METHODOLOGIES FOR DESIGNING AND DEVELOPING HYPERMEDIA APPLICATIONS

by

Tomas Isakowitz

Manfred Thuering

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

METHODOLOGIES FOR DESIGNING AND DEVELOPING HYPERMEDIA APPLICATIONS

EDITED BY

TOMAS ISAKOWITZ
Information Systems Department
New York University
44 West 4th St.
New York, NY 10012
USA
tomas@stern.nyu.edu

MANFRED THÜRING
empirica Communications and
Technology Research
Oxfordstr. 2
D-53111 Bonn
Germany
thuering@bifoa.Uni-Koeln.DE

December 1994

The articles in this collection were originally presented at the workshop on hypermedia design and development that took place in Edinburgh, on September 18, 1994.

Working Paper Series
Stern #IS-94-8

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

TABLE OF CONTENTS

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

INTRODUCTION

Tomas Isakowitz and Manfred Thüring

The articles in this collection were presented first at the First International Workshop on Hypermedia Evaluation and Design organized in Edinburgh in September of 1994, in combination with the European Conference on Hypermedia Technologies (ECHT-94).

Hypermedia design for commercial purposes, e.g., technical documentation, catalogues, encyclopedias and training manuals, is a taxing task that usually results in large and complex applications. These applications must be of high quality and easy to maintain. Moreover, their development must be timely and often has to be accomplished with limited resources. As a consequence of the current lack of development methodologies for designing such applications, hypermedia development projects are unduly lengthy and hard to manage, frequently producing systems that are also difficult to handle and to maintain.

These problems arise from at least three particularities of hypermedia design. First, there are many different components involved, such as user-interface, navigation, database store and content preparation. Second, there are no conceptual frameworks to the process because more traditional data models, e.g., data flow diagrams, E-R diagrams, flowcharts as well as object-oriented hierarchies, are insufficient or even inappropriate. Third, the synergy among the various aspects of a hypermedia application stretches the boundaries of commonly established roles for software developers.

To take account of these specificities, the design of hypermedia applications cannot be based on common sense rules of thumb, nor can it be undertaken in an ad hoc fashion. Instead, there is a need for systematic design procedures and quality criteria to guide the design, development and evaluation of hypermedia applications. The articles that appear in this collection address different aspects of hypermedia design.

The articles by Schuler and Nanard and Nanard deal with understanding and supporting the design process. Schuler describes how a *design space* to support the design activity on a collaborative environment can be based upon a software engineering tool (HIFI). Nanard and Nanard focus on the human aspects of an interactive design process based on observations on MacWeb. They describe the synergetic activity between human and tool and the intricate process that enables the design to evolve through a series of continuous feedback loops.

At the moment there are no guidelines, or row models to enable an objective evaluation of a hypermedia application. Such criteria are in demand, as users and deveklopers face the uncertainties of measuring the quality of hypermedia applications. The article by Garzotto, Mainetti and Paolini focuses on the problem of evaluating hypermedia applications and proposes a framework that can be used to define such criteria.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

The articles by Balasubramanian, Isakowitz and Stohr and by Schwabe and Rossi address methodological issues by describing a life-cycle approach to the design and development of hypermedia applications. These methodologies are apt for structured domains where economies of scale can apply, i.e., there are many objects of the same kind. The key difference between the approaches lies in that Schwabe and Rossi is object-oriented, while Balasubramanian, Isakowitz and Stohr's is based on an entity-relationship approach.

The article by Thüring focuses on the intra-organizational issues that arise as hypermedia applications are designed and developed. There are various human and organizational complexities involved in the design process. Different people, designers, users, analysts, developers, etc. are part of it, as are the various departments in an organization, such as marketing, software development, planning, etc. Thüring's paper provides a framework to guide organizations involved in commercial development of hypermedia applications.

We want to thank those people that made the workshop and this publication possible. In particular, we thanks David Meyer for his efforts in compiling and editing this collection.

We hope you enjoy these readings and that you can benefit from them by applying some of the ideas in your own activities.

(New York, December 1994)

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Key features in a hypertext design environment for supporting the process of hypertext structure design

Jocelyne Nanard, Marc Nanard
LIRMM
161 Rue Ada, 34392 Montpellier Cedex 5, FRANCE
Tel. : (33) 67 41 85 17, Fax : (33) 67 41 85 00
e-mail: jnanard@lirmm.fr, mnanard@lirmm.fr

ABSTRACT
Hypermedia design, as any other design activity, may be observed according to two points of view: *methods* which suggest milestones to guide the designer's work and *process* which concerns the actual detailed behavior of the designer at work. Cognitive studies assess that mental processes involved in any design process show widely shared human characteristics regardless to the used design method. Thereby, they provide general keys to help designers. Thus, a hypertext design environment should equally consider the two dimensions of a hypertext design activity, in particular it should support the natural design process specificities, mainly the incremental and opportunist aspects. The paper focuses on the hypertext design as a computer supported human activity. It examines what is general both in the design methods and in the design process of hypertexts in order to determine which general features are helpful to designers. This analysis has raised from the observation of the behavior of MacWeb users during design tasks. It is related to sound and well known results in cognitive science. The paper also describes how the proposed features are implemented in the MacWeb system.

KEYWORDS: Design process, Hypertext structure, Hypertext structuring tools, Hypertext rhetoric.

1. INTRODUCTION

The need for making hypertext structure explicit [22] has been mentioned by so many authors that it is no longer necessary to argue for it. But a lot of divergence exist between approaches for designing a hypertext structure.
In some approaches, the structure is made explicit before producing the hypertext. For instance, HDM offers a helpful design approach and design model [4] to escape hand-crafted development for large hypertexts in order to ensure their consistency. Documents structured editing has been extended to hypertext [16]. Many hypertexts for technical documentation are directly produced from SGML documents taking advantage of their DTD. HyTime [15], though at its beginning, illustrates the increasing importance of such approaches.
Conversely, in some other approaches the structure results from the interactive process of hypertext. development. Marshall's experience of Aquanet has pointed out how structure emerges from visual presentation [8]. The absence of an explicit schema is not incompatible with a clear and highly elaborated structure. A lot of hypertexts that are famous for their readability have been freely authorized and do not directly result from instantiating any pre-designed schema. WWW exponential growing during the last year relies on a free worldwide cooperation. Its weak but existing structure is a *a posteriori* one and is a significant result of collaborative work. Mark Bernstein suggests that all of these divergence rely on 'three fundamentally different and incompatible metaphors: Mining, Manufacturing and Farming [3], which 'shape the design and the rhetorical style' of hypertexts.
Anyhow, hypertext applications are so different that no single method is relevant for designing all of them. But since designers are humans, they run a design process depending upon general human factors regardless to the used method. This paper focuses on the hypertext design itself as a computer supported activity. Thus, it does not aim at

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

promoting a given design method versus another. It examines what is general in the design process of hypertexts in order to determine which features are helpful to designers. This analysis has raised from the observation of the behavior of MacWeb users during design tasks and is related to sound and well known results in cognitive science. The paper also describes how the proposed features are implemented in the MacWeb system.

2 DIMENSIONS OF HYPERTEXT DESIGN ACTIVITY

Hypermedia design, as any other design activity, may be observed according to two points of view:

- prescribed methods to organize the design process
- observed facts on how the design process is actually run.

Hypermedia works mainly focus on the first point. Observation of the behavior of hypertext designers using constraintless hypertext design environments has shown us that the design process itself, whereas it is often neglected, is an important aspect. This section argues that hypertext design should consider the two dimensions and respect the natural design process specificities, mainly the incremental and opportunist aspects. It explains the concerns of the two points of view and focuses on cognitive aspects of the design process. As a consequence, it claims that a hypertext design environment should equally support both aspects of the design activity.

2.1 Design methods and design process

The first point of view of a design activity is product-oriented, it addresses the management of intermediate states of the product development ; it is concerned with *design methods*. The second one is human-oriented, it addresses the actual structure of sequences of user's actions and thoughts actually used to reach the goal ; it is concerned with cognitive aspects of the design processes. Since the efficiency of the designer's work is as important as the final product itself, a model for hypertext design should take into account both methods and processes.

A *design method* is a recipe which guides someone along a task and provides with a set of milestones for finding his or her way. A method suggests directions, even constrains the user to follow them, but is not concerned with the efforts nor the stepping actually done to reach each milestone. A *design model* offers convenient concepts for representing the result of a design. A *design method* enforces some order and some rules to produce a design with respect to a design model. For instance, RDM [1] or OOHDM [19] provide concepts for representing design specific to each of the steps of the methods axis described in figure 1.

Unlike a method, a cognitive process is concerned with the actual sequence of mental states and actions which are followed to step from milestones to milestones. Whereas different methods propose different milestones to achieve the design, processes used by designers to step towards them present great similarities. Regularities that are observed in human behavior during the design process of hypertexts rely on universal mental schemes of humans. Thereby, they provide general keys to help designers at work. They are described in next section.

The rigor of methods and the apparently erratical organization of processes are not contradictory, they are complementary. They are two independent aspects of the same phenomena observed from different sides. Thereby the design activity takes place in a space based on these two directions. Accordingly, a design environment must take both into account. Since most of papers in these proceedings are concerned with design methods -ranging from conceptual model design to marketting and usage issues [23]-, the present paper focuses more on helping mental processes involved during design.

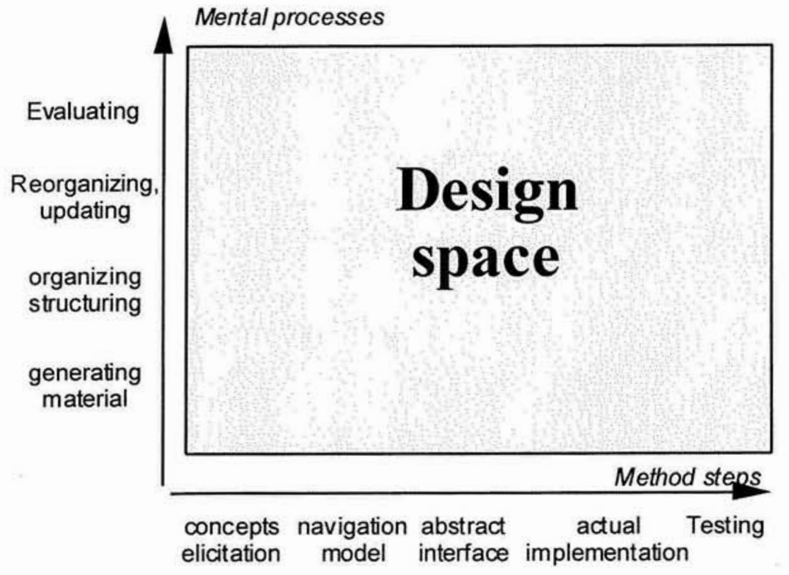
2.2 Cognitive aspects of design processes

In order to determine what kind of help the user needs to support the design process of a hypertext, this section presents a brief overview of general cognitive results about design processes and their relationships to hypertext design. Additional information can be found, for instance, in [24].

