NodeId_i, ContentId_i, AnchorPos_i)\}$
$AnchorId_i \in \{\text{alphanumeric strings}\}$
$NodeId_i = -1 \cup \{NodeId_j \mid \exists N_j \in N, NodeId_j \in N_j, j = 0,1,...,n, n \in N\}$
$ContentId_i = -1 \cup \{ContentId_j \mid \exists C_j \in C, ContentId_j \in C_j, j = 0,1,...,n, n \in N\}$
$AnchorPos_i = \{Position_i, Extension_i\}$
$Position_i$ = Location of the initial point of the anchor in the node or content
$Extension_i$ = Extension of the anchor into the node or content

Operation name	Updates	Description of the operation
createAnchor	A --- A^P	A new anchor is added, whether in the basic hyperdocument (A) or in a personalized one (A^P).
deleteAnchor	A, L, al, el --- A^P, L^P, al^P, el^P	An existing anchor is removed from the basic hyperdocument (A) or from a personalized one (A^P). If it is the unique source or target of a link, then the link is also removed and, therefore, this operation may also modify L, al and el or L^P, al^P and el^P.
modifyAnchor	A --- A^P	It modifies the values of $NodeId_i$ and/or $ContentId_i$ of an anchor definition, whether in the basic hyperdocument (A) or in a personalized one (A^P).
modifyAnchorPosition	A --- A^P	It modifies the position and/or extension of an anchor, whether in the basic hyperdocument (A) or in a personalized one (A^P).
getLinksAnchor	none	It returns the list of the links associated to an anchor. L or L^P is accessed but not modified.

hyperdocument from the user's perspective (e.g., a page of an electronic book) since contents can not be retrieved without being presented in a node. Each node should be self-aimed, that is, it has to use as many media as needed to wholly present a particular concept, topic, problem, or question to be discussed.

From a static point of view (see Tables 3a and 3b), a node (N_i) has a unique identifier ($NodeId$), which, in the system implementation, can be resolved to an internal value automatically assigned by a centralized system or to a URI (Uniform Resource Identifier) referring to external resources of a distributed environment. Each node has a security category ($NodeCategory$) that establishes the most permissive type of operation that both the node and its components (links, attributes, and events) can undergo. The $NodeCategory$ has to be set to one of three possible values, each one enlarging the privileges of the previous one: "browsing," "personalizing," or "editing," which permit visiting, adding personalizations, and updating the Basic Hyperdocument, respectively.

Apart from information holders, nodes can act as composites that, combined with typed links, are used to represent two structural relationships: aggregation and generalization. When a composite generalizes/aggregates a number of nodes, a Generalization/Aggregation link is established from the composite to its components. Thus, composites and typed links provide a formal way to define the hyperdocument structure.

Contents and anchors can be placed into nodes at a precise moment and/or location (see Sections 3.9 and 3.5, respectively). Moreover, any number of attributes that enrich the node definition can be assigned to a node by means of the al function (see Section 3.10). In particular, nodes have two mandatory attributes automatically assigned whenever a node is created: "Label," that adds meaningful information to the node description and, "Author," that holds the user who created the node. Finally, the dynamic characteristics of a node can be specified by means of events tied to it through the el function (see Section 3.11).

The set of Nodes is managed by means of several operations related to the composition and management of nodes as well as with the definition of aggregations and generalizations. Versions can be defined and nodes, either simple or composite, can be transferred from a personalized hyperdocument to the basic one and vice versa (see Tables 3a and 3b).

3.4 The Set of Contents (C)

A content is an information item (e.g., a text, a graphic, or an animation) that can be placed in different nodes. According to its static definition (see Tables 4a and 4b), a content (C_i) consists of an identifier ($ContentId_i$), a security category ($ContentCategory_i$), used as in the nodes, and a type ($ContentType_i$) which defines the kind of information represented ($Type_i$) and the different units that make up the content's representation space ($Units_i$). The content identifier can be implemented in the same way as the node identifier to refer to both external and local resources.

TABLE 6
Links: Static Definition and Dynamic Management

$L = \{L_i \mid i=0,...,n, n \in N$ where $L_i = (LinkId_i, LinkStart_i, LinkTarget_i)\}$ $LinkId_i \in$ {alphanumeric strings} $LinkStart_i = \{AnchorId_{ij} \mid \exists A_j \in A, AnchorId_{ij} \in A_j, j = 0,1,...,n, n \in N\}$ $LinkTarget_i = \{AnchorId_{ij} \mid \exists A_j \in A, AnchorId_{ij} \in A_j, j = 0,...,q, q \in N\}$		
Operation name	**Updates**	**Description of the operation**
createLink	L, al --- L^P, al^P	A new link is added, whether to the basic hyperdocument (L) or to a personalized one (L^P). To set the value of the mandatory attributes, al or al^P is modified.
deleteLink	A, L, al, el --- A^P, L^P, al^P, el^P	An existing link is removed, whether from the basic hyperdocument (L) or from a personalized one (L^P), as well as its anchors provided that no other components are associated to them. The list of events and attributes must also be updated. Therefore, A, al and el or A^P, al^P and el^P are modified.
activateLink	none	It activates a link. A or A^P is accessed but not modified.
getLinkTargets	none	It returns the list of targets of a given link. A or A^P is accessed but not modified.
getLinkSources	none	It returns the list of sources of a given link. A or A^P is accessed but not modified.
getILinksTo	none	It returns the list of links whose target is a specific node or content. A and L or A^P and L^P are accessed but not modified. If the element is a generalized node or content, then a recursive process retrieves the inherited links.
getLinksFrom	none	It returns the list of links whose source is a specific node or content. A and L or A^P and L^P are accessed but not modified. If the element is a generalized node or content, then a recursive process retrieves the inherited links.

TABLE 7
Attributes: Static Definition and Dynamic Management

$B = \{B_i \mid i=0,...,n, n \in N$ where $B_i = (AttributeName_i, Value_i)\}$ $AttributeName_i \in$ {alphanumeric strings} $Value_i \in$ {alphanumeric strings}		
Operation name	**Updates**	**Description of the operation**
createAttribute	B --- B^P	A new attribute is added, whether to the basic hyperdocument (B) or to a personalized one (B^P).
deleteAttribute	B, al --- B^P, al^P	An existing attribute is removed, whether from the basic hyperdocument (B) or from a personalized one (B^P), and the attribute list (al or al^P) of the components associated to this attribute are also modified.
modifyDefaultValue	B --- B^P	The default value of an existing attribute is modified, whether in the basic hyperdocument (B) or in a personalized one (B^P).
getAttributeList	none	It returns the list of the attributes of a Hyperdocument. B or B^P is accesed but not modified.

The same as nodes, contents can also be composites and they have associated some attributes and events. Mandatory attributes of contents are the same as those of nodes.

Operations concerning this set allow contents to be added, removed, duplicated, and moved from one hyperdocument to another (see Tables 4a and 4b).

