1. Introduction

Hypermedia applications include static media such as text and graphics, as well as active media such as video, animation, and audio, coupled with sophisticated navigation capabilities similar to those found in hypertext applications. Designing hypermedia applications that are significantly larger than a few dozen nodes and links is a complex task, if compared to other, more traditional classes of information systems. Complexity is induced by a number of factors: the non-linear organisation of the material and the need, for the designer, of controlling the potential “explosion” of the number of links; the co-existence of multiple media of substantially different nature, that induce a variety of styles of user interaction; the lack of standardisation, both in the architecture of hypermedia systems, and in the definition of information structures and interaction paradigms that should be provided in hypermedia applications.

In this paper, we will analyse the process of hypermedia application design, and we will propose some guidelines to address the design task in an organised way. According to our approach, the design of a hypermedia application can be organised in four different (but interdependent) tasks: content analysis, structure design, dynamic design, and lay-out design [7]. Each of these tasks focuses on different aspects of a hypermedia application and requires different methods, description terminology, and expertise. In this paper, we will also propose a set of primitives that can be useful for describing the output of the design process and are based on a well-known design model called HDM [1, 3, 4, 5].

Our approach will be exemplified by discussing the design of a hypermedia in the law domain. This case study application is named CLICK and is commercially available on a CD-ROM. It is produced and distributed by an Italian publisher specialised in economic and legal publications [2].
2. HYPERMEDIA CONTENT ANALYSIS

Content analysis is the task of selecting and producing the actual information that must be included in the hypermedia application. The content can be represented by static, passive media such as formatted data, text, image, or by dynamic, active media such as video clips, sound tracks, animation, etc.

This design phase requires one or more specialists in the application domain, supported by an application analysts, in order to define the user requirements and the intended goals of an application. In this design phase, it must also be decided if the material must be created *ad hoc* for the specific application, or if pre-existing material can be reused.

Content analysis results in the specification of the knowledge requirements of the application and in the identification of the source material and of the methods to produce it.

The organisation of the content and the specification of dynamics and lay-out features of a hypermedia application are concerns of the design phases, discussed in the following sections.

Example

Our case study “CLICK” [2] concerns Italian fiscal regulation, and is mainly intended for professionals such as fiscal specialists or lawyers. The CLICK CD-ROM stores the laws regarding Italian fiscal regulation from 1924 to 19941, and their relationships to the related “practice” (i.e., law enforcement) and case law. This material is integrated with comments and arguments that discuss legal interpretation in order to orient the users in the application of the various laws. CLICK partially reuses the material already published in text books and manuals by the publisher, updated with the text of the most recent laws and their related case law and practice.

3. HYPERMEDIA STRUCTURE DESIGN

Structure design refers to the way information is organised within a hypermedia application. Structure design requires the ability of abstracting the general organisation patterns from the analysis of the application content and user requirements. This design phase also requires a structural design model, i.e., a set of primitives to describe precisely and concisely the structu-

---

1 CLICK is regularly updated every three months.
reral aspects of an application. Structure design of a hypermedia application results in the specification of a set of structure types, at two levels, in-the-small and in-the-large. In the following sections, the description of the structural design will be discussed using the terminology of the HDM Model. Further information about this hypermedia design model can be found in references [1, 3, 4, 5].

3.1. Structure Design in-the-Small

Hypermedia structures in-the-small are called nodes and slots in HDM terminology. A node represents a semantic unit that can be regarded as atomic, i.e., it cannot be further decomposed. A node aggregates atomic or composite pieces of information that are called slots and are made of simple or composite values of different nature (text, formatted data, images, sound...) [9].

The visualisation of slots within a node, their use, and their interaction with the user, concerns dynamic and lay-out design and will be discussed in sections 4 and 5.

Structure design in-the-small results in a set of in-the-small structure types, i.e., slot-types and node-types. Slot-types can be simple, denoting multimedia data-types such as “text”, “video-clip”, “still”, or can be composite, denoting aggregations of simple or in turn composite slots.

A node type defines a class of nodes that aggregate slots of the same type, and represents semantic units of the same “class”. Examples of node types in our case study applications will be discussed in section 3.3.