The main result from psychological and experimental studies is that design activities involve an opportunist mental process. The structure which is observed in resulting objects rarely rises at once, but rather results from a long and recursive process with backtracking steps, switching erratically between spring of ideas, production, re-organization, modification, evaluation... A good illustration of such a structure is provided by the model of the writing process by Hayes and Flowers [7], based on sound experimental studies of text writing. We have observed that the process of hypertext design involve the same sub-processes. They are represented in the mental processes axis of figure 1. Only strategies of sequencing these sub-processes differ between designers. Thus an interesting observation we have found is that the process is as well top-down or bottom-up. Explanations of involved cognitive mechanisms are given in [13], [17].

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Other studies exhibit the importance of activity spaces [20] and focus on the fast switching between these activity spaces in the design environment in order to support the design process. For instance Sepia [21] is based on distinct but tightly interconnected activity spaces.
The complexity and the apparent chaotic organization of the human activity is natural. It must not be hold up, the author's creativity would otherwise be drastically reduced. Moreover studies in literature show that the quality and the richness of the resulting document is often correlated to the apparent complexity of its creation process.



It results from the previous discussion that the major characteristics of any design process is a set of intricate feedback loops evaluating the effect of design choices on the final result. As a consequence, the actual design activity is not a simple stepping along the methods axis. It rather is a Brownian motion within the activity space.

Figure 1: The design activity as a two dimensions space: method steps and mental processes.

Practically, any design activity is always a complex mix of sub-activities. Some of them take advantage of skill, some others are more like pure creation. Designing hypertext is such a mix. Even when a method drives the design, the actual underlying process is rarely simple. For instance, designing the data schema of a hypertext rarely is a backtrackless activity. It often involves many backtracking including evaluation sub-processes, at least in the designer's mind. Who denies it? When most of the design is done 'on paper', backtracking and feedback loops are often hidden since some of them take place only in the designer's brain. The written information is simply a notation of the final stage of each design step; only very few information are transcript on scratch pads, but they assert the chaotic nature of design process. At the opposite, when a computer-aided design environment is flexible enough to support the designer actual activity, the feedback loops become observable. Studying designers at work makes obvious that cycles exist across method steps -even when the method aims at avoiding them-, It also makes obvious that choices done at one step often implicitly depend on choices which are to be done in further steps. The dependence of steps is often hidden by the skill of the designer who reasons on the feedback loops necessary to find a consistent global solution. Anyhow the drawbacks of early choosing, like in pure top-down oriented approaches in software engineering, have been deeply focused on. We claim that hypertext engineering do not escape the problem. The structure of a hypertext emerges as the result of the author's efforts to tune-up a rhetoric which makes the information induce the wished mental schemes in the reader's mind [10]. Structure is not a pre-defined container in which an author pours information.

2.3 Consequences on hypertext design environments

A *design environment* provides tools, operations and concepts that help produce some design and sometimes develop the corresponding product. I order to be truly efficient, it must equally help both aspects of the activity space and not be restricted to only one of its aspects. Especially, it should take into account the cognitive aspects related to the design process as well as enforce the global strategy suggested by the method.
Design must be supported as a whole, without shame of respecting the specificity of human behavior. Opportunistic and incremental does not mean messy and is fully compatible with the rigor of a method! La Bruyère, a French author of the 17th century, well known for the quality of his writing, suggested a universal method to produce good

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

works in the following terms: “vingt fois sur le métier remettez votre ouvrage, polissez le et le repolissez sans cesse.”[1]

We conclude that since the hypermedia design process is incremental and opportunist:

- "Man in the loop" is an essential factor.
- Hypermedia prototyping should be part of the design process.

This is why the MacWeb hypertext design environment, similarly to computer-aided design environment and beyond specific hypertext design methods, provides the designer with tools for prototyping, evaluating, backtracking and so on. Furthermore, unlike Sepia, MacWeb offers a continuous activity space in order to easily support interaction between different products of activity stages (mainly structure or mainly material or mainly presentation). However, the computable hypertext model, the associated formalism and the environment provide concepts and operations allowing to take into account each level of the design activity (generating material, structuring and so on) separately.

3 MACWEB AS A HYPERTEXT DESIGN ENVIRONMENT

In this section, we take advantage of the MacWeb[2] system to exemplify the points that we consider as important in a hypertext design environment. It must :

- provide designers with *a computable model and formalism* able to handle in a unified manner any levels of abstraction of design, i.e. take into account the efforts along the method axis,
- support the incremental, explicit and opportunist design of the hypertext structure, i.e. take into account the efforts along the process axis,

We start by briefly reminding the MacWeb system itself and its basic features.

3.1 A few words about MacWeb

MacWeb[2] is a hypertext system research prototype, developed, incrementally improved and tuned at LIRMM since 1988. It is used as a flexible test bed for validating theoretical works on hypertext as well as for experimenting on users or designers behavior in real work situation. Different scales of applications have been handled with this system. The widest one currently concerns more than 35000 links. Several strategies for producing hypertext have also been used, ranging from fully automated ones (relying on natural language analysis and knowledge elicitation), to pure hand crafting. Tested applications belong to the three classes defined by Mark Bernstein as ‘Mining, Manufacturing and Gardening’.

MacWeb handles a structure called a ‘Web’ which is a network of typed nodes connected by typed links. The system architecture relies on the Dexter Model [6]. Conversely to many other hypertext systems, MacWeb gives a very important role to the typing of nodes and of links. It often uses typing as means for representing semantic relationships between nodes. The object-oriented model is convenient for representing knowledge in the hypertext as well as for managing the hypertext structure at several levels of abstraction. Since this aspect has been presented in other papers, it is no longer developed here. We just remind the principle of the mechanism ; for more details, see [11] [12].

Types of nodes and types of links are freely defined. A reflexive mechanism allows to specify both the semantics of types and the hypertext: types themselves are represented as nodes connected by typed links which stand for the relationships between types. So, MacWeb makes it possible to edit in a unified graphical manner the types specification as well as the hypertext graph itself. The ‘web’ is compound of a ‘main web’ and a ‘web of types’. According to the object-oriented model, the nodes in the main web can be considered as instances of the classes defined in the web of types. ‘Script’ typed nodes are used to define methods owned by the classes. Inheritance, multiple inheritance, overriding are very simply expressed in a graphical manner in the web of types. Since the

[1]free translation : “iterate on your work design, tune it and trim it again and again.”

[2]MacWeb is the name of a system we developed at LIRMM since 1988. The same name is now used by MCC for its WWW displayer running on Mac. The two software are strongly different.

structures are dynamically interpreted, MacWeb supports run time interactive and incremental edition of the classes specification.

3.2 A computable model and formalism

The major feature that a hypertext design environment should provide is *a computable model and formalism* -as suggested in OOHDM [19]-. It would be helpful to handle in a unified manner any levels of abstraction of design (analysis and requirements, specifications, conceptual design, as well as the evaluation of the product [5] and the design rationale [20]). Today, none or very few systems integrate all of these features within a single environment. A Hypertext Design Environment aims at fully supporting the design activity. Thus it must be able to handle, in a unified manner, all kinds of structures that the designer has to deal with when designing. Due to the existence of feedback loops, the designer frequently jumps forward and backward from one step to another. It is important to reduce the burden of switching from one piece to another and to avoid conversions from one formalism into another. So, it is important that the models and formalisms used at each step of the design be directly compatible, and integrated within a single and unified work environment.
The MacWeb system uses a single model and a single graphic formalism to handle it. We show how this object-oriented model is well adapted to handle the different design aspects of the method axis in figure 1.

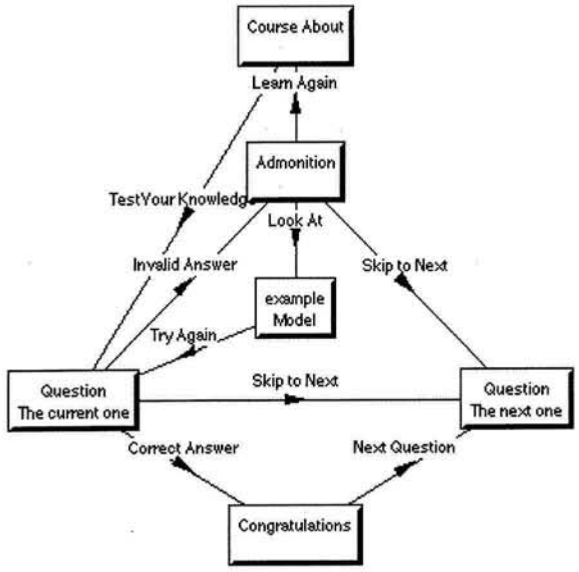
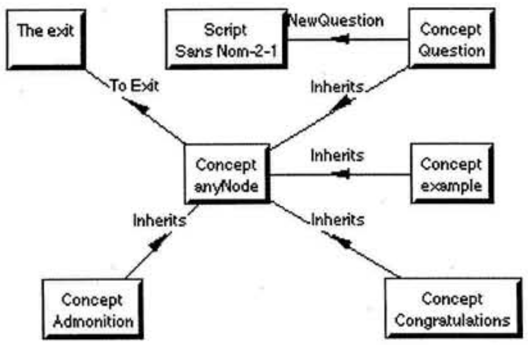


Figure 2 (above): Abstract navigation model.

Figure 3 (right): Using inheritance to specify abstract navigation.