3.5 The Set of Anchors (A)

An anchor is an interval of coordinates of a representation space that is used as a starting or ending point of links (e.g., a hotword in a text, and a hotspot in an image). Thus, anchors can be embedded into nodes and contents and, if the anchor interval is not specified, it refers to the whole

TABLE 8
Events: Static Definition and Dynamic Management

ac: $\forall U_i \in U, \forall x \in (N \cup C)$,ac($U_i$, x) = AccessCategory$_i$ AccessCategory$_i$ = {0, browsing, personalizing, editing}		
Operation name	**Updates**	**Description of the operation**
setUserCategory	ac	It sets the security category of a user for a specific node, content or subhyperdocument. Therefore, it modifies the ac function. It can also be used to deny the access by assigning the zero value.
getUserCategory	none	It returns the security category of a user for a specific node, content or subhyperdocument. The ac function is accesed but not modified.
getUsersNode	none	It returns the list of the users that can access a node. The ac function is accesed but not modified.
getNodesUser	none	It returns the list of the nodes that can be accessed by a user. The ac function is accesed but not modified.
getUsersContent	none	It returns the list of the users that can access a content. The ac function is accesed but not modified.
getContentsUser	none	It returns the list of the contents that can be accessed by a user. The ac function is accesed but not modified.
getUsersDomain	none	It returns the list of the users that can access a subhyperdocument. The ac function is accesed but not modified.
getDomainUser	none	It returns the list of the domains that can be accessed by a user. The ac function is accesed but not modified.

component. Anchors were introduced in classical hypertext systems such as Neptune [6] and Intermedia [42].

From a static point of view (see Table 5), an anchor (A_i) has an identifier ($AnchorId_i$), references to the elements where it is embedded ($NodeId_i$ and $ContentId_i$), and an $AnchorPos_i$, which defines its initial location ($Position_i$) and extension ($Extension_i$).

Anchors can be embedded into nodes and contents in three different ways:

a. An anchor can be tied to a content and only be active when the content is presented in a particular node by assigning a value different from -1 to both $ContentId_i$ and $NodeId_i$.
b. An anchor can be tied to a content and be active in every node where the content is placed by assigning -1 to $NodeId_i$.
c. An anchor can be tied to a node by setting $ContentId_i$ to -1.

In cases a and b, $AnchorPos_i$ is expressed using the different units that make up the representation space of the involved content. Thus, an anchor can be embedded into any place and any type of content. If the anchor is tied to a node, case c, $AnchorPos_i$ refers to a temporal and/or spatial locus or interval (e.g., the right area of a node can be used to navigate to the next node whereas the left one can link to the previous node).

Some operations describe the dynamic management of the link sources and targets (see Table 5). They include creation, deletion, and one access operation that retrieves the links which contain a particular anchor. In agreement to [37], anchors hold the highest security category of the objects involved in their definition (i.e., the node and/or content where they are embedded). Therefore, to determine whether a user is allowed to carry out an operation or not, the security check analyzes the user access category for the nodes and/or contents included in the definition of the anchor. If the user is granted access to a node, only those contents for which he/she has at least a browsing category will be delivered. Thus, different users can have distinct views of the same node depending on their clearances.

3.6 The Set of Links (L)

A link defines a relationship between two sets of anchors: the starting and ending points of a connection that can have navigational or structural purposes. All the links of the application are included in this set, whose formal definition is shown in Table 6. Each link (L_i) consists of an identifier ($LinkId_i$) and two lists that contain the start ($LinkStart_i$) and end anchors ($LinkTarget_i$), respectively. Consequently, n-ary links can be defined. Moreover, since both the source and the target can be procedurally defined assigning an event to the link (see Section 3.11), virtual links (including dangling, warm, and hot links) can also be modeled.

Links can have associated events and attributes. Mandatory attributes include those of the nodes and two additional ones: "LinkType" and "Direction." When a new link is created, the attribute, called "LinkType," is assigned to it and its value is used to discriminate among three semantically different types of links: referential links, that define navigation paths; aggregation and generalization links, that establish a hierarchical relation between composites and their components; and, version links, that sequentially connect different versions of a node or a content. Likewise, the "Direction" attribute is associated to each new link and it is used to establish whether the link is unidirectional or bidirectional; the latter is included in hypermedia systems like Intermedia and Concordia [43].

TABLE 9
Location Function: Static Definition and Dynamic Management

lo: $lo(C_i, N_j) = \{Position, Time\}$, $C_i \in C, N_j \in N \mid i, j = 0,...,n, n \in N$ Position: takes values in node spatial dimension units Position=-1, if no space component is defined Time = {StartTime, Duration} StartTime and Duration take values in time units StartTime=-1, if no time component is defined		
Operation name	**Updates**	**Description of the operation**
placeContentNode	lo --- lo^P	A content is placed into a particular node, whether in the basic hyperdocument (lo) or in a personalized one (lo^P).
removeContentNode	lo, A, L, al --- lo^P, A^P, L^P, al^P	A content is removed from a node, whether from the basic hyperdocument (lo) or from a personalized one (lo^P). If there is an anchor tied to the node and to the content, it is also removed.
alignContents	lo --- lo^P	The location function of two contents is compelled to fulfil a particular spatial relation, whether in the basic hyperdocument (lo) or in a personalized one (lo^P).
getSpatialRelation	none	It returns the spatial relation that exists between two contents. The lo or lo^P function is accessed but not modified.
synchroniseContents	lo --- lo^P	The location function of two contents is compelled to fulfil a particular temporal relation, whether in the basic hyperdocument (lo) or in a personalized one (lo^P).
getTemporalRelation	none	It returns the temporal relation that exists between two contents. The lo or lo^P function is accessed but not modified.
changeLocation	lo --- lo^P	A content is moved in the representation space of a node, whether in the basic hyperdocument (lo) or in a personalized one (lo^P).
getLocation	none	It returns the position of a content in a specific node. The lo or lo^P function is accessed but not modified.
getContentsNode	none	It returns the list of the contents of a node. The lo or lo^P function is accessed but not modified.
getNodesContent	none	It returns the list of the nodes to which a content is placed. The lo or lo^P function is accessed but not modified.

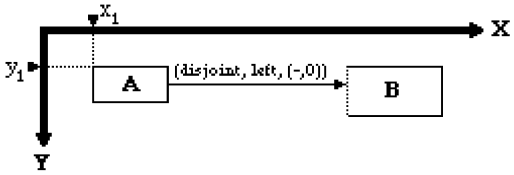
Selecting a link can give rise to a number of reactions, including navigating to a different node, delivering, or hiding a particular content within the same node (e.g., the stretchtext technique, that is in-line expansion of text, introduced in Guide [44]) or opening an external application or program. In the Labyrinth model, the link reaction can be specified associating an event with the appropriate actions to the link (see Section 3.11).

The conceptual and physical separation between the link and the content/node where it is embedded makes the hyperdocument easier to maintain, increasing its openness. This approach has been successfully implemented in open hypermedia systems such as Microcosm [30], [45]. Moreover, since the identifiers of nodes and contents refer to external resources, links point at documents which do not belong to the hyperdocument.

The dynamic management of links (described in Table 6) includes such operations as creation, deletion, duplication, activation, as well as operations to obtain information about the structure of the hyperdocument (for instance, to retrieve the nodes connected by a link).

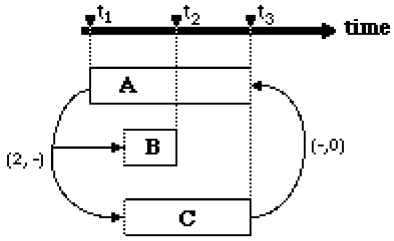
3.7 The Set of Attributes (A)

An attribute is a characteristic that will be used to add semantic information to an element of the application. From a formal point of view (see Table 7), an attribute (B_i) is defined by means of a label ($AttributeName_i$) and a



(disjoint, left, (-,0)) means that:
A and B are always disjoint
A is always on the left of B
The distance between A and B in the vertical axis is always 0

Fig. 1. Aligning contents.



where the synchronization arc between A and B, C means:
start(B) = start(A) + 2 => start(B) = t_1 + 2
start(C) = start(A) + 2 => start(C) = t_1 + 2

and the synchronization arc between C and A means
end(A) = end(C) => end(A)=t_3

Fig. 2. Synchronizing contents.

value ($Value_i$) that will be used by default if no other value is specified during the attributes assignment (see Section 3.10). There are some mandatory attributes, such as "Label," "Author," "LinkType," or "Direction," that always exist in a hyperdocument (see the specification of the operation "createHyperdocument" in Table 13).