3.2. Structure Design in-the-Large

In-the-large structures are “large” granules of information that aggregate several nodes and represent complex objects or relationships of the application domain. HDM provides two primitives for in-the-large structures, called collections and webs.

A collection represents a physical or conceptual entity of the application domain, or a set of information structures – node or collections – that are aggregated for a specific purpose (e.g., because they are concerned with the same general topic) [6]. A web is used to represent a relationship between a node and another node, or between a node and a collection.

A collection is made up of a node called collection node, a set of members, and a topology, that defines how members are organised within a collection. Typical topologies for collections are sequences and trees. Different topologies induce different navigation patterns, as we will discuss in section 4.
Collection members can be nodes, or other collections in turn. In the latter case, collections will be called nested. The collection node has the purpose of introducing the collection, and of allowing access to its members. Very often, its content is (partially) derived from the collection members, in that it reuses some slots occurring in the nodes of the members.

Structure design in-the-large of a hypermedia application results in the specification of a set of collection types and a set of web types.

Collection types define classes of collections representing similar objects, or sets of objects that are aggregated with similar purposes. Collections of the same type all share the same topology, and have collection nodes and members of the same types. A web type defines a class of relationships that hold among nodes or collections of given types.

3.3. Examples of Structure Design

The discussion of structure design in CLICK will proceed top-down, from the analysis of slot and node types (structures in-the-small) to the description of collections and webs (structures in-the-large).

Slot Types in CLICK

In CLICK, simple slot types denote strings of text, positive numbers, or dates. Examples of aggregated slot types are:

- “title”, made up of the composite slots “identification”, “subject”, and “general topic”; “identification” is in turn made of the slots “regulation type” (e.g., Law, Decree, EC Regulation, etc.), “number” and “publication date”;
- “paragraph”, that represents a portion of the plain text of a legal document or an article of a law;
- “annotated paragraph”, made of a paragraph slot and a set of additional text slots that either comment on its interpretation or list some related laws;
- “paragraph with versions”, made of an annotated paragraph and a text slot which contains the “evolution” of a law paragraph, i.e., its previous versions.

Node Types in CLICK

Nodes of CLICK are of various types. Some of them define different categories of legal knowledge (of different granularity): “Law Article”, “Official Law”, “Law Enforcement” (i.e., practice), and “Case Law”.
A node of type “Law Article” stores an article of a law integrated with a detailed interpretation and discussion of its content; it is made of a slot “title”, and a list of text slots of type “paragraph with versions”.

Nodes of types “Official Law”, “Law Enforcement”, and “Case Law” contain a slot “title”, and a list of text slots of type “paragraph” or “annotated paragraph”.

Finally, our case study includes various node types describing collections nodes. The simplest type – “Document List” – denote collection nodes made of a list of slots of type “title”. Other collection node types will be discussed below, when we will explain the semantics of the various collections defined in CLICK.

**Fig. 1. A Node of Type “Law Article”**

Fig. 1 shows a node of the type “Official Law Article”; the text slots storing paragraph interpretation and related laws are hidden, and the slot with the paragraph evolution appears close to the related paragraph.

**Collections and Webs in CLICK**

CLICK provides many collections of a different nature and with different levels of nesting, that aggregate nodes and collections of various types according to different criteria:
• Collections of type “Interpreted Law” group all nodes of type “Law Article” that refer to the same law and are particularly relevant for a specific fiscal problem (e.g., VAT payment). Their collection node contains a title slot, that identifies the whole law, and a list of slots that identify the number and topics of each specific article considered in the collection (see Fig. 2).

**Fig. 2. A Collection Node of a Collection of Type “Interpreted Law”**

- Collections of type “Complementary Legislation” group all nodes of type “Official Law”, “Law Enforcement”, or “Case Law” that discuss a complementary issue of, or an integration to, the rules stated in a given law article or in a set of law articles. Collection nodes are of type “Document List”.
- Collections of type “Interpreted Law”, and the corresponding collections of type “Complementary Legislation”, are all grouped in a higher level nested collection called “Summary”, shown in Fig. 3. The Summary collection node is the first one presented to the user when he has access to CLICK.
- All nodes of type “Case Law” that are relevant for the application of a given law article are first grouped by topic in a collection of type “Case Law References by Article & Topic”, and then organised into
a higher level nested collection of type “Case Law References by Article”. The collection nodes of the latter collections contain the list of the titles of “Case Law” member nodes, and, for each member, a text slot that summarises the issues that are relevant for the associated law article.