Figure 3 (left side) illustrates the abstract navigation model within a set of exercises associated to a course. Observe how links from one question to the next one are expressed between two instances of the same class. The navigation structure is specified here regardless to the actual anchoring which obviously depends upon the actual node contents. Similarly, in the figure 3, the 'To Exit' link is defined at the level of the abstraction 'any node'. It appears in 'question' nodes due to the inheritance relationship. 'Question' node have a method named 'New question' which enable the designer to instanciate the model.



- ***Data model***

  Working on the 'web of types' allows to specify an abstract data model of the hypertext, regardless to its actual content. Classes are used to structure information, dependencies between classes can be specified in terms of inheritance as well as in terms of aggregation. Possible actions on information classes can be specified as methods and their semantics specified in scripts. Since a lot of works have already focused on the interest of object-orientation for data modeling, it is not necessary that we develop this point any more.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

- ***Navigation model***

  The strong typing mechanism of MacWeb can be used to specify an abstract navigation model, regardless to its actual anchoring. See [12].

- ***Implementation level***

  Since the abstract model description takes advantage of the MacWeb typed nodes metaphor, it is easy to attach the implementation model to the navigation model and to the data model. The implementation of a class is provided by the content of the nodes describing it : an active background image is computed for each node according to its class. This image is the 2 1/2D composition of the contents of nodes describing its class in the inheritance graph.

  The image is active. It supports the anchoring which is defined in the model. Thereby, factorizing visual elements between classes is easy. Inheritance allows to define links which are identical in several nodes only once by just placing them in the node describing the class or in any of its ancestor classes. Similarly, computed links can be described as methods once for all in the model and inherited, with possible overriding, according to the inheritance tree. Overriding allows to tune up the semantics of a generic link according to the actual class since method names are denoted by link names, they can link to different scripts in different classes.

  Furthermore, multiple inheritance makes an elegant separation of the logical structure and of the presentation structure possible.

  The anchoring of links can be overridden in the actual node instances. In the following example, the generic navigation model (see figure 2) specifies an 'invalid answer' link and a 'correct answer' link. Their anchors are left unspecified in the implementation model (figure 4). They are specified only within the instances (e.g. on figure 5, by assigning them to the gate pin, the drain pin, the collector pin). Conversely, the link 'ToExit' is fully specified in the model: its anchor and the look of its button are defined in the navigation specification of 'any Node'. It is the same for the 'Next question' link, which is specified as a method of 'question' (see figure 3).

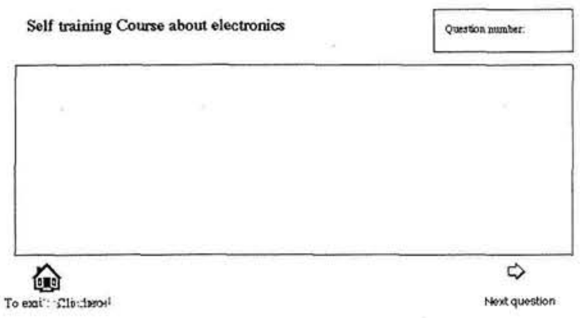


Figure 4: Specifying the look and the behavior.

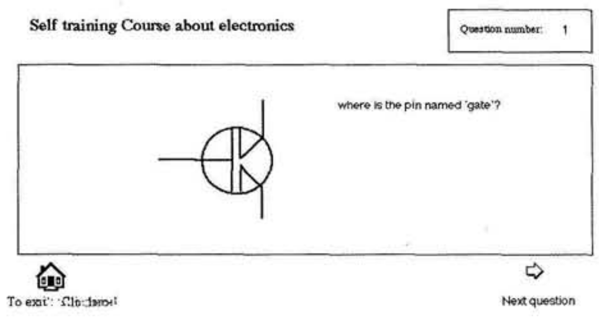

Self training Course about electronics
Question number: 1
where is the pin named 'gate'?
To exit: [illegible]
Next question

Figure 5: the actual look of a given instance node.

Figure 6 shows a web which has been produced by iterative instanciation of a navigation model. Observe the last node whose name is still '?The next one'. As any 'question' node, it owns a 'New Question' method (see figure 3) which triggers a new instanciation of the model and goes on polymerizing the question chain. This method is typically dedicated to the course developer. It illustrates the great flexibility of the design environment.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

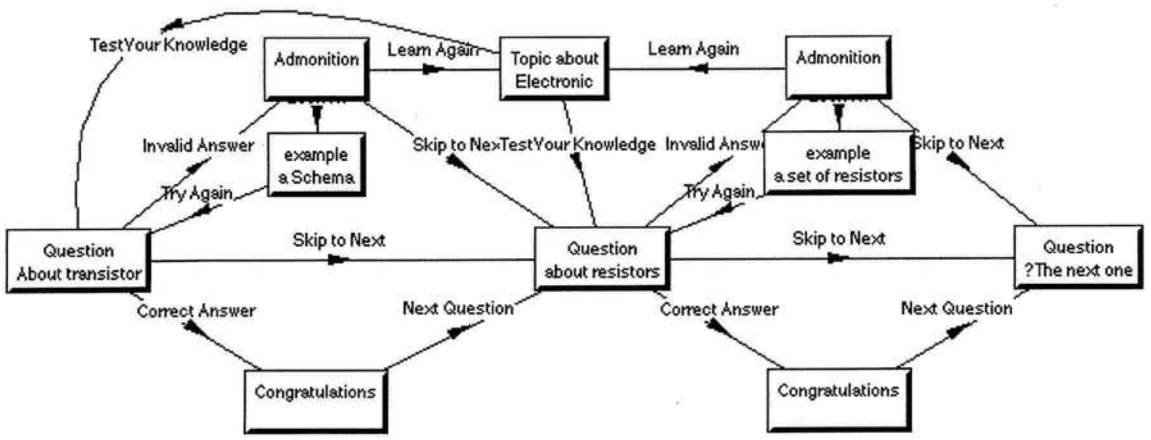


Figure 6: In the main web, the pattern resulting of the instanciation of the abstract navigation model is visible.

3.3 Incremental and opportunist design of the hypertext structure

Now, we consider key features that a design environment should provide in order to support also the second axis of the design space - the design process - mainly the incremental and opportunist design of the hypertext structure. However, any hypertext design environment has to help building a strong structure that is clear for the reader. Thus, beyond the feeling of 'no constrain' that the system should provide, it must also friendly help the designer to structure his work and make good structures emerge. For that purpose, a hypertext design environment should provide tools which make strong and reusable structure emergence easier than developing messy structures. So, it must support the abstraction process of building a hypertext structure as a structure emergence process mixing bottom-up and top-down approaches.
Such techniques, oriented towards structure emergence, are currently well tuned and used in information gardening approaches [9][14]. The idea is that they can be very efficient too when correctly adapted to information manufacturing approaches, especially for building a hypertext explicit structure during design. Let us remark also that these techniques present the great advantage of being independent of any particular design method and respect the inherent nature of human cognitive processes [13].
We claim that the following key points should be supported in a hypertext design environment:

- Easy and unconstrained prototyping tools directly available within the design environment for supporting the *generating material* sub process,
- Fast feedback loop to make the *evaluating* sub process easier and thereby improve the design quality,
- Operations for mixing bottom-up and top-down approaches in order to promote the *reorganizing and updating* sub process.
- Supporting modularity in design to improve the *organizing and structuring* sub process.

These points are discussed and illustrated with MacWeb.

3.3.1 Easy, unconstrained and integrated prototyping possibilities

By essence, hypertexts are concerned with human-computer interaction. On this aspect, they cannot be assimilated to mathematical entities on which formal reasoning is possible. Bernstein has pointed out the complexity of structure perception by the reader [2] and the difficulty to correlate it to formal properties of the structure. No valid theory nor technique is currently able to formally assess that a conceptual model of a hypertext is clear for the readers. Only testing the hypertext in real use situation is possible. This lack of formal tools is a handicap for the designer who operates blindly unless he has experimental feedback during the design. Since hypertexts are produced to be used exclusively by humans, the user's satisfaction is the main quality criteria.
As a consequence, it is necessary to:

- be able to build without constraints of models various solutions which may be directly evaluated whatever the step on the method axis. This aims at trying solutions.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

- integrate prototyping in the global process and the corresponding tools in the hypertext design environment in order to immediately evaluate consequence of design choice at any levels, the conceptual level as well as other levels such as presentation or navigation.

Light prototyping is strongly suitable for improving the quality of hypertext design. We put the stress on the need for direct evaluation of the effects of design choices, in real use situation and with a tool as close as possible to the final one. Furthermore, using a homogeneous formalism at any levels improve switching from one level to the other.
Prototyping should not be *constrained* by the design method. It should be supported as exception to the method. Precisely, its purpose is to allow the designer to check an idea, a change,... without needing to change the model he or she is currently working on. Thus, the prototype should:

- be created, edited, manipulated at will, without needing the preexistence of any given model,
- directly take advantage of any existing structure, those existing elsewhere in the prototype as well as in the models (data model, navigation model...),
- it should also support the process of structure emergence, i.e. allow to generate or update the model according to some specificity of the prototype structure.

The unified interaction style of MacWeb allows to create and edit at will any structure. There is no difference at all between operating on a node which belongs to the model description or on a node belonging to a prototype. So, the designer may work on the modeling as well as on a prototype. Due to the interpretative approach, any changes in the model are effective when the next command is issued. This allows a very direct and efficient prototyping.
For instance, suppose that the designer forgot to provide with an exit each node in the navigation model given on figure 2. By testing the design on an instance, he or she observes the need and the lack of the 'To exit' link. By taking advantage of inheritance for specifying it, as shown on figure 3, the link becomes instantaneously present in the prototype without needing any other actions. The designer can now go on testing the updated prototype.
Operations provided for supporting the incremental and opportunist aspect of the design process enforces also prototyping facilities (see section 3.3.3 and [13] for detailed operations).