The dynamic part of the model includes operations to create, delete, and modify attributes (see Table 7). Moreover, information about the attributes can be retrieved. With respect to the security rules, only users granted an editing category for the basic hyperdocument (that is, for its "hub node") will be able to modify all the attributes. Attributes defined in a personalized hyperdocument can only be modified by the owners of the personalization.

3.8 The Set of Events (E)

A Labyrinth event is formally specified as shown in Table 8. Each event (E_i) includes an identifier ($EventId_i$), a Boolean expression ($Condition_i$) where the conditions determining the event activation are specified, and the list of actions ($ActionList_i$) that have to be performed whenever the event is enabled (that is, when its conditions are fulfilled or when the event is directly triggered from another event).

The list of conditions includes a Boolean expression whose terms can include references to internal values (e.g., time) or external values (e.g., user interactions). The list of actions may contain any operation carried out over a model element and low-level actions such as searching for a string, comparing values, etc.

Events can be tied to nodes, contents, and links by means of the el function with a view to gathering the most dynamic aspects of hyperdocuments.

Operations on the set of Events (see Table 8) include processes related to their maintenance (such as creation, deletion, duplication). The same security rules that apply to attributes are applied for events. Thus, events in the basic hyperdocument can only be modified by users granted an editing category for the basic hyperdocument (that is, for its "hub node") while those defined in a personalized hyperdocument are modified by the owners of the personalization.

3.9 The Location Function (lo)

Contents can be placed into nodes by giving a value to the lo function, whose definition is shown in Table 9. A logical value ($Position_i$) specifies a spatial locus in the node representation space, while another one ($Time_i$) defines the time when it is presented ($StartTime_i$) and the duration of this presentation ($Duration_i$).

TABLE 10
Attributes List Function: Static Definition and Dynamic Management

al: $\forall x \in (U \cup N \cup C \cup L)$, al(x) = {($AttributeName_i$,$Value_i$) | i = 0,...,n, n ∈ N} $AttributeName_i \in B_i$, $B_i \in B$ $Value_i$ ∈ {alphanumeric strings}		
Operation name	**Updates**	**Description of the operation**
assignAttribute	al --- al^P	It assigns an attribute to a user, node, content or link, whether in the basic hyperdocument (al) or in a personalized one (al^P). The attribute can be bound to a specific value.
deassignAttribute	al --- al^P	It deletes the assignment between an attribute and a user, node, content or link, whether in the basic hyperdocument (al) or in a personalized one (al^P).
modifyValue	al --- al^P	It modifies the value of an attribute, whether in the basic hyperdocument (al) or in a personalized one (al^P).
getAttributesList	none	It returns the list of the attributes of a user, node, content or link. al or al^P is accessed but not modified. If the element is a generalized node or content, then a recursive process retrieves the inherited attributes.
getAttributeValue	none	It returns the value of an attribute. al or al^P are accesed but not modified.

TABLE 11
Events List Function: Static Definition and Dynamic Management

el: el(x) = {($EventId_i$, $Priority_i$) | i = 0,...,n, n ∈ N}, x ∈ (N ∪ C ∪ L) $Priority_i$ ∈ {1,...,p} p = maximum number of events		
Operation name	**Updates**	**Description of the operation**
assignEvent	el --- el^P	An event is assigned to a node, content or link, whether in the basic hyperdocument (el) or in a personalized one (el^P). If necessary, a priority value can be specified.
deassignEvent	el --- el^P	It deletes the assignment of an event to a node, content or link, whether in the basic hyperdocument (el) or in a personalized one (el^P).
modifyEventPriority	el --- el^P	The priority value of an event is modified, whether in the basic hyperdocument (el) or in a personalized one (el^P).
getElementEvents	none	It returns the list of the events associated to a node, content or link. el or el^P is accessed but not modified. If it is a generalized node,then a recursive process retrieves the inherited events.
getEventPriority	none	It returns the priority value of an event. el or el^P are accesed but not modified.

The ability to specify the location of contents in different axes and to define a duration can be exploited to create multimedia presentations. The dynamic management of this function (see Table 9) allows contents to be placed into nodes or to be removed from nodes. The latter operation is a deletion of a tuple of the location function and does not imply the deletion of the content from the Hyperdocument. It is also possible to get information about the nodes that include a particular content and vice versa.

There are two crucial operations that allow richer multimedia presentations to be specified: the alignment and the synchronization. Both operations are used to gather spatial and temporal relationships among contents within the scope of a particular node. When such relations are established, they can be compelled to be maintained even if the contents are moved across the representation space of the node. In that case, the alignment or synchronization is represented in the static elements of the hyperdocument by means of an event tied to the node. Since the event is

TABLE 12
Access List Function: Static Definition and Dynamic Management

ac: ∀ U_i ∈ U, ∀ x ∈ (N ∪ C),ac(U_i, x) = $AccessCategory_i$ $AccessCategory_i$ = {0, browsing, personalizing, editing}		
Operation name	**Updates**	**Description of the operation**
setUserCategory	ac	It sets the security category of a user for a specific node, content or subhyperdocument. Therefore, it modifies the ac function. It can also be used to deny the access by assigning the zero value.
getUserCategory	none	It returns the security category of a user for a specific node, content or subhyperdocument. The ac function is accesed but not modified.
getUsersNode	none	It returns the list of the users that can access a node. The ac function is accesed but not modified.
getNodesUser	none	It returns the list of the nodes that can be accessed by a user. The ac function is accesed but not modified.
getUsersContent	none	It returns the list of the users that can access a content. The ac function is accesed but not modified.
getContentsUser	none	It returns the list of the contents that can be accessed by a user. The ac function is accesed but not modified.
getUsersDomain	none	It returns the list of the users that can access a subhyperdocument. The ac function is accesed but not modified.
getDomainUser	none	It returns the list of the domains that can be accessed by a user. The ac function is accesed but not modified.

TABLE 13
Description of the createHyperdocument Operation

$HD = HD^B \approx HD^P$ where
$HD^B = (U, N, C, A, L, B, E, lo, al, el, ac)$
$HD^P = \{HD^P_i \mid i=0,...,n, n \in N$ where
$HDP_i = (UserId, N^p, C^p, A^p, L^p, B^p, E^p, lo^p, al^p, el^p)\}$

Operation name	Updates	Description of the operation
createHyperdocument	HD^B	It creates a basic hyperdocument (HD^B) with the minimum information needed. All sets and functions are empty except for U (that includes one user: the security manager), B (that contains the mandatory attributes) and ac (that includes the security manager ability for the hyperdocument). Personalized Hyperdocuments are created when new users with the adequate security category are added.
deleteHyperdocument	HD	It deletes a basic hyperdocument as well as the personalized hyperdocuments associated to it.

independent from nodes and contents, its activation condition can be increased to represent richer space and time-based relationships. For instance, if we want to synchronize a speech with its written text only if students are at the beginners stage, an event is tied to the corresponding node and its condition states that "the node is activated and the user is at the beginners stage."