Similar collections are defined for the set of “Law Enforcement” nodes, and their type will be called “Law Enforcement References by Article & Topic” and “Law Enforcement References by Article” (an example of which is shown in Fig. 4).

- Some collections have the purpose of grouping “Official Law” nodes, or “Case Law” nodes, or “Law Enforcement” nodes by type only, or by type and by time period. For all these collections, the collection node is of type “Document List”.

- By-type collections will be called “Standard Collections” in the rest of this paper. All collections “by type and period”, such as “Case Law - 1994”, are in turn grouped in higher level nested collections, such as “Case Law - Chronological Index” (see Fig. 5).

- Other collections group documents by topic, such as for example, “Documents concerning international fiscal rules”. They are all grouped into a higher level nested collection called “Analytic Index”.

Fig. 3. A Collection Node of the “Summary” Collection
Fig. 4. The small window in the foreground on the screen shows the Collection Node of Collection “Law Enforcement References by Topic - [DPR 633 - 26/10/72, art. 5]”. This collection is a member of a higher level collection of type “Enforcement Law References”, the collection node of which appears in the background.

The Analytic Index and the various chronological indexes are grouped in a higher level nested collection called “General Index”.

Finally, collections can also be dynamically generated by query, i.e., by specifying multiple conditions on nodes of the various types. In all these collections, the collection node is of type “Document List”, and contains the titles of all member nodes (in chronological order) that satisfy the query specification. This issue will be further discussed in the examples presented in section 4.

Only two web types are defined in CLICK. They represent the relationships between nodes of type “Law Article” (first argument) and the corresponding “Case Law References by Article” collections (second argument), or the corresponding “Law Enforcement References by Article”, respectively. These two web types can be called “Case Law References” and “Law Enforcement References”. No web is defined to represent the inverse relationship.
Fig. 5. The small window in the foreground on the screen shows a node of type “Document List” which denotes the Collection Node of Collection “Case Law - Chronological Index”

4. Dynamic Design

Dynamic design concerns two different aspects of a hypermedia application: dynamic in-the-large, that refers to the way the user can move around across the different pieces of information, and dynamic in-the-small, that refers to the interaction of the user with individual slots within nodes.

4.1. Dynamic Design in-the-Large

In modern hypermedia applications, dynamic in-the-large amounts to a combination of Navigation (i.e., link traversing), Data Base Queries, and Content-based Search, with relative relevance depending upon the “style” and the intended use of the application.

Data base queries and content-based search are particularly relevant in hypermedia applications for legal domains, which typically store a large amount of documents and should provide facilities to retrieve them. This paper will focus only on the dynamic features concerning navigation, since they are the most specific.
Designing the navigation features of a hypermedia application requires specifying the navigation links, i.e., the navigational interpretation of the various structures in-the-large defined during the structure design phase. According to the HDM model, various categories of navigation links can be defined [8]:

- **Structural Links**
  Structural links connect the constituents of a collection, and are induced by its topology. Very often, structural navigation in hypermedia applications is a combination of three main patterns, induced by the two main topologies of collections (sequences and trees). The *tree pattern* is induced by tree-shaped collections. According to this pattern, links connect each member to its children and father node, as well as to the collection node.
  The other patterns are induced by sequence topologies. According to the *index pattern*, links connect the collection node to each member, and vice versa. They allow the random access of any member from the collection node, and navigating from any member back to the collection node.
  According to the *guided tour pattern* [14], there is a “start” link from the collection node to the first member (as well as a “restart” link – its inverse one); in addition, there are the links “next” from each member (but the last one) to the next one, “previous”, from each member (but the first one) to the previous one, and “first” and “last” from any node to the first and the last nodes respectively. Therefore, in guided tour navigation, it is possible to move from one member directly to the next and previous member without the need to traverse each time the collection node\(^2\).