3.3.2 The Fast Feedback Loop of MacWeb

Light prototyping is suitable for improving the quality of hypertext design. The designer may iteratively use the hypertext prototype to evaluate the consequences of structure design choices and take advantage of frequent feedback to improve and refine the model. Thus, switching between modeling, instanciation, production and evaluation activities spaces is needed. The friendliness of the feedback loop is a critical parameter for the efficiency of an incremental tuning of the structure design.
Switching time between activities spaces often is a handicap to efficiently mix complex activities. The user may loose the focus due to a cognitive load on the short term memory when switching is not direct. Thus, reducing switching time is fundamental to enable efficient feedback. Handling the design in the large and the design in the small with separated tools perhaps enforces a better clarity but conversely get the drawback of a less efficient feedback. As a consequence, it is suitable that effects of changes in the conceptual model be observed very quickly at least on a prototype of the hypertext. Integration of structure design tools within the hypertext authoring environment, or at least the integration of a light-prototyping tool within an external design environment, is highly suitable.
MacWeb relies on a unified interaction style in which both the models and the instances are manipulated in the same manner. The designer may shift continuously from the data model, to the navigation model, to the implementation model and to the actual hypertext. Furthermore, its interpretative approach makes it possible to reflect immediately any change in any part of the model on the hypertext itself. Thus, a very fast feedback loop is possible to tune-up the model according to observation of the actual use of the hypertext.
An other important and original issue in MacWeb is the role of visual representation of structure. It facilitates *observation* and *evaluation* of any structures, those which slowly emerge from the designer's activity as well as already known ones. Visual observation is not limited to nodes content. Visual representation of structures is also very efficient to help the designer catch at a glance regularities and irregularities, symmetries, relationships and so on....

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

MacWeb allows direct manipulation of the hypertext network by providing interactive manual layout. Both the class structure and the instance structure are drawn and may be edited on the graphic view of the web. The node and link placement may be controlled by authors who organizes the structure spatial representation at their will. The interest of 'perceiving the intended structure in space, just by noticing the geometrical relationships…' as been stressed by Marshall in [10].
Thus, sub-structures which are freely built on instances may be observed in a natural way. They appear as visual patterns, which, when identified, may lead back to emergence of abstractions in the author's mind and so far to emergence of some conceptual model. But, specific tools can also take advantage of the computabilty of the model to present more formal aspects of the data or of the structure.

3.3.3 Mixing bottom-up and top-down approaches

Most of methods claim that their rigorous stepping directly leads in a backtrakless manner to the specification of each stage of the design, Nevertheless, observation shows that most of structure starts emerging in the designer's mind by mental process based on ideas association and similitude recognition, which involve in a tango like process both bottom-up and top-down steps. Very often the designer 'sees" how his design will look like, even in an abstract manner. Turning the mental view into an abstract model often starts as a bottom-up structuring and is followed by a top-down expression of the result.
When a design environment aims at really helping the design process from scratch, and not only to allow reporting it with a rigorous formalism it is important to support the free alternation of these tow kinds of approaches.
MacWeb provides the designer with a set of three features which are useful for improving the design of complex structures:

- 'Prototyping' i.e. freely and directly cloning any sub-graph selected as prototype.
- 'Abstracting' i.e. making any selected sub-graph as a single entity and identifying it as a model,
- 'Instanciating' i.e. generating structures consistent with a given model

• *Prototyping*

Prototyping[3] is the ability to take any part of hypertext network as a prototype to generate a new structure isomorphic to the prototype. Prototyping is useful since designers often try several structures on instances before abstracting the good one. The designer select within a messy draft the substructure which seems pleasant and 'clones' it, regardless to the preexistence of a model of it. This ability of starting the work without model, is important in terms of the efficiency of the design process. The resulting structure emerges because it has been observed as significant, not because it has been claimed so at an abstract level before testing.
In MacWeb, once a set of nodes is selected, the clone operation becomes available. It generates a structure isomorphic to the first one, in which the nodes content (anchoring included) is duplicated. Links to or from outside the selection are duplicated to the new homologous nodes. (see figure 6). The names of the new nodes are deduced from the previous one with a '?' prefix, allowing the designer to recognize them and to set the new names. It is a very simple and direct operation which enable the easy generation of sequences of regular patterns.

• *Abstracting*

Abstracting is the ability of considering as simple a complex entity and naming it. MacWeb provide a grouping mechanism, which allows the handling of sub-structures. A group is node whose content is a sub-web. As a node, a group has a name. Thus, it is easy to create a group to refer to a sub-web and use it as named structure prototype. When a designer wants to turn a pattern into a model, he or she clones it first, then eliminates the instance specific data from it in order to keep only general data and finally groups it to use it later as a named model. This is typically a bottom up approach in which a model emerges from a set of instances.

[3]in the sense of classless objects named prototypes.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

• *Instanciating*

Instanciating is the ability to generate a structure according to a model. This is a basic functionality of MacWeb, and several approaches of instanciation have been experimented. The most appreciated one seems to be the cloning of prototypes.
A web which has mainly been produced by instanciation of several kinds of templates is an assembly of recognizable patterns which are similar to chemical radicals in a formula. Most of them are local replications of named templates. And, similarly to a polymer, the web does not grow anarchically: it has active sites where specific radicals (templates clones) can fit.

3.3.4 Modularity

Along the design process, choices about node size and content nature may vary. Modularity helps humans to deal with large or complex structures by decomposing them into smaller ones. It provides mechanisms for working on small structures and merging them later. In software engineering, the complementary process of modularity is link edition between modules. Similarly, we need to merge hypertext modules to built up large hypertexts.
In MacWeb, *node fusion* and *node splitting* are original and helpful features for interactively editing a web. Fusion aims at gathering into a single node information and links which are distributed in several nodes. Splitting a node allows to produce two nodes from one, for instance to divide an imported document into pieces. The split limit is expressed by the current selection. Both features preserve links and anchors.
A hypertext module allows the author to focus on a smaller set of aspects of the whole hypertext. This is more efficient from a cognitive point of view. This also allows separate teams to independently work on the design of complementary parts. Merging hypertexts modules is a complex operation since many links may exist between modules. MacWeb offers a very simple solution. When modules are merged, the name driven fusion is triggered: nodes with the same name and the same type in different modules are unified, as shown on figures 7 and 8. Their contents are concatenated into a single node with preservation of links and anchors.

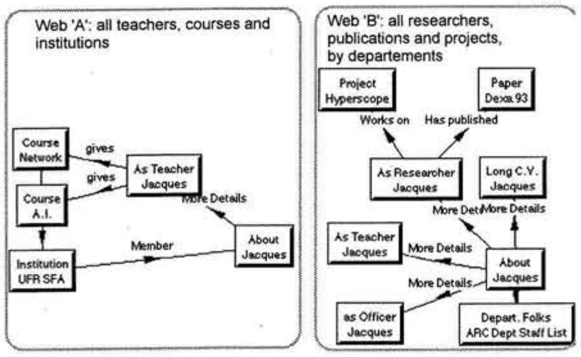


Figure 7: *Modularity*. Two hypertexts 'A' and 'B' are handled separately. But many persons are both researchers and teacher.

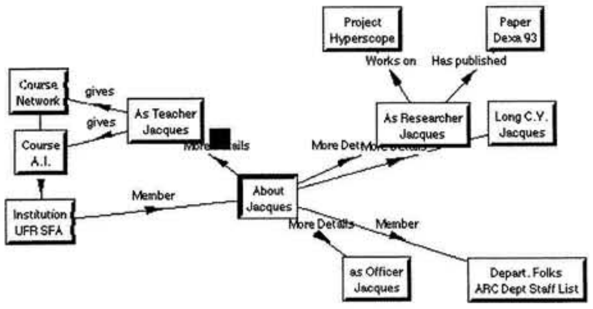


Figure 8: *Web fusion*. The two hypertexts A and B are merged. Nodes with the same name and the same type are unified into a single one.

So, each author may work on a module with little conflict with other authors. Even when they share nodes, each of them operates only on the part of the node he is responsible for. The merging mechanism does not require an explicit entry/external description of the modules interconnection. The name identity mechanism is simple and efficient, with respect to the sharing of a naming protocol by the authoring team.

5. CONCLUSION

Arguing for bottom-up, incremental structure elaboration and late deep changes may look paradoxical when most of authors strongly argue for modeling, structure description languages and even for schema driven compilative approaches. It is not a paradox, it is a more precise observation of the actual behavior of a designer at work. The incremental design processes are a major characteristic of any creative activity. For instance, studying a highly structured classical painting with X rays soemtimes makes visible the reliefs of the design process; it makes obvious the interest of trying and backtracking in the design process for producing strong strcutures

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Even when a method guides the designer along a schema elaboration, he or she may enjoy experimental feedback, help to concepts emergence, and so on. We propose to allow the designer to take advantage of the hypertext system to reflexively use it as an integrated interactive draft instead of scribbling hidden design steps on scratch pads... before exhibiting the ultimate schema like a rabbit from a hat.
Anyhow, designers of hand crafted hypertext and users who make structure emerge are fully aware of the complexity of the design task and of the importance of feedback. Techniques and tools for helping the design process have already been tuned for them and have proved their efficiency. We claim that these features are not gadgets dedicated simply to information gardening but are quite general and helpful for helping any kind of design.

6 REFERENCES

[1] P. Balasubramanian, T. Isakowitz, T. Stohr, "RDM a Methodology for the Design of Hypermedia Applications", in these proceedings, 1994.

[2] M. Bernstein, M. Joyce, D. Levine, "Contours of Constructive Hypertexts", *Proc. ECHT'92, 4th ACM Conf. on Hypertext*, ACM Press, Milano, pp. 11-22, Dec. 1992.

[3] M. Bernstein, "Enactment in Information Farming", Proc. Hypertext'93, Seattle, ACM Press, Nov. 1993, pp. 242-249.

[4] F. Garzotto, P. Paolini, D. Schwabe, "HDM - a Model for the Design of Hypertext Applications", ACM TOIS, 1993.

[5] F. Garzotto, L. Mainetti, P. Paolini, "Preliminary Considerations on the Evaluation of Hypermedia Applications", in these proceedings, 1994.

[6] Halasz F., "The Dexter Hypertext Reference Model", CACM, Feb. 1994, Vol.37 N°2.

[7] J.H. Hayes, L.S. Flower, "Identifying the Organization of Writing Processes", in *Cognitive processes in writing*, L.W. Gregg, E. R. Steinberg, edts, Lawrence Erlbaum Associates, Publishers, 1980.

[8] C.C. Marshall, F. M. Shipman, "Searching for the Missing Link: Discovering the Implicit Structure in Spatial Hypertext", *Proc. Hypertext'93*, Seattle, ACM Press, Nov. 1993, pp. 217-230.

[9] C.C. Marshall, F. M. Shipman, "Viki, Spatial Hypertext supporting emergent Structure", *Proc. ECHT'94*, Edinburgh, ACM Press, Sept. 1994, pp. 13-23.