Alignments can be defined taking into account at least one of the following three aspects as recommended in [27]:

- The topological relation existing among the contents, which assumes the 4-intersection model presented in [46], where the relationships among spatial regions are defined considering the intersections between their interiors and boundaries. The topological relation can be set to one of the following values: disjoint, touch, covers, contains, equal, overlapDisjoint, and overlapNotDisjoint.
- A directional relation that can be bound to one of the following eight values: up, down, right, left, right-up, left-up, right-down, and left-down.
- The distance between the contents that is specified by means of a pair (x,y) that states a displacement in the two spatial axis of the node.

An alignment is a one-to-many directed arc between contents which represents a particular space-based relation. Each alignment has a label that includes three parameters whose value represents the topology, direction, and distance, respectively, which have to be maintained among the source and the targets of the arc. If no value is specified for any of them, it is assumed that no constraint applies for that aspect of the space-based relation. Several aspects can be bound to a value to express more complex dependencies. For instance, Fig. 1 shows an example of alignment of two contents, A and B, whose position can be changed as far as they always fulfill the three conditions stated in the alignment arc. Dashed lines are locations calculated from the alignment.

A synchronism is a one-to-many directed arc between contents. The synchronization has a label that includes a pair of values which represent the temporal delay which has to be added to the start and/or the end of the source to derive the start and/or the end of the targets. Such values can also be expressed making use of the duration of the content specified in its formal definition. This synchronism model follows the principles of interval-based conceptual models which use temporal intervals to represent the presentation time of multimedia contents [47] and it encompasses the set of temporal relationships presented in two representative models of time-based relationships [48] and [49] and the choices considered in Amsterdam. Fig. 2 illustrates a synchronization among three contents. Dashed lines represent locations calculated from a synchronism.

3.10 The Attribute List Function (al)

Attributes are tied to users, nodes, contents, and links by means of the al function (see formal definition in Table 10), which acts as a repository of properties. The default value specified in the attribute definition (see Table 7) can be changed giving a new value ($Value_i$). Since each element has at least one mandatory attribute ("Label"), all users, nodes, contents, and links defined in the hyperdocument take part in this function. Indeed, whenever a new user, node, content, or link is created, the list of its mandatory attributes is added to this function and if no value is specified for them, the default value given in the attribute definition is assumed.

Attributes constitute a powerful source of information about users, nodes, contents, and links, that can be exploited in different ways. For instance, assigning nodes and links a type can provide the basis for the automatic generation of richer navigation maps that use colors to distinguish among the different elements of the map as in [50]. Another use of attributes can be to hide valuable information for indexing [51] or information retrieval purposes [52], including the runtime management of the hyperdocument elements.

The dynamic part of the model (shown in Table 10) contains operations that permit assigning attributes to nodes, contents, links, and users, as well as modifying their value. Designers are also provided with several access operations which can be used to seek information about the different tuples of this function. With respect to the security rules, each attribute assignment inherits the security category of the element to which it is tied. Indeed, to

TABLE 14
Description of the createNode Function

Inputs provided by the system	userId, scope, personalizedHyperdocument, hubNodeId
Inputs provided by the user	usersList, nodeLabel

Process	IF (getUserCategory(userId, hubNodeId)="editing") OR
	((getUserCategory(userId, hubNodeId)="personalizing") AND (scope="personal"))
	THEN
	nodeIdentifier=newIdentifier()
	newNode=(nodeIdentifier, "editing")
	assignAttribute (nodeIdentifier, Label, nodeLabel)
	assignAttribute (nodeIdentifier, Author, userId)
	IF scope="personal"
	THEN
	add (newNode, N^P)
	ELSE
	add (newNode, N)
	applicationUsers=getUsersList()
	FOR EACH user IN applicationUsers DO
	IF user IS IN usersList
	THEN
	setUserCategory (user, nodeIdentifier, "browsing")
	ELSE
	setUserCategory (user, nodeIdentifier, 0)
	END-IF
	END-FOR
	setUserCategory (userId, nodeIdentifier, "editing")
	END-IF
	END-IF

Output	N_{new} = (nodeIdentifier, "editing") $N = N \cup N_{new}$; al(nodeIdentifier)={(Label, nodeLabel), (Author, userId)} OR $N^P = N^P \cup N_{new}$; al^P(nodeIdentifier)={(Label, nodeLabel), (Author, userId)}

decide whether an attribute assignment can be modified it must be determined if the user is permitted to modify the node, content, or link involved. Attributes tied to a particular user can only be modified by that user.

3.11 The Events List Function (el)

The el function (see Table 11) allows events ($EventId_i$) to be associated to nodes, contents, and links and to define a priority ($Priority_i$) that helps solve events' concurrence.

Events can be used for different purposes including: the modeling of virtual elements (nodes, contents, anchors, links, attributes, events, or their relationships) that are created at runtime, conditional links, whose target depends on some circumstances, and any kind of interactive behavior.

Operations to assign, deassign, and modify the priority of events are provided in the dynamic model (see Table 11). Moreover, there are some access operations that return the events tied to a particular element.

3.12 The Access List Function (ac)

The access list function (see Table 12) assigns access categories ($AccessCategory_i$) to users (U_i) depending on the context (node, content, or domain) where they are working. Consequently, and since a mandatory security policy is assumed, each user must have a specific privilege to manipulate each node and content in the hyperdocument.

The ac function is used to preserve information confidentiality, and integrity, assuming the security model presented in [40].

On the one hand, and in order to guarantee confidentiality, an access matrix model [53] is simulated by assigning the zero value to the users who cannot access a node, content, or domain.

On the other hand, and with a view to preserving information integrity, a specific user clearance can be assigned once access has been pemitted, so that the user manipulation ability will depend on its role in a given context as recommended in [54]. Three clearances are considered (browsing, personalizing, and editing) each one enlarging the privileges of the previous one. Thus, "browsing" users will be able to access the elements (that is, they can retrieve them and interact with them); "personalizing" users will also be able to include personal elements and "editing" users will also be able to modify the Basic

Hyperdocument. If the user is an individual, its clearance prevails to the clearance of the group to which it belongs.

Operations concerning the ac function (see Table 12) are related to the management of users, groups, and abilities to access the hyperdocuments. Only a security manager should be authorized to perform these operations in order to implement a mandatory security policy that will guarantee information integrity. To ease the management of this function, the security privileges of users can be specified for a specific node, content, or domain.

3.13 Examples of the Specification of Operations

In this section, the specification of the dynamic part of Labyrinth is illustrated by means of two examples: the creation of a hyperdocument and the inclusion of a new node.

3.13.1 Creating a Hyperdocument

The first operation to be performed in any application is the creation of a hyperdocument. This task should be carried out by the security manager that will be responsible for generating a hyperdocument with the minimum information needed. In this minimum hyperdocument, no personalizations are considered and, therefore, there is only a basic hyperdocument whose unique user is the security manager. With this purpose, the user will execute the "createHyperdocument" operation whose detailed specification is shown in Table 13, providing a value for her own identifier and a value for her attribute "Label" that could be her name. The following user identifiers will be automatically assigned. This operation creates each set and function and assigns them the appropriate values. For instance, the "hub" node is created and all the mandatory attributes are included in B.