- **Web Links**
  Additional navigation patterns are derived from the definitions of webs. A web defines a relationship between two objects – nodes or collections – and induces a link from the first “argument” of the relationship to the second one.

- **History Links**
  Hypermedia applications very often keep track of the navigation history, and store the set of visited nodes in a chronological order.

\(^2\) A minor variation of the guided tour pattern is the *circular guided tour pattern*, in which the link “next” is defined also for the last node and allows the user to move from there to the first member of the collection.
This structure can be regarded as a dynamically created collection, whose links are defined at run time, as the navigation session proceeds. Typical navigation on a history collection follows the index pattern, to allow a random selection of any previously visited node. In addition, or in alternative, many hypermedia applications provide dynamically created “back” links defined according to the activation order of the visited nodes, that allow the user to backtrack on the navigation session.

Once the set of links is defined, a further design choice is to define the semantics of link traversing. In general terms, activating a link from its source leads to the activation of its destination; if the latter is a collection, the natural interpretation is that the collection node is activated. The concepts of node activation, however, need to be elaborated by the designer. When the destination node contains several slots that cannot be shown simultaneously on the screen, for example, the designer must specify which of them are immediately visible when the user has access to the destination node, and which of them are instead shown upon the user’s request. In addition, this interpretation needs some extensions when active, time-dependent media are taken into account. If active pieces of information were being played in the source node, for example, the state in which the source itself is left must be defined (e.g., the original state, or the state reached at the moment of departure from the node). If, for example, a video and a sound were being played, it must be specified whether those slots are to be suspended, or reset at the beginning, or whether one of them must continue playing [10, 11].

Finally, the designer must specify whether the link has a replacement behaviour, i.e., source node is de-activated and hidden, or if it has an additive behaviour, i.e., the source node remains active on the screen.

4.2. Dynamic Design in-the-Small

Dynamic in-the-small concerns the user interaction with information values within nodes, such as hiding a slot, moving a slot in the screen, opening a hidden slot. Dynamic in-the-small is related to the type of value

---

3 As we will discuss in the examples presented in section 4.3, the activation of a sequence of additive links may result in having too many nodes simultaneously active. This situation raises the issue, for the designer, of defining a sound strategy to automatically de-activate some nodes, in order to take into account lay-out constraints and to reduce the potential visual overhead for the user.
stored in the slots. While the interaction with static values such as text, picture, or numbers is very limited, the interaction with a time-based multimedia value can potentially have complex features. If a single active value is included in a node (an audio track, for example), it may be played by default, or it might be played upon request, as an option to the reader. Playing control over video, sound, or animation, etc., are all examples of hypermedia interaction in-the-small. When several active media slots are involved (say multiple sound tracks, videos, animation, etc.), there are more complex issues concerning parallel presentations with proper synchronisation, or well-sequenced presentations.

4.3. Examples in CLICK

CLICK provides both query-based and navigation-based access to the information.

The query mechanism allows the user to retrieve all nodes of type “Law Article”, “Official Law”, “Practice” or “Case Law” that satisfy given properties. Such properties are described by specifying a value, or a range of values, on an AND/OR combination of attributes: their type (e.g., State Law), their “category” (e.g., High Court Decision), their identification code,

Fig. 6. Query Form of CLICK, and Specification of a Simple Query Statement
The execution of a query returns a dynamically generated, temporary collection, the members of which are all nodes that satisfy the query condition. Its collection node is of type “Document List” and contains the list of the titles of all retrieved nodes. Navigation of this collection follows the index pattern.

The query mask of CLICK is shown in Fig. 6, where a simple query is specified. After executing this query, the user can visualise the collection node of the results, shown in Fig. 7.

In CLICK, a navigation session can start either from the Summary collection node (see Fig. 3) or from the General Index. From any node, it is always possible to return to both collections.