[10] S. Michalak, M. Coney, "Hypertext and the Author/Reader Dialog", *Proc. Hypertext'93*, Seattle, ACM Press, Nov. 1993, pp. 174-182.

[11] J. Nanard, M. Nanard, "Using Types to Incorporate Knowledge in Hypertext", *Proc. Conf. Hypertext'91*, ACM Press, San Antonio, pp. 329-344, Dec. 1991.

[12] J. Nanard, M. Nanard, "Should Anchors be Typed Too? An Experiment with MacWeb", *Proc. HTX93, 5th ACM Conf. on Hypertext*, ACM Press, Seattle, Nov.1993.

[13] M. Nanard, J. Nanard, "Helping Emergence of Structures in Incremental Design of Hypertext", *Proc. Interface to Real and Virtual Worlds*, Montpellier, ISBN 2-2-906899-83-6, Mar. 1994.

[14] M. Nanard, J. Nanard, "Hypertext as a Tool for Information Gardening for Legal Applications", Informatica e Dirito, 1994 (to appear).

[15] S.R. Newcomb, N.A. Kipp, V.T. Newcomb, "The HyTime Hypermedia/ Time-Based Document Structuring Language, CACM 34(11), 67-83 Nov. 1991.

[16] V. Quint et al., "Combining Hypertext and Structured Documents in Grif", in *Proc. ECHT'92, 4th ACM Conf. on Hypertext*, ACM, Milano, pp. 11-22, Dec. 1992.

[17] D.E. Rumelhart, D.A. Norman, "Accretion, Tuning and Restructuring : three modes of learning" in *Semantic factors in cognition*, R.C. Anderson et al., Lawrence Erlbaum, 1978.

[18] D. Schuler, G. Rossi, "A design Space for Hypermedia Interfaces", in these proceedings, 1994.

[19] D. Schwabe, G. Rossi, "From Domain Models to hypermedia Applications. An Object-Oriented Approach.", in these proceedings 1994.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

[20] J.B. Smith, S.F. Weiss, G.J. Ferguson, "A Hypertext Writing Environment and its Cognitive Basis", *Hypertext'87 Proceedings*, Chapel Hill, North Carolina, ACM Press, Nov. 13-15 1987.

[21] N. Streitz *et al.*, "Sepia: a Collaborative Hypertext Writing System", *Proc. ECHT'92, 4th ACM Conf. on Hypertext*, ACM Press, Milano, pp. 11-22, Dec. 1992.

[22] M. Thüring, J.M. Haake, J. Hannemann, "What's Eliza doing in the chinese room? Incoherent hyperdocuments and how to avoid them", *Proc. 3rd ACM Conference on Hypertexts (HTX'91)*, ACM Press, San Antonio, pp. 161-177, Dec. 1991.

[23] M. Thüring, "A Conceptual Framework for Hypermedia Design Methodologies", in these proceedings, 1994.

[24] Y. Waern, *Cognitive Aspects of Computer Supported Tasks*, Wiley, 1986.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

A Design Space for Hypermedia Interface

Wolfgang Schuler

Integrated Publication and Information Systems Institute (IPSI)
Gesellschaft für Mathematik und Datenverarbeitung (GMD)
Dolivostraße 15, D_64 293 Darmstadt, F.R.G.
e-mail: schuler@darmstadt.gmd.de

1 Introduction

The task of constructing a hypermedia interface is a crucial aspect of the creation of large hypermedia applications. Because the interface directly affects usability and acceptance, its design is crucial for any hypertext. In fact, many researchers seem to regard it as the most relevant feature of hypertext at all: According to Nielsen (1990), the classification of a system as a hypertext system should depend more on the "look and feel" of its user interface than on its commands and data structures.

Prior experiences with this class of applications have shown that their design is a very complex task which requires dedicated methodological support. Since until now sufficient theoretical approaches have not been proposed for that purpose, a methodology for the design of hypertext and hypermedia should be developed which can be used to guide the construction of specific hypermedia interfaces and applications. In order to address the above issues, we organized a workshop (Schuler & Hannemann, 1993a) as part of our activities in the ESPRIT project HIFI (Hypertext Interface For Information: Multimedia and Relational Databases).

The goal of HIFI (HIFI, 1992; Cavallaro & Paolini, 1993) is to create a set of tools to allow readers to access a large body of information managed by external databases via a hypertext-based interface. HIFI adopts a model-based approach to hypermedia application design applying the Hypertext Design Model (HDM) of Garzotto, Paolini, Schwabe (1991). HDM is similar to the Dexter Hypertext Reference Model (Halasz & Schwartz, 1990). The hypertext-based interface is intended as a device to access existing databases, not to modify their content. The conceptual and visual structure of the interface should be determined by the readers' needs rather than by the structure of the databases. Thus, methodological issues of how to develop, maintain, and use such a hypermedia interface are of great importance.

Our objective is to articulate the design issues which have to be addressed in order to create such model-based hypermedia applications. These issues should constitute a comprehensive Design Space (DS) introduced by MacLean et al. (1991) which comprises the design issues for hypertext interfaces as an important part. By using HDM for the structural design of a hypermedia application, the design sub-space for hypermedia interfaces is not restricted by structural conditions related to a specific tool, but it presupposes fundamental structural concepts. An outstanding feature of hypertext is the tight coupling of interface components with structural components of the hyperdocument (Waterworth, 1990), because the structural components entail specific requirements for presentation and navigation.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

2 The Design Space Method

For developing the DS, we have used the IBIS method (Kunz & Rittel 1970) together with a commercial word processing system. This method has been used to capture the Design Rationale (DR) in many different contexts, such as architecture or software design, by allowing for the specification of issues, positions answering the issues and arguments supporting or objecting to the positions. Basics about the DR method, the DS and the IBIS method are described for example, in topic 'Design Process' of the HIFI DS on 'General Design Issues'. (See also Appendix 3 about the IBIS-Method)

Objectives

The main objective of this Design Space is threefold:

(1) First it shall function as a '*seed*' of design issues which are important in order to design a hypermedia interface (Fischer et al. 1992). The crucial main issues constitute the space 'dimensions'. Design alternatives formulated as possible answers (positions) to these issues are offered together with supporting and objecting arguments. Therefore, we have an argumentative DS. Additionally, many definition or factual issues have been added to the original design issues, or other issues which are not design issue in the original sense, i.e. issues referring to guidelines. An example issue: '*Should the system use browsers or cards?*' is shown in Appendix 2.

(2) Then, the DS can be used as the basis for developing and using an *application* d*esign IBIS* during the whole design process of a specific application to be developed. Having selected the issues relevant for a specific application and also after having decided about the offered alternatives, the subspace of the DS defined by this procedure will be the starting point. Already during this specification and decision process, additional issues, positions, arguments and facts will be raised and handled. These additional issues etc. can be added to this application design IBIS. And further, during the following design process this IBIS will be used in order structure the design and the necessary decisions, to document the rationale of the decisions and probably other open issues to be handled later.

(3) A third additional objective is a *research-oriented* one: As stated already above, this experimental starting DS is also conceived to gain more experience with DR methods in general and the conditions of how to organize, present and use a DS.

For example, a first very crucial issue is to show the usefulness of the whole approach and of a specific DS itself (by usability evaluation). It concerns the two fundamental claims made by argumentation-based DR approaches: that expressing DR as argumentation is useful, and that designers can use such notations. These claims are examined in a survey by Buckingham Shum & Hammond (1994) and their analysis of argumentation research sets an agenda for future research

Another important issue is how to gather relevant positions and arguments for a specific issue. This is not only time-consuming but it requires expertise to find the relevant empirical human factors results if they exist. Thus, this issue is concerning to the connection between design requirements and the various empirical findings of human computer interface

Some general crucial issues be solved are listed below:

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

How to organize a DS?

Which are the most important design issues to decide on?

How to prioritize these issues?

How to get the correct input (positions, arguments)?

How to find the usability of a DS?

Is the DS 'complete' (i.e., Is it sufficient for most applications)?

Are issues or answers missing? Are relevant arguments missing?

Is the DS competent? Are the answers or arguments correct?

Is its organisation and presentation useful?

3 The HIFI Design Space

As a basis we use a systematic theoretically_motivated set of issues of interface design for hyperdocument presentation and navigation (Thüring, Hanneman & Haake, 1993). They describe a theoretical framework which refers to hypertext readability and is based on the concept of coherence from cognitive research on text comprehension. These issues are extensions of the design questions described in "PHD" (Schuler & Thüring, 1992), a methodology for "Pragmatic Hypertext Design" which specifies design procedures as well as criteria and options for answering design issues. Another important source for input are the issues formulated at the methodology workshop.

In the present paper, we will primarily sketch a *practical DS for the Design of Hypermedia Interfaces.* We have divided the overall DS into three subspaces. Each of them is described by its most crucial issues with positions and arguments. Due to time and space limitations, we did not elaborate the issues in full detail, i.e., we specified relevant issues, but refrain from advancing all possible positions as well as any related arguments. Nevertheless, we believe that the resulting DS is a good starting point and may serve as a forum for further discussions and further developments.

The three Design Spaces focus on:

General design issues on hypermedia interface design
Browser based interfaces: hyperdocument presentation
Card based interfaces

Our main idea is, to use the DS methodology for artefact design in general, and specially for the design of model-based hypermedia applications. The hypermedia design model used in general will be the Hypermedia Application Design Model *HDM.* In the ESPRIT project HIFI, it will be *HDM+,* the revised and extended version of HDM. HDM+ addresses the design of a specific class of hypermedia applications, i.e. *hypermedia interfaces to data bases* (or, more generally , to information systems).

We have not represented the total hypermedia application development cycle in this DS. This design cycle has been described in HIFI deliverable D8 (Schuler, 1993) and also by Garzotto, Mainetti & Paolini in (Schuler, Hannemann & Streitz, 1994), first presented in the HIFI workshop (Schuler & Hannemann, 1993). But we think this should be done later, or at least it

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

should be tried to do so. However additionally, the DS for the total application development cycle would turn out to be fairly very large and complex. And it could not be handled effectively unless powerful hypermedia software is available which is specially dedicated to the IBIS-method used for the formulation of the DS.