Once the hyperdocument has been created, the security manager ought to include the users of the application using the "createUser" operation (see Table 2). The next step will consist of assigning users the appropriate access category to manipulate the hyperdocument ($hubNodeId$) by means of

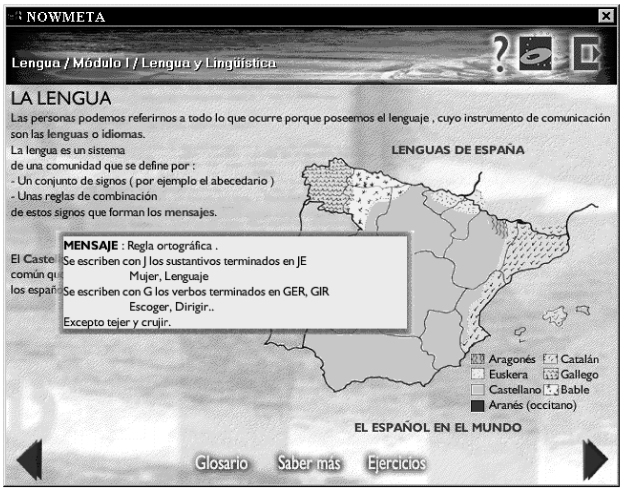


Fig. 3. Screen of META with a glossary link.

the "setUserCategory" operation (see Table 12). Then, users will be able to define the structure of the hyperdocument and include their contents and other elements. Those users granted an "editing" category will be able to create nodes, contents, and other elements and, therefore, they will contribute to the hyperdocument development. Those users having a "personalizing" category will be able to create personalized hyperdocuments that will not be seen by other users. Finally, users assigned a "browsing" category will only be able to read and interact with the hyperdocument.

3.13.2 Including a New Node

To create a new node, whether composite or not, the "createNode" operation can be invoked (see description on Table 14). To create a new node in the basic hyperdocument the user needs to have an "editing" category for the hyperdocument whereas to generate a personalized node a "personalizing" category is required. If the operation is allowed (i.e., security checks succeed), a new node, with an identifier and an appropriate security category, is included

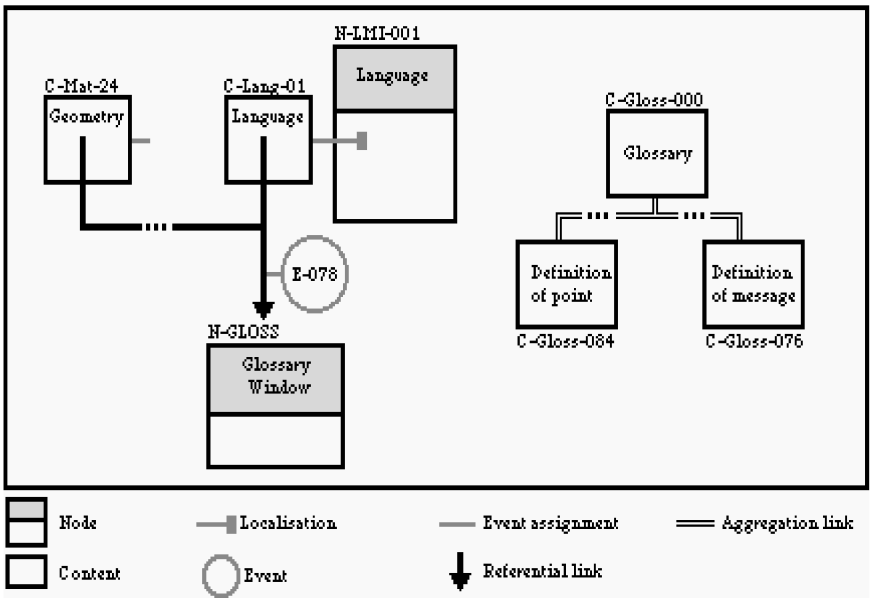


Fig. 4. The graphical representation of the modeling of a glossary link.

TABLE 15
Event Specification for the Glossary Link

```
EventId: E-078
Conditions: selectedLink
Actions:
    key = copyChunk (basicHD, self.LinkStart.ContentId,
          self.LinkStart.AnchorPos.Position, self.LinkStart.AnchorPos.Extension)
    glossaryTerms = getContentComponents (basicHD, C-Gloss-000)
    target = search (key, glossaryTerms)
    if target is not null then
          placeContentNode (basicHD, target.ContentId, N-Gloss, (0,0))
          activateLink (basicHD, self.LinkId)
    endif
```

in the set N or N^P, depending on the value of the parameter "scope" (see Table 14). The node will be assigned its two mandatory attributes: "Label," whose value is explicitly given by the user and "Author," which is bound to the value of the user identifier executing the operation. According to the mandatory security policy assumed in the model to preserve information integrity, the node category can not be decided by its author but rather by the security manager. With this purpose, a default value, which is the most permissive one (i.e., "editing"), is automatically assigned, although the security manager can modify it by means of the "setNodeCategory" operation (see Tables 4a and 4b). Since there is also a discretionary access model intended to preserve information confidentiality, authors can restrict the access to the nodes they create. For this reason, the user has to specify those users ($userList$) that will not be excluded from the access list of the node. Initially, they will be assigned the most restrictive category (that is, "browsing") and it can be latter changed by the security manager by means of the "setUserCategory" operation (see Table 12). Although this approach might seem too restrictive, the severity of the expected risks in a distributed environment require mandatory policies which translate the security responsibilities from the user to a special manager. Users whose identifiers are not included in $userList$ of a node are not allowed even to browse the node. This $userList$ can be modified by adding and deleting users through the "setUserCategory" operation. The same security rules are assumed for the contents. Finally, the author of the node is assigned the most permissive category, that is, "editing."

4 Using the Labyrinth Model to Specify Dynamic Behaviors

The model can be used as a design tool since its components allow structures and behaviors to be unambiguously specified for any hypermedia application. With this purpose and, in order to increase the usability of the model, a graphical representation of its components can be used. In this section, we present two examples of this kind of usage, both of which are based on the META educational application [55].

META is a hypermedia system aimed at helping low-qualified women study the most difficult subjects of the primary school. The system contents are mainly centered on social and natural science subjects which are used to introduce linguistic and mathematical concepts. Contents are structured as a set of modules made up of lessons. Each lesson includes several nodes with theoretical explanations, visual examples, and interactive exercises that provide students with self-evaluation mechanisms. Hypermedia links are defined among concepts and explanations, providing a quick access to related information. META has been developed as a CD, working under the Windows 95 operating system.

The two examples have been selected because their modeling involves intrinsic features of hypermedia applications and they illustrate the suitability of Labyrinth to model the dynamic behavior of this kind of applications. Thus, the first one is concerned with the specification of virtual links (that is, links whose source or target is dynamically calculated), whereas the second one consists of the use of dynamic management of events. Both of them represent quite complex examples of modeling, but since they were used many times during the implementation, the effort required to develop them was worthwhile.

Example 1 (Modeling links to the glossary entries). Some terms of the textual contents included in the lessons are linked to a glossary, where definitions for difficult concepts are provided. Fig. 3 shows an example of a glossary link included in the lesson "La lengua" (Language), where the main characteristics of a language and the different languages spoken in Spain are

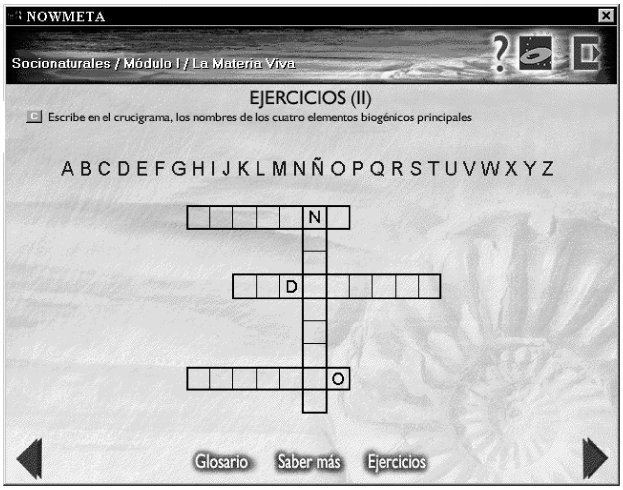


Fig. 5. The crossword exercise.