Collection navigation mainly follows the index pattern. In general, from the collection node it is possible to select any member, and from any member to return to the collection node. Surprisingly enough, a different pattern is used by collections of type “Interpreted Law” grouping nodes of type “Law Article” that refer to the same law (see Fig. 2). The user can

---

4 This collection is temporary in that it is not persistently stored in the application, but is destroyed each time a new query is executed, or when the application is closed.
access any law article from this collection node, but there is no explicit link to directly return back. In order to return to the collection node, the user can either activate the history collection, to “Go Back” to the previously visited node, or can return to the collection node of the Summary collection and then select again the specific “Interpreted Law” collection he is interested in. The first solution can be disorienting, since the same navigation action, “Go Back”, is used both for history navigation and for index navigation. The second solution requires the user to perform a movement in two navigation steps which he might expect to perform in one shot.

With respect to possible navigation links for collections, a number of possibilities are missing, e.g., backward scan with respect to the collection order (independently from the historical order of navigation) and direct jump to the last or the first member. The lack of these connections has the advantage of keeping the interface simple (as discussed in section 5), but it has the disadvantage of an obvious lack of richness in the navigation features. If the user has to perform a well defined search task that requires extensive navigation, the limited power of collection navigation in CLICK can be quite disappointing and may create usability problems [12, 13, 15]. If the user, instead, has just the purpose of browsing around, looking for interesting information with no specific task in mind, the lack of richer navigation features may probably not even surface.

More powerful navigation is provided in the so called “Standard Collections” (grouping all “Official Law” nodes, or “Case Law” nodes, or “Enforcement Law” nodes, as discussed in section 3.3). Members of these collections are chronologically ordered, and it is possible to move from any member to the next or previous node.

The problem with these navigation links is the fact that they are always available in a member node even if a collection different from a Standard one is used to access it. Consider for example the situation in which the user is exploring a given law article, and, from here, he accesses the collection “Case Law References by Article” to explore the Case Law documents that are relevant for the article’s interpretation. After selecting a specific “Case Law” node in this collection, he would probably interpret a “next” command as “go to the next Case Law which is of interest for the current law article”. Unfortunately, that “next” link takes the user to the Case Law node which chronologically follows the document he is searching, and might have nothing to do with his current topic of interest.

5 The “Back” link in the navigation session history is dynamically created on any visited node.
Web navigation is extremely limited, since there are only two web types defined in the application. Web links connect a nodes of type “Law Article” and the related “Case Law References by Article” collections, or the related “Enforcement Law References by Article”. Since no web is defined to represent the inverse relationship, there is no link that allows, for example, the user to navigate from a Case Law to all relevant “Law Article” nodes.

As far as the dynamics of link activation is concerned, it is defined in several different ways. In general, the behaviour of links is a combined property of the nature of the link, of the source and destination nodes. As a consequence, there is a variety of navigation behaviours in this application, some of them confusing or inconsistent. Any link to the Summary collection nodes has the effect of hiding the link source, unless it is a collection node of type Document List, or is the collection node of collections “Case Law References by article” or “Enforcement Law References by Article”. However, if the user moves from a node A to a node B (by an additive link, so that both A and B are visible), and them from B to C (by an additive link), when C is shown A is sometime hidden, while it sometime remains visible, based on totally unpredictable rules. Similarly, we have noticed that if the user activates a sequence of additive links, so that three nodes are already visible on the screen, say A, B, and C, the activation of a further additive link has sometime the effect that only the first node (“A”) is hidden, while in other cases both A and B are. Apparently, different dynamic strategies are applied in very similar situations, with no evident reason for different behaviour.

As far as dynamic in-the-small is concerned, the behaviour of our case study is rather simple, since information values are only static data, such as text or number. The only user interaction in-the-small concerns composite slots “annotated paragraph” or “referenced paragraphs”. When a node is activated, only the paragraph components of these composite slots are visible. References, comments, or previous versions of the paragraphs are displayed upon the user’s request (by selecting the correspondent buttons, as discussed in the following section).

5. Lay-out Design

Lay-out design concerns the pure aesthetics of an application, and refers to how the content, the structures, and the dynamic features of the application are presented to the user. Lay-out design defines the arrange-

6 As we will discuss in the following section, three is usually the maximum number of nodes (e.g., windows) that are simultaneously displayed in this application.
ment of the screen of individual information values as well as of groups of information values, and the way of displaying to the user the functionalities for interacting with these presentation elements. This design phase involves expertise in visual design and visual communication, i.e., the ability of using expressive visual cues to communicate the meaning of information objects and functionalities. Lay-out design usually results in the definition of a set of visual templates, defining the general lay-out pattern of the various types of nodes (corresponding to different types of information structures).