Only the interface related part of hypermedia design has been realized. And also this topic has not been elaborated in every extent. Especially the interconnection of the Browser and Card System DSs should be refined (and probably further elaborated).. The reason is, that we first have to gain more experience with the usefulness of this IBIS based method. In this respect, the DS is still in an experimental condition. Thus, the space should be elaborated and used in real design conditions. There were no time and not sufficient resources to do this along with the HIFI applications. The results of such 'usability' testing will indicate how to reorganize and enlarge the DS and also which issues are of lower importance.

3.1 Organization of the Design Space

In order to have an introductory overview of the three DSs a list of the total issue trees is offered to the user. The objective of these issue trees is to function like an 'issue outline' or as a navigation outline if we had represented the DSs in a hypermedia system. With such a system, the designer as a user could identify an issue and then get the argumentation or the content connected to it. Or, he could explode or implode an issue tree using outliner functionality.

Based on this issue organisation the content of the total DSs can be accessed because its identical issue structure. However, it contains additionally all the information elements attached to the issues: their positions, positive and negative arguments on a position, and additional factual information like comments and figures. The Browser system and Card system are very much interconnected, that means there exist many pointers from elements of one part to elements of the other, and vice versa. But we did not elaborate the resulting network in detail and indicated it only in few cases, because we cannot easily represent these relationships in a linear form. An example of the issue structure for *Buttons* and *Guided Tours* is shown in Appendix 1.

3.2 The General Design Issues

This part contains important general design issues for hypermedia applications. These issues are not typical for hypermedia. However, we think that these issues - and may be even more - are important just because there does not exist too much documented experience in the development of hypermedia applications, especially large and complex ones. Many conceptual issues discussed in the contribution of Thuering (this volume) fall into this area.About these issues has to be dicided at the beginning of an application project. Exmples are:issues concerning the usage of this DS, the IBIS-method as an argumentative design method, the guidelines, and the checklist for heuristic evaluation which will be both descibed later. We have organized the general design issues into the following topics:

Design

- Design Process
- User Requirements
- System Requirements
    - Standard System Functions

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

- Usability Evaluation

Design Utilities

- Design Rationale IBIS
- The IBIS-Method
- Metaphors
- General Design Guidelines
- Examples of prototypical system interfaces

3.3 The Browser System: Design Space for hyperdocument presentation

This DS is based on the metaphor of a hyperdocument which is offered to a reader somehow in a similar way as a classical document. Compared to a normal document, a hyperdocument or a hypermedia document is characterized by its non-linear hypertext structure and by using multimedia. Although not directly implied by the used hyperdocument metaphor, another constraint of this DS is that the document structure can be shown with a *graphical browser system*. The function of a graphical browser for a hyperdocument can be compared with the function of an 'outline' for a traditional document. Additionally, like in a linear document, the author 'guides' the reader in a hyperdocument, and he has organized navigation accordingly by providing reading paths (guided tours) for the reader. Thus, the author has planned carefully the navigation and access possibilities of his document. And his main issues are

How to support the reading process? or, more general,
How to support the reader's information reception?

The construction of the leading issue structure for this DS is governed by two *principles* derived from a cognitive model of understanding (reading):

- Coherence of data elements (texts)
- Reduction of additional workload (navigation)

The basic information reception issue can be decomposed into three subissues:

I: *How to support information reception (reading)?*

I1: *How to create a coherent hyperdocument?*

I2: *How to convey coherence to the reader?*

I3: *How to facilitate navigation?*

This DS for browser-based hyperdocument presentation is based on the ideas of Thüring, Hannemann & Haake (1993)

3.4 The Card System

The 'card'metaphor is well-known and is used for example in HyperCard or ToolBook. It is the basic metaphor of this DS part:navigation is realised by jumping from card to card and these syetems do not offer in general a dynamic browser.

We have organized the design issues into the following topics, which have also been used for the Design Guidelines and the Checklist described later (Schuler & Hol, 1994):

General aspects

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

- System Structure
- Interaction Styles

Window Organisation and Screen Design

- Menus
- Buttons
- Forms
- Boxes
- Other Interaction Styles: Touchscreen, Pen
- Special Interaction Issues

Presentation

- Node contents
- Naming
- Color
- Icons
- Typography
- Multimedia
- Text
- Graphics
- Sounds
- Video
- General Aspects of Multimedia
- Multimedia in Combination.

Navigation and Browsing

- Access Structures: Guided Tours and Indexes
- Guided Tours
- Indexes
- Crossing between Several Access Structures
- Navigation
- Searching

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

3.5 By-products of the Design Space

On the other side, we used the issues, positions, and arguments of the DS to develop

- **Guidelines for the Design of Hypermedia User Interfaces** and a
- **Checklist for the Evaluation of Hypermedia User Interfaces**

Being part of the DS they are documented as separate products in Appendix D and E of the DS description (Schuler 1994). And they are now available separately (Schuler & Hol 1994). We did not indicate at the elements of the DS where all the Guidelines are derived >From or are connected with nor did we try to deduce the Guidelines and the Checklist systematically from the DS.

Both Guidelines and the Checklist have been given to the partners who are concerned with the HIFI applications, in order to use them. They were asked to make comments or amendments. The Checklist has been tested with several applications. It has been used successfully to describe and characterize the demonstration application 'Art Gallery' (Microsoft). It has also been used to test the pre-versions of all three HIFI applications. In another test the Checklist has been applied to a version of the 'HyperMilano' application developed by Politecnico di Milano. However, we had not available the latest final versions of the HIFI applications. The Checklist was finally used for the heuristic usability evaluation of the HIFI applications by the partners (the evaluation issue is described in the topic 'Usability Evaluation' of the DS on 'General Design Issues').

3.6 Demonstration applications and screendumps

Another important part of the DS constitute additional facts which are also shown in the Appendix A-C of the DS document.. These facts would be natural parts of a hypermedia realization of the DS and could be reached from the corresponding issues to which they are connected. These additions are:

Demonstration applications
They are examples of prototypical hypermedia interfaces and are intended to be stimulate the designing activities. Some of them are commented along the criteria of the Checklist.

Screendumps of other applications with short comments.

Standard design objects for ToolBook Version 1.5
It is a characterization of the available design objects with their attributes in form of lists. We did this as an example for the card-based hypermedia authoring and presentation system ToolBook because the HIFI applications are realized with it.

3.7 Main design issues for both Design Spaces

One outstanding feature of hypertext is the tight coupling of interface components with structural components of the hyperdocument (Waterworth 1990), because the structural components entail specific requirements for presentation and navigation. For example, a hyperbase which consists of different layers, requires presentation formats and navigation facilities which differ from those of a flat hypertext net, i.e., it must enable the user (as a reader) to assess information at different layers and must ensure his traversal from one layer to another.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

The overall goal of each designer of a hypermedia application is to satisfy the information needs of his readers. Depending on their needs, users (as readers) may engage in three different kinds of activities:

1. **Information scanning** aims at deciding whether a document contains any information of interest.

2. **Information search** aims at finding specific information that is relevant with respect to a particular interest or task at hand.

3. **Information reception** aims at comprehending the complete document or parts of it selected after information scanning or information search.

This leads to a first separation of the DS into three main issues:

How to support information needs of a hyperdocument reader

  How to support information scanning

  How to support information search

  How to support information reception

The DS for browser based interfaces focuses mainly on the issue of how to support information reception. And the DS for card based interfaces focuses more on the information search and scanning issues.

4 Conclusions

We use here a *hierarchical IBIS* where the indentation of the issues indicates their hierarchical level. This has mainly been done in order to use a word processor like Yakemovic & Conclin (1990) did in their experiment. Only in this way a commercial text processing system can be used. However, by this procedure the generation, updating and later presentation of a larger DS is very troublesome and by no means elegant. Furthermore, having a larger DS it turned out to be harder and harder to get an overview of the Space as well as to use it favorably. The need of a powerful hypertext / hypermedia system is obvious. However, we had not available a suitable hypermedia system running on PC.

And we think that *ToolBook* is not applicable for the purpose of generating a DS because it is not flexible enough. May be an adaptable outliner would be useful which can be configured according to user needs. However, a card-based system as ToolBook can be used with an existing, already developed DS. Just this has been now realized by Siemens: they implemented our DS as a hyperdocument on ToolBook 1.5 as a test version called '*Methodological Handbook*'. It offers a card-based presentation as a fixed 4-level hierarchical issue structure with different link types, index and retrieval functions as well as a history function. The handbook has integrated additional text about *Methodological Aspects of Hypertext User Interface Design* (Schuler 1993b). The final hyperdocument has a total number of 1276 pages with 103 figures. This handbook application can then be used for further evaluation and testing as well as for further development of this DS approach.

Further, we are just going to adapt the functionality of GMD's cooperative hypermedia authoring environment *SEPIA* (Streitz et al. 1992) for the purpose of DR Capturing. One of its

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

components, the 'Planning Space' and its corresponding graph browser, is already based on the IBIS-method. For the purpose of DR a single user version of SEPIA running on PC is useful, while SEPIA normally runs on Sun Sparc stations. The SEPIA prototype enables the collaborative generation of Issue Based Design Documents (IBD), i.e. to make collaborative entries into such a IBD. It is being further developed as DOLPHIN to provide support for face-to-face and distributed meetings (Haake & Streitz, 1994). We hope that soon a robust basic version of SEPIA will be available for interested test users.

5 References

S. Buckingham Shum & N. Hammond (1994). *Argumentation-based design rationale: what use at what cost?* Int. Journal of Human-Computer Studies. 40 (4), 603-652.

U. Cavallaro & P. Paolini (1993). *HIFI: Hypertext Interface For Information - Relational and Multimedia Databases.* The Electronic Libraries, Vol 11, n° 2, Apr. 1993, p. 65-72

G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves & F. Shipman (1992).*Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environments*. Human-Computer Interaction, Vol.7/3, pp. 281-314

F. Garzotto, P. Paolini, D. Schwabe (1991). *HDM - A model for the design of hypertext applications.* In Proceedings of the 3nd ACM Conference on Hypertext (Hypertext '91), pages 313-328, San Antonio, Texas, December 15-18, 1991.

J. Haake & N. Streitz (1994). *Coexistence and Transformation of Informal and Formal Structures: Requirements for More Flexible Hypermedia Systems*. Proceedings of the European Conference on Hypermedia Technology (ECHT'94), Edinburgh, September . 18-23, 1994, ACM Press, New York.