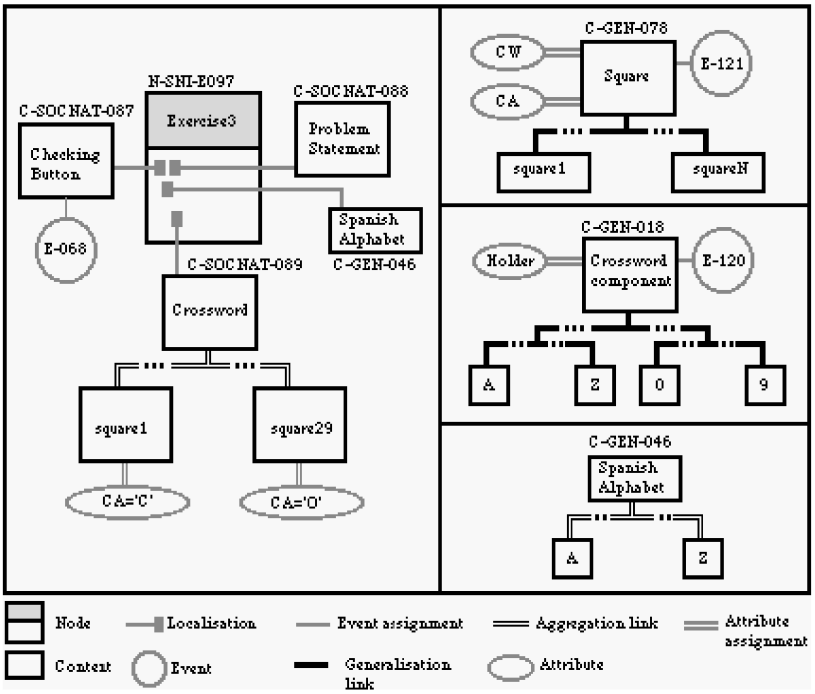


Fig. 6. The crossword modeling.

presented. The word "mensaje" (message) included in the definition of language is linked to the glossary, whose "mensaje" entry is shown in a pop-up window.

Links to the glossary can be modeled in two different ways:

- A specific link is defined from each term to its entry in the glossary and, therefore, as many links as glossary entries are maintained.
- A unique virtual link is established, whose target is calculated taking into account the content where the source is embedded.

The modeling of the latter solution using Labyrinth, which eases the resulting design and decreases the number of links to be maintained, is graphically shown in Fig. 4. The node (N-LMI-001) represents the screen shown in Fig. 3, and the localization function is used to place the content concerning the language definition (C-Lang-01), which includes a link to the glossary. This link is defined as an n-ary link among all the contents including a reference to the glossary and the "Glossary window" node (see the link anchor in C-Mat-24 that connects a geometry description with the definition of "point"). When selected, the specification of the source anchor of the link is used to put the appropriate content into the "Glossary window." In order to optimize the searching process, all entries of the glossary are aggregated by means of a composite content (C-Gloss-000) so that they can be referred to by a unique identifier.

Thus, the target of the glossary link is virtually specified tying an event (E-078) to the link, whose formal definition is presented in Table 15.

Example 2 (The crossword exercise). The second example, shown in Fig. 5, consists of an interactive exercise that asks students to fill a crossword with the four main biogenic elements ("Carbono," "Nitrógeno," "Oxígeno," and "Hidrógeno"). With this purpose, women are presented a crossword structure and a sequence of characters that have to be placed into the squares using the mouse. When a character is selected, it follows the cursor movement till it is dropped into a square. Students can remove a character from the crossword by clicking on it and they can check the exercise correctness selecting a checking button (the one placed before the exercise statement). This exercise represents a good example of interactivity, one of the basic pillars of multimedia.

The exercise (see Fig. 6) is modeled by means of a node (N-SNI-E097) where four contents are placed:

- the checking button (C-SOCNAT-087), that allows the exercise to be checked,
- the problem statement (C-SOCNAT-088), which asks students to solve the exercise,
- the Spanish alphabet (C-GEN-046), that can be used by students to write, and
- the crossword (C-SOCNAT-089) where the biogenic elements have to be written.

To generate a cleaner and clearer design, several composite contents gather the logical characteristics of this type of exercise. First, an aggregation is used to collect all the crossword squares involved in this particular exercise so that they can be referred to by means of a unique identifier: C-SOCNAT-089. Second, three composites are used to gather the crossword behavior:

- A generalization content (C-GEN-078) gathers the commonalties of all squares and is used to decide if a student guesses which character is hidden

TABLE 16
Events Specification for the Crossword Exercise

EventId: E-068
Conditions: mouseUp
Actions:
```
squaresList=getAggregatedContents (basicHD, C-SOCNAT-089)
equal=TRUE
FOR each item in squaresList DO
    equal= equal AND (getAttributeValue (basicHD, item.ContentId, "CW") =
        getAttributeValue (basicHD, item.ContentId, "CA"))
END-FOR
IF equal THEN
    changeLocation (basicHD, correcto, currentNode.NodeId, (a,b))
ELSE
    changeLocation (basicHD, incorrecto, currentNode.NodeId, (a,b))
END-IF
```

EventId: E-120
Conditions: mouseUp
Actions:
```
duplicateContent (basicHD, self.ContentId, newChar)
deassignEvent (basicHD, newChar.ContentId, E-120)
assignEvent (basicHD, newChar.ContentId, E-190)
```

EventId: E-121
Conditions: objectDropped
Actions:
```
deassignEvent (basicHD, currentDropped.ContentId, E-190)
modifyValue(basicHD, self.ContentId, "CW",
                retrieveContent(currentDropped.ContentId))
assignEvent (basicHD, currentDropped.ContentId, E-191)
modifyValue (basicHD, currentDropped.ContentId, "Holder", self.contentId)
```

EventId: E-190
Conditions: idle
Actions:
```
changeLocation (basicHD, self.ContentId, currentNode.NodeId, (mousePosition, -1))
```

EventId: E-191
Conditions: mouseUp
Actions:
```
deleteContent (basicHD, self.ContentId)
modifyValue(basicHD, getAttributeValue(basicHD, self.ContentId,"Holder"), "CW", "")
```

under the square. With this purpose, C-GEN-078 includes two attributes: the correct character assigned to the square ("CA") and the character actually written by the student ("CW") which is updated runtime. This composite has also an event (E-121) that updates the "CW" attribute when a character has been dropped into the square.

- A composite content (C-GEN-018) generalizes the elements (characters and digits) that can be put into the crossword squares and it has associated one event (E-120), that formalizes the process of writing into the crossword and one attribute ("Holder") that is used to directly refer to the square where the character is written.
- A composite content (C-GEN-046) aggregates the characters making up the Spanish Alphabet so that they can be placed anywhere by using a unique identifier.

Since these three composites formalize a general behavior for a crossword that does not depend on the words involved, they can be reused in similar exercises.

The behavior of any crossword is modeled through the events presented in Table 16. The crossword has two basic elements: a sequence of characters and one of squares. Each character in the sequence has associated an event (E-120) that creates a copy of the character to be written in a square. With this purpose, another event (E-190), that is tied in runtime to the new character, assigns the position of the new character to the position of the cursor. Once the new character has been dropped into a square, another event (E-121) deassigns E-190 from the character (so that its position cannot be modified), updates the attribute "CW" (character written) of the square, and ties an event (E-191) to the character in order to allow its deletion. The checking process of the exercise is modeled through an event (E-068) tied to the checking button.