5.1. Examples

The lay-out design of CLICK is based on several templates. Some of them are full screen templates while the others are windows of approximately 1/3 x 1/2 of the screen that can be resized by the user (see figures 1, 2, 3, 4, and 5). We will discuss here only the main template, that we will call "document template" and it is used to present nodes of type "Law Article" (see Fig. 1), "Official Law", "Case Law", and "Law Enforcement".

A "document template" is represented by a full screen window, structured in four main sections. On top of the screen we can see the menu bar, showing a number of functional buttons that provides operational facilities other than information access, such as annotation, marking, printing, etc.

Below the menu bar we can find a collection navigation bar, which allows the user to activate the query tool or displays the buttons for the links to the Summary collection, to the General Index, to the History Collection, to the previously visited node ("Back"), or to the previous or next node in the Standard Collections of the current node.

The rest of the space is structured in a heading, showing the document title, and a body, showing the actual content of the document stored in the node. In nodes of type "Law Article", the heading can also contain two buttons denoting the links induced by webs of types "Case Law References" and "Law Enforcement References".

Different background colours for the heading are used to highlight the different types of nodes. Heading backgrounds of nodes "Official Law" and "Law Article", for example, are red, while in "Case Law" nodes they are pink.

Both in the heading and in the body, various small buttons may appear, shaped as red numbers in brackets or a green small letter "E". These buttons are used for interaction in-the-small, to open the notes associated to the various text paragraphs. Notes are shaped as small scrollable windows (that appear close to the related paragraph). Different shapes of the buttons
denote different kinds of notes. Square brackets stand for interpretation notes, while round brackets stand for the notes listing the references to related laws, and the “E” button allows the user to access the versioning note.

6. Conclusions

Until a few years ago, hypermedia application development was mainly conceived as an artisan activity, performed on the basis of individual creativity and experience. Assessed methodologies and models to support the development of in a systematic, engineered, and efficient way, were absent. Today this approach is less acceptable, due to the increasing complexity and size of new hypermedia applications (and the stronger role of active media such as video, sound, and animation). It has become crucial, therefore, to transform hypermedia development into a more organised, engineered and efficient process.

In this perspective, the design phase plays a fundamental role in the overall process of hypermedia application development. Systematic design methods force a team of developers to focus on the rationale of the application to be developed, and to take decisions at the proper level abstracting as much as possible from implementation requirements. Most of the “conceptual” (i.e., technology-independent) problems and inconsistencies can be potentially detected and solved at design level, before actually implementing the application. As a consequence, precise and coherent design may reduce implementation time and potential mistakes, and, in principle, the better the design the higher the quality of the final application, in terms of coherency and consistency with respect to its requirements.

This paper has discussed our approach to the hypermedia design problem. The proposed method is based on several years of experience gained by the authors in developing hypermedia applications in various domains, and in analysing the design of applications developed by others. We have identified the main activities that, from our experience, are involved in the design process: content analysis, structure design in-the-large and in-the-small, dynamic design in-the-large and in-the-small, and lay-out design. They can be regarded as a check list of crucial tasks to guide designers in performing their work more systematically and in a better organised way.

Finally, we have introduced a set of primitives, based on the HDM design model, that allow the design team to describe the output of the design process and to specify precisely the application they wish to develop.

The method and model we have proposed can be used not only to
design a new application, but also to describe, to analyse, and possibly to evaluate applications developed by others, without having our approach in mind. In fact, once an existing application has been analysed according to the various design dimensions and has been precisely described with HDM, some general usability criteria [12] can be applied, in order to identify possible weaknesses.

To exemplify the overall approach, we have analysed and discussed a hypermedia application in the legal domain, pointing out the relevant design choices and outlining potential usability problems.

REFERENCES

2. EBC, Click - La Banca Dati Fiscale, Milano, 1994.