F. G. Halasz, M. Schwartz.(1990). *The Dexter Hypertext Reference Model.* Proceedings of the NIST Hypertext Standardization Workshop. Gaitherburgh, MD, January 16-18, 1990.

HIFI (1992). *HIFI: Hypertext Interface For Information: Multimedia and relational databases.* Technical Annex of the ESPRIT Project 6532.

W. Kunz & H. Rittel (1970). *Issues as elements of information systems.* Working Paper 131, Berkeley, CA: University of California, Center for Planning and Development Research.

A. MacLean, R. M. Young, V. M. E. Bellotti & T. P. Moran (1991). *Questions, Options, and Criteria: Elements of Design Space Analysis*. Human-Computer Interaction, 6 (3&4): 201-250, 1991.

J. Nielsen (1990). *Hypertext and Hypermedia.* San Diego: Academic Press, 1990.

H. W. Rittel & M. Webber (1984). *Planning Problems are Wicked Problems.* In: N. Cross (Ed.) Developments in Design Methodology. John Wiley & Sons Ltd, Chichester, 1984, pp. 135-144.

W. Schuler & M. Thüring (1993). *Pragmatical Hypertext Design (PHD)*. GMD Report 813. January 1994. Based on ESPRIT Project 5252 (HYTEA), Technical Report. GMD, 15. 3. 1993.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

W. Schuler & J. Hannemann (1993). *Workshop on Methodological Issues on the Design of Hypertext-based User Interfaces.* GMD-IPSI, July 13-14, 1993.
HIFI Project Report D8.2. ESPRIT Project 6532. To be published (see below).

W. Schuler, J. Hannemann & N. Streitz (Eds.) (1994). *Designing User Interfaces for Hypermedia.* Springer Publication, Heidelberg; ESPRIT-Series (in publication).

W. Schuler (1993). *Methodological Aspects of Hypertext User Interface Design.* HIFI Project Report D8. . GMD-IPSI, June 18, 1993.

W. Schuler (1994). *Design Space for the Design of Hypermedia User Interfaces.* HIFI Project Report D7. ESPRIT Project 6532. GMD-IPSI, June 15, 1994.

W. Schuler & J. Hol (1994). *Guidelines and a Checklist for the Design and Evaluation of Hypermedia User Interfaces.* Technical Report. Arbeitspapiere der GMD Nr. 866. September 1994.

N. Streitz, J. Haake, J. Hannemann, A. Lemke, H. Schütt, W. Schuler & M. Thüring (1992). *SEPIA: A cooperative hypermedia authoring environment*. In D. Lucarella, J. Nanard, M. Nanard, P. Paolini (Eds.), Proceedings of the 4th ACM Conference on Hypertext (ECHT'92), pages 11-22, Milano, Italy, November 30 - December 4, 1992, New York: ACM Press, 1992.

M. Thüring, J. Hannemann & J. Haake (1993). *Hyperdocument Presentation: Facing the Interface.* GMD Report 784. September 1993.

K. Yakemovic & E. Conclin (1990): *Report on a Development Project Use of an Issue-Based Information System.* CSCW '90 Proceedings, pp. 105-118.

J. Waterworth (1990). *Hypermedia interfaces for hypermedia documents*. In A. Rizk, N. A. Streitz & J. André (Eds.), Hypertext: Concepts, Systems and Applications, (Proceedings of the European Conference on Hypertext, ECHT '90, Paris, France, November 1990), pages 356-358. Cambridge: University Press, 1990.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Appendix 1: Example Issue Trees on *Buttons* and *Guided Tours*

Buttons

How to design buttons?

- *What is a button?*
  - *Button specifics in ToolBook?*
- *Which button types exist?*
- *How many buttons should be used?*
- *How to design a button that the user can spot it?*
- *Where to put how many buttons of which kind?*
- *What to do that the user can trust a button?*
- *Which buttons shall be used? (What are the wants the user?)*
  How can the designer tell what the user wants?
- *Embedded buttons in the text or buttons separated from text?*
- *Size, location, and form of the buttons?*

Guided Tours

How to design a guided tour?

- *What is a guided tour?*
  - *Purpose of guided tours?*
  - *Structural aspects of guided tours?*
  - *Navigational aspects of guided tours*
  - *Types of guided tours (paths)?*
- *Which moves are appropriate in which situation (browsing semantics)?*
  - *Which standard moves are appropriate in which situation?*
  - *Which additional moves are appropriate in which situation?*
- *How to present and organize guided tours?*
- *How many guided tours should the application have?*

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Appendix 2: Example Issue

Should the system use browsers or cards?

Position 1: *Cards*

Pro Arguments:

+ There are programs available (e.g. HyperCard on Mac, ToolBook on PC).
+ Some of the available programs are high-level languages (i.e. Hypercard, ToolBook).
+ Available programs take away design choices. The makers have thought through some design issue. This makes it possible to concentrate on the application domain because one does not have to worry about a lot of basic decisions
+ Available programs have produced standard conventions.
+ With one card, one does not have the problem of window management overhead.
+ No browser overhead
+ Cards are very natural and recognizable for users. Hypermedia structures and browsers are not natural, nor familiar for users.
+ Cards are easy to use for metaphors. For example pages in a book, a pile of photos, etc.
+ The chance that a potential user has experience with, or is familiar with a card-based system is higher, than that she has ever come in contact with a system that uses browsers.
+ Most operating, available systems use cards.

Position 2: *Browsers*

Pro Arguments:

+ There are some programs available (e.g. Storyspace on Mac, Sepia prototype on Sun).
+ When the data structure is part of the information that is presented than it makes sense to use browsers. An example is the SEARLE application described in Thüring, Hannemann & Haake (1993).

Contra Arguments:

- Operating a browser is a complex task for a user, that requires insight in the structure of a hypertext document.
- Learning to use a browser or even multi-browsers cost effort.
- Browsers have difficulties with complex and big structures. Hypertext structures are per definition complex and big.
- Browsers are suited for the computer literates.
- Browsers show data structures and not semantic structures.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Appendix 3: The IBIS-Method

What is IBIS?

IBIS (Issue Based Information System) was developed by Rittel (Kunz & Rittel 1970). It is an argumentation method using controversial issues and relations between these issues as basic elements. Its main objective is to construct a network (or in most cases a quasi-hierarchy) of issues (design questions) to be resolved in an issue deliberation process. Positions (as answers) are generated and later selected and rejected in order to solve an issue; and arguments are supporting and objecting these positions. Positions as well as arguments can normally also have sub-positions and sub-arguments. Also issues can be refined by sub-issues. To issues, positions, and arguments various material (text, graphics, drawings, tables, spreadsheet, etc.) can generally be attached as facts.

Theoretical basis of IBIS ?

IBIS was suggested by Rittel (1984). He was the first to advocate systematic documentation of DR as part of design. He sees design problems as fundamentally open ended and controversial in the sense that there are no objective criteria for closing problem definitions and settling disagreements. Such closing and settling are necessary for design, but they are determined by practical constraints like time and efforts, not by reasons of the design process itself.

Objectives of IBIS?

- Argumentative supported design kid
- DS: a basis for DR capturing

Structural elements (node types) of IBIS?

The original IBIS is based on the following structural elements carrying normally text (in hypertext realized as node types):

- issues
- positions
- arguments
- facts

Basic Relations between the structural elements (basic link types) of IBIS?

• an issues	'*serves*'	another issue (sub-issue)
• a position	'*answers*'	an issue
• an argument	'*supports*' or '*objects-to*' a position	
• a fact	'*references*'	an issue, a position, an argument
• a position	'*contributes*'	another position (sub-position)
• an argument	'*contributes*'	another argument (sub-argument)

RMD: A Methodology for the Design of Hypermedia Applications

P. Balasubramanian, Tomas Isakowitz and Ted Stohr

ABSTRACT

We describe a step-by-step methodology for the design and construction of hypermedia applications and illustrate our approach using experience gained from building several applications. The Relationship Management Design (RMD) methodology begins with a data model of the application domain and proceeds through the design of the hypertext network, user interface and run-time dynamics finally concluding with the construction and testing of the target hypermedia system. Our ultimate objective is to use the RMD approach as the basis for the construction of computerized tools to support the design and development of hypermedia applications.

1 - INTRODUCTION

Hypermedia design for commercial purposes, e.g., technicaldocumentation, catalogues, encyclopedias and training manuals, is ataxing task that usually results in large and complex applications.These applications must be of high quality and easy to maintain.Moreover, their development must be timely and often has to beaccomplished with limited resources. Although there is a richliterature on on hypermedia [4][3] and its applications, there arerelatively few research efforts addressing the design methodologyissue [14][9]. As a consequence, hypermedia development projects areunduly lengthy and hard to manage, frequently producing systems thatare also difficult to handle and to maintain.

In this work we propose a methodology for designing and constructingorganizational hypermedia applications. By design methodology we meana systematic procedure to design and build hypermedia applications. Asdescribed in [2] sound design methods bring discipline to thedevelopment process and are comprised of the following: notation (toestablish means for communicating), process (to guide developers) andtools (to ease the, often tedious, work).

We view hypermedia as one means to manage relationships amonginformation units. Hypermedia can help in managing certain aspects ofsome of these relationships, but not others. Specifically, hypermediasupports navigation and exploration activities, and underperforms incalculation intensive tasks. It is in this sense that we considerhypermedia as a means of managing relationships. Hence the names forour data model and for our methodology,

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

relationship management designmodel (RMD) and relationship management design methodology (RMDM).

2 - RELATED WORK

There is surprisingly little research on designing hypermediaapplications. Although several data models have been proposed in theliterature many of them deal with the concept of hypertext per se,i.e. what constitutes a hypertext system, as opposed to the design anddevelopment of hypermedia applications.

In the first category, data models for hypertext systems, we find forexample, the Dexter model [10], Tompa's hypergraph data model [20],Akscyn's KMS [1], Schnase's Semantic Data Model [20], Furuta TrellisModel [7] and Lange's model [lange-90] all enter into the firstcategory.