This event will present one of two messages: "correcto" if the answer provided by the student is correct and "incorrecto" in other case.

5 Conclusions

There is an increasing number of models aimed at gathering hypermedia semantics. Hypertext-oriented models (including [6], [7], [8], [9], [10], [13]) are inadvisable for the specification hypermedia applications since they do not provide mechanisms to gather both the special features of multimedia components (e.g., the existence of temporal and spatial relationships among contents) and their interactive behaviors. Hypermedia-oriented models (including Trellis, Dexter, Amsterdam, OOHDM, or extended RMM) do not properly face all the problems related to the conceptual design of this kind of application. In particular, the absence of concrete mechanisms to represent interaction capabilities, multimedia presentations, and security policies that will ensure information integrity and confidentiality represent lacks in these models.

In this paper, we have presented the Labyrinth model and, in particular, we have focused on the dynamic management and behavior of applications that can be formally defined by means of the events and the complete set of operations making up the dynamic part of the model. Events, as sets of actions triggered under particular conditions, provide the basis for the specification of behaviors that can go beyond browsing capabilities of hypermedia applications, as shown in the examples presented in the previous section (definition of virtual links and dynamic management of events and attributes). Moreover, other relevant features of Labyrinth are the following:

- It provides elements to specify the static structure of hyperdocuments including composition mechanisms that allow defining and managing generalizations (and the reverse relation, particularization) and aggregations that can be applied to nodes and contents. Structures can be procedurally specified using events.
- Representation spaces offer a mechanism for the definition of several coordinates for each multimedia content and, therefore, the more "natural" and appropriate measurement units can be specified for each type of content. Thus, anchors can be placed into any position of any content by only specifying an initial position and an extension, both of them expressed in terms of the representation space of the involved content.
- Synchronizations and space-based relationships (called alignments in Labyrinth) can be used to compose richer multimedia presentations.
- Users and groups of users can work in their own private views of the hyperdocument, called Personalized Hyperdocuments.
- The inclusion of users and the definition of access categories both provide a formal basis for the specification of a mandatory security policy during the design stage.

There are some aspects of implementation that are not covered by the model, such as the specification of the presentation characteristics of multimedia elements or the inclusion of a scripting language.

The model is supported by a design methodology [56] that guides the designers from their particular conception of the problem to a logical solution expressed in a formal specification that can be understood by the programmers or even used for automatic generation of code. The methodology consists of two phases: the Conceptual Design, aimed at producing a logical skeleton of the application in terms of structure, processes and behaviors, navigation capabilities, and presentation features; the Detailed Design, where the logical skeleton is instanced to completely specify the hyperdocument.

Acknowledgments

The authors would like to thank the three anonymous referees and Dr. Carl E. Landwher for their very helpful comments.

References

[1] N. Benamou and A. Celentano, "Production or Interactive Multimedia Courseware with Mathesis," *Prod. Design and Production of Multimedia and Simulation-Based Learning Material*. pp. 61-82, 1994.

[2] S. Bagui, "Reasons for Increased Learning Using Multimedia," *J. Educational Multimedia and Hypermedia*, vol. 7, no. 1, pp. 3-18, 1998.

[3] P. Díaz, I. Aedo, N. Torra, P. Miranda, and M. Martín, "Meeting the Needs of Teachers and Students within the CESAR Training System," *British J. Educational Technology*, vol 29, no. 1, pp. 35-46, 1998.

[4] M. Fuller, R. Sacks-Davis, R. Wilkinson, and J. Zobel, "Hyperbase Systems: A Structured Architecture," *Proc. Second Far East Workshop Future Database Systems*, pp. 222-230, Apr. 1992.

[5] F. Garzotto, P. Paolini, and D. Schwabe, "HDM—A Model-Based Approach to Hypertext Application Design," *ACM Trans. Information Systems*, vol. 11, no. 1, Jan. 1993.

[6] N.M. Delisle and M.D. Schwartz, "Neptune: A Hipertext System for CAD Applications," *Proc. Int'l Conf. Management of Data*, pp. 132-143, May 1986.

[7] F. Tompa, "A Data Model for Flexible Hypertext Database Systems," *ACM Trans. Information Systems*, vol. 7, no. 1, pp. 85-100, 1989.

[8] P.D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. Office Information Systems*, vol. 7, no. 1, pp. 3-29, 1989.

[9] F.G. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model," *Proc. World Conf. Hypertext*, pp. 95-133, 1990.

[10] D.B. Lange, "A Formal Model of Hypertext," *Proc. Hypertext Standardization Workshop*, pp. 145-166, 1990.

[11] P.D. de Bra, G. Houben, and Y. Kornatzky, "An Extensible Data Model for Hyperdocuments," *Proc. of the ACM Conf. Hypertext*, pp. 222-231, 1992.

[12] L. Hardman, D. Bulterman, and G. Van Rossum, "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model," *Comm. ACM*, vol. 37, no. 2, pp. 50-62, 1994.

[13] B. Wang and P. Hitchcock, "InterSect_DM: A Hypertext Data Model Based on OODBMS," *Information and Software Technology*, vol. 37, no. 3, pp. 177-190, 1995.

[14] P. Díaz, I. Aedo, and F. Panetsos, "Labyrinth, An Abstract Model for Hypermedia Applications. Description of Its Static Components," *Information Systems*, vol. 22, no. 8, pp. 447-464, 1997.

[15] D. Schwabe and D. Rossi, "Developing Hypermedia Applications Using OOHDM," *Proc. HT98 Workshop Hypermedia Development Processes, Methods, and Models*, 1998.

[16] T. Isakowitz, "Structured Design of WWW and Intranet Applications," *CRIS Working Paper Series, Information Systems Dept. Stern School of Business. New York Univ.*, 1998.

[17] F. Garzotto, L. Mainetti, and P. Paolini, "Adding Multimedia Collections to the Dexter Model," *Proc. European Conf. Hypertext*, pp. 70-80, 1994.

[18] P. Kommers, A. Ferreira, and A. Kwak, *Document Management for Hypermedia Design.* Springer-Verlag, 1998.
[19] *Metafile for Interactive Documents.* Available in www.nawcsti.navy.mil/mid/mid.html.
[20] F. Tompa, "A Data Model for Flexible Hypertext Database Systems," *ACM Trans. Information Systems,* vol. 7, no. 1, pp. 85-100, 1989.
[21] P.K. Garg, "Abstraction Mechanisms in Hypertext," *Comm. ACM,* vol. 31, no. 7, pp. 862-870, July 1988.
[22] G. Richard and A. Rizk, "Quelques Idées pour une Modélisation des Systèmes Hypertextes," *Technique et Science Informatiques,* vol. 9, no. 6, pp. 505-514, 1990.
[23] P.J. Nürnberg and J.J. Leggett, "A Vision for Open Hypermedia Systems," *J. Digital Information,* vol. 1, no. 2, 1997, http://ojfpc.ecs.soton.ac.uk.
[24] W. Mahdi, L. Chen, and D. Fontaine, "Improving the Spatial-Temporal Clue Based Segmentation by the Use of Rhythm," *Research and Advanced Technology for Digital Libraries,* pp. 169-181, 1998.
[25] ISO 13522-1 Information Technology-Coding of Multimedia and Hypermedia Information-Part I: MHEG Object Representation Base Notation.
[26] A. Dix and G. Abowd, "Modelling Status and Event Behaviour of Interactive Systems," *Software Eng. J.,* pp. 334-346, Nov. 1996.
[27] M. Varziginannis and S. Boll, "Events in Interactive Multimedia Applications Modeling and Implementation Design," *Proc. Int'l Conf. Multimedia Computing and Systems, ICMCS '97,* pp. 244-251, June 1997.
[28] K.U. Wiil and J.J. Leggett, "The HyperDisco Approach to Open Hypermedia Systems," *Proc. Hypertext 96,* pp. 140-148, 1994.
[29] F.G. Halasz, "Reflection on NoteCards: Seven Issues for The Next Generation of Hypermedia Systems," *Comm. ACM,* vol. 31, no. 7, pp. 836-852, 1988.
[30] A.M. Fountain, W. Hall, I. Heath, and C. Davis, "MICROCOSM: An Open Model for Hypermedia With Dynamic Linking," *Proc. European Conf. Hypertext. Hypertext: Concepts, Systems and Applications,* A. Rizk, N. Streitz, and J. Andre, eds. pp. 298-311, Nov. 1990.
[31] N. Meyrowitz, "The Link to Tomorrow," *Unix Rev.,* vol. 8, no. 2, pp. 58-67, 1990.
[32] R. Rada, *Interactive Media.* Springer-Verlag, 1995.
[33] R. Gonzalez, "Hypermedia Data Modeling, Coding and Semiotics," *Proc. IEEE,* vol. 85, no. 7, pp. 1111-1140, 1997.
[34] F. Garzotto, L. Mainetti, and P. Paolini, "Hypermedia Application Design: A Structured Approach," *Designing User Interfaces for Hypermedia,* W. Shuler, J. Hanneman, and N. Streitz, eds., pp. 5-17, 1995.
[35] N.M. Delisle and M.D. Schwartz, "Contexts—A Partitioning Concept for Hypertext," *ACM Trans. Office Information Systems,* vol. 5, no. 2, pp. 168-186, 1987.
[36] P. Samarati, E. Bertino, and S. Jajodia, "An Authorization Model for a Distributed Hypertext System," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 4, pp. 555-562, July/Aug. 1996.
[37] D. Merkl and G. Pernul, "Security for Next Generation Hypertext Systems," *Hypermedia,* vol. 6, no. 1, pp. 1-19, 1994.
[38] B. Thuraisingham, "Multilevel Security for Information Retrieval Systems-II," *Information and Management 28,* pp. 49-61, 1995.
[39] C.F. Goldfarb, "HyTime: A Standard for Structured Hypermedia Interchange," *Computer,* vol. 24, no. 8, pp. 81-84, Aug. 1991.
[40] P. Díaz, I. Aedo, F. Panetsos, and A. Ribagorda, "A Security Model for the Design of Hypermedia Systems," *Proc. 14th Information Security Conf. SEC '98.* pp. 251-260, 1998.
[41] T.F. Lunt, "Security in Database Systems," *Computers & Security,* vol. 11, pp. 41-56, 1992.
[42] N. Meyrowitz, "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework," *Proc. Conf. Object-Oriented Programming Systems, Languages and Applications (OOPSLA' 86),* pp. 186-201, 1986.
[43] J.H. Walker, "Supporting Document Development with Concordia," *Computer,* vol. 21, no. 1, pp. 48-59, Jan. 1989.
[44] J. Nielsen, *Multimedia and Hypertext: The Internet and Beyond.* Academic Press, 1995.
[45] G. Hill, W. Hall, D. De Roure, and L. Carr, "Applying Open Hypertext Principles to the WWW," *Proc. Hypermedia Design,* 1995.
[46] M. Egenhofer and R. Franzosa, "Point-Set Topological Spatial Relations," *Int'l J. Geographic Information Systems,* vol. 55, no. 2, pp. 160-174, 1991.
[47] K. Hadouda, C. Djeraba, and H. Briand, "Modelling of the Interactive Application in Term of Scenario in a Multimedia Database," *Proc. Eighth Int'l Workshop Database and Expert Systems,* pp. 246-251, 1997.
[48] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. ACM,* vol. 26, no. 11, pp. 832-843, Nov. 1983.
[49] M. Li, Y. Sun, and H. Sheng, "Temporal Relations in Multimedia Systems," *Computer and Graphics,* vol. 21, no. 3, pp. 315-320, 1997.
[50] J.C. Wild, K.K. Maly, C. Zhang, D.E. Eckhardt, C.C. Rorberts, D. Rosca, and T. Taylor, "Project Management Using Hypermedia CASE Tools," *Proc. Int'l Conf. Data and Knowledge Systems for Manufacturing and Eng.,* pp. 722-727, May 1994.
[51] I. Aedo, T. Ayllon, M. Landoni, and F. Panetsos, "SIHEN: A Hypertext-Based Environment for Automatic Creation of Encyclopaedias and Dictionaries," *HyperMedia,* vol. 6, no. 2, pp. 111-123, 1994.
[52] P. Gloor, *Elements of Hypermedia Design: Techniques for Navigation and Visualization in Cyberspace.* Boston, Mass.: Birkhäuser, 1997.
[53] G.S. Graham and P. Denning, "Protection-Principles and Practice," *Proc. Spring Join Comp. Conf., 40,* pp. 417-429, 1972.
[54] D.B. Lange, K. Østerbye, and H. Schütt, "Hypermedia Storage," 1992, Available through WWW ftp://ftp.iesd.auc.dk/pub/reports/techreports/R92-2009.ps.Z
[55] I. Aedo, P. Díaz, F. Panetsos, M. Carmona, S. Ortega, and E. Huete, "A Hypermedia Tool for Teaching Primary School Concepts to Adults," *Proc. IFIP WG 3.3 Working Conf. Human Computer Interaction and Educational Tools,* pp. 180-188, 1997.
[56] P. Díaz, I. Aedo, and F. Panetsos, "A Methodological Framework for the Conceptual Design of Hypermedia Systems," *Proc. Fifth Conf. Hypertexts and Hypermedia: Products, Tools, and Methods (H2PTM '99),* pp. 213-228, Sept. 1999.

Paloma Díaz received a degree and a doctorate both in computer science from the Universidad Politcnica de Madrid, Madrid, Spain. Since 1992, she has been working as a teacher in the Universidad Carlos III de Madrid where she is currently an associate professor in the Computer Science Department. She has been mainly researching the software engineering and hypermedia fields, although other areas of interest include tele-education, electronic journalism, information security, and digital libraries. She is a coauthor of several articles and books concerning her research and teaching activities. Dr. Díaz is a member of the IEEE and ACM.

Ignacio Aedo received a degree and a doctorate both in computer science from the Universidad Politcnica de Madrid, Madrid, Spain. Since 1991, he has been working as a teacher in the Universidad Carlos III de Madrid where he is currently an associate professor in the Computer Science Department. He has been mainly researching the use of new technologies in educational environments. He has also worked in several fields including hypermedia models, electronic books, electronic journalism, and digital libraries. He is a coauthor of several articles and books concerning his research and teaching activities. Dr. Aedo is a member of the ACM.

Fivos Panetsos studied applied mathematics at the Pavia Unversity in Italy and carried out a PhD degree in artificial intelligence at the Complutense University and a second PhD degree in neural science at the Autonomous University, both in Madrid, Spain. His interests include information processing in the brain, neural networks and modelling, and the development of hypermedia systems. Presently, he is an assistant professor at the Complutense University of Madrid.