Our contributions falls in the latter category, namely the design anddevelopment of hypermedia applications. Most contributions in thisarea provide data models and in some cases, design guidelines, but norigorous methodology. Hannemann, Thuring and Haake [12], for example,propose design guidelines for structuring a hypertext network and forbuilding its user-interface to support comprehension of hyperdocumentsat a cognitive level. HDM [10] and HDM2 [8], are data models meant tocapture domain entities and relationships, as well as a limited rangeof navigation methods. Their approach serves as the basis for our datamodel. There have also been some developments based on anobject-oriented approach, such as Lange [15] and Schwabe and Rossi[21]. Although, to some extent, all of these approaches also provideguidelines for design and development, none of them incorporates adetailed step by step methodology that contemplates all aspects ofdesign and development, as we do here.

4 - RELATIONSHIP MANAGEMENT DATA MODEL (RMD)

The relationship management data model (RMD) we propose is based onthe HDM data model [10] and on its successor, HDM2 [8].

HDM consists of structural, applicative and perspective links. Inaddition, HDM2 provides for three kinds of access structures: indices,guided tours, and groupings. Each access structure has its ownbrowsing semantics, that determine how navigation is to occur.

RMD extends HDM by enabling parametrization of links, indices andguided tours. Whereas in HDM, an index or a guided tour is eithermanually created to include specific instances of an entity, containsall instances of an entity, in RMD indices and guided tours have aparameter, a condition, which allows designers to specify whichinstances of an entity are to participate. For example, in ahyperdocument about an academic department, one can specify a link>From each faculty member to an index of all the courses s/he teaches.The point is that the index is different for each faculty member. Ournotation helps automate the process of building indices.

5 - RELATIONSHIP MANAGEMENT DESIGN METHODOLOGY (RMDM)

The design methodology we propose consists of seven steps listedbelow, which are to be iterated as needed.

Step-1: E-R Design

We first determine the aspects of the domain that are to be supportedby the application and represent these with an E-R diagram [6]. Sincemany designers have experience with E-R diagrams we can build on suchexperience to provide a familiar environment for the design ofhypermedia documents. Following the HDM nomenclature we call theserelationships ``applicative''. The E-R diagram does not indicate theways in which users may navigate through the information space; ratherit is a design of the entities and relationships among them.

Step-2: Entity Design

The second step involves breaking up entities into components to betreated as individual nodes by the hypermedia application. An entitymay contain a lot of information. Breaking it up into self-containedinformation units and setting up relevant relationships among theseunits is a quintessential design requirement. We follow HDM notation,and call these relationships ``structural''.

Step-3: Navigational Design

In this step all relationships (applicative and structural) aretransformed into RMD access structures. These will ultimatelydetermine the extent of navigational support the hypermediaapplication is capable of providing. The outcome of this stage helpsprovide global structure for the navigation which is one of the designissues raised by Kahn et. al. in [24].

The outcome of this step is an RMD diagram, which depictsall entities, their components and all access structures tobe present in the application. Navigational design isimportant because it will help reduce disorientation[17].

Step-4: Conversion Protocol Design

The next step is to specify how the RMD diagram is to be realized inthe target hypermedia application platform. This is done via a set of ``conversion rules'' that translates the RMD diagram into a web ofnodes and links. A set of formal conversion rules is used to achievethe translation.

Step-5: User-Interface Design

This stage involves the design of screen layouts for every object inthe RMD model. As described by Kahn et. al. [24] issues like definingthe space on screen, dividing the page into content and orientationspace, communicating active and passive areas and item are relevanthere. This includes button layouts, appearance of nodes and indicesand location of navigational aids. User-Interface guidelines [18][19]and user-involvement are crucial in this step.

Step-6: Run-time Behavior Design

This step involves the design of the application's run-time behaviorby specifying the semantics of link traversal and how to obtain theinformation to be presented. It is at this stage that designersdetermine the dynamic or static nature of links and nodes. Traversalof a static link involves accessing the endpoint of an existing link,whereas traversal of a dynamic link involves extracting informationnot present in the link from other sources (perhaps from an outsideapplication, e.g. an on-line database) to compute a link's destinationnode. (The node in turn can also exhibit dynamic behavior)

Step-7: Construction and Testing

During this stage, which is not part of design, but of implementation,the hypermedia application is constructed by populating the derivednode-&-link data model with domain data, as by the run-time behaviordesigns. After construction, the application is tested and theprocess (from step-1 on) is iterated as many times as needed.

It is important to note that there are restrictions to the kinds ofapplications one can build with RMD. For example, ad-hoc relationshipsbetween individual instances of such entities are not supported byRMD. Although, this restricts the generality of our approach, itguarantees the correctness of each and every link in the application.Moreover, the class of structured hypermedia applications to which theRMD methodology applies is quite large, covering many businessapplications.

6 - BENEFITS AND SUMMARY

Our experiences with the RMD methodology indicate that it is helpfulin designing, building and maintaining structured hypermediaapplications. Its RMD data model enables the specification of animportant class of hypermedia applications, and can be used to developa standard across such applications. In sum, RMD developedapplications are robust, the links therein are 100% correct, usersbenefit from the consistency of their look an feel, and developers>From improved support for design and development activities.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

REFERENCES

[1] Robert Akscyn and Donald McCracken and Elise Yoder. KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. Communications of the ACM, 31(7):820-834, July 1988.

[2] Grady Booch. Object-Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Inc. 1994.

[3] Vannevar Bush. As we may think. Atlantic Monthly, pages 101-108, July 1945.

[4] Jeff Conklin and Michael L. Begeman. gIBIS: A Tool for All Reasons. Journal of the American Society for Information Science, 20(3):200-213, 1989.

[5] P.P. Chen. The Entity Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems,1(1):9-36, 1976.

[6] Ramez Elmasri and Shamkant Navathe. Fundamentals of Database Systems. The Benjamin/Cummings Publishing Company, Inc. 1989

[7] Richard Furuta and David Stotts. The Trellis Hypertext Reference Model. In Judi Moline, Dan Beningni, and Jean Baronas, Editors, Proceedings of the Hypertext Standardization Workshop, pages 83-93. Gaithersburg, MD 20899, March 1990. National Institute of Standards and Technology. NIST special publications 500-178.

[8] Franca Garzotto, Paolo Paolini, and Luca Mainetti. Navigation in Hypermedia Applications: Modeling and Semantics. Journal of Organization Computing.

[9] Franca Garzotto, Paolo Paolini, and Luca Mainetti. Navigation Patterns in Hypermedia Data Bases. In Proceedings of the 26th Hawaii International Conference on System Sciences, Vol. III, Pages 370-379. IEEE Computer Society Press, January 1993.

[10] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM: A Model-Based Approach to Hypertext Application Design. ACM Transactions of Office Information Systems,(1):1-26, January 1993.

[11] Frank Halasz and Mayer Schwartz. The Dexter Hypertext Reference Model. Communications of the ACM, 37(2):30-39, February 1994.

[12] Hannemann and Thuring and Haake. Cognitive Considerations in User-Interface and Navigation Design. Elsewhere in this issue.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

[13] Tomas Isakowitz and Edward A. Stohr. Hypertext-based Relationship Management for DSS. Working Paper IS-92-22, CRIS, New York University, IS Department, New York, NY 10003, 1992.

[14] Danny Lange. Object-Oriented Hypermodeling of Hypermedia Supported Infomation Systems. Proceedings of the 26th Hawaii International Conference on System Sciences, III:389, 1993.

[15] Danny Lange. Journal of Organizational Computing.

[16] R.P Minch. Applications and Research Areas for Hypertext in Decision Support Systems. Journal of Management and Information Systems, 6(3):119-138, Winter 1989-90.

[17] Jakob Nielsen. The Art of Navigating Through Hypertext. Communications of the ACM, 33(3):297-310. March 1990.

[18] Ben Shneiderman. Designing the User Interface: Strategies for Effective Human Computer Interaction. Addison-Wesley, Reading, Massachusetts. 1987.

[19] Ben Shneiderman and Greg Kearsley. Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information. Addison-Wesley, Reading, Massachusetts. 1992.

[20] John Schnase and John Leggett and David Hicks and Ron Szabo. Semantic Data Modeling of Hypermedia Associations. ACM Transactions on Information Systems, 11(1):27-50, January 1993.

[21] Schwabe and Rossi. OOHDM: An Object Oriented Hypermedia Design Model. Elsewhere in this issue.

[22] Frank Tompa. A Data Model for Flexible Hypertext Database Systems. ACM Transactions on Information Systems, 7(1):85-100, January 1989.

[23] E. Yourdon. Modern Structured Analysis. Yourdon Press, 1989.

[24] Paul Kahn and Ronnie Peters and Cindy Perthou. Presentation Design Issues for Hypertext and Multimedia Publications. Elsewhere in this issue.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

From Domain Models to Hypermedia Applications: an Object-Oriented Approach.

Daniel Schwabe and Gustavo Rossi [1]

Departamento de Informática
Pontificia Universidade Católica
R. M. de S. Vicente, 225
Rio de Janeiro, RJ 22453-900, Brazil
Fax: +55-21-511 5645. Telephone: +55-21-529 9544
E-mail: [schwabe,rossi]@inf.puc-rio.br

1 also LIFIA, Universidad Nacional de La Plata
La Plata, Argentina, and Conicet.

Abstract

In this paper we present an object-oriented method for designing hypermedia applications. The approach divides the development process in four steps, namely: domain (or content) design, navigational design, abstract interface design and implementation. We use similar modeling primitives (object and classes) and abstraction mechanisms (aggregation, generalization), during the whole process thus improving traceability; design decisions like the use of complex navigational structures are made explicit through a uniform notation thus allowing a coherent document structure that simplifies the construction of a CAHDE.

1 - Introduction and rationale

Building large hypermedia applications is difficult, and is further complicated by the fact that, once an application has been built, its maintenance is correspondingly more complicated. Moreover, as in other software domains (such as information systems, databases, etc.) hypermedia applications are usually built from scratch: reuse is still a dream. As it has been stated elsewhere [Garzotto91], hypermedia design models and in particular object-oriented models [Lucarella93, Lange94] allow the description of a hypermedia application using high level constructs in an implementation independent way. Step-by-step methodologies can then be defined on top of existing design models [Balasubramaniam 94]. However, design models are still in their infancy; design decisions are often taken at the wrong time during the development life-cycle or are poorly documented, thus difficulting evolution.
In this paper we propose a step-by-step method based on the construction of a sequence of object-oriented models that leads from domain analysis to implementation. Although based on object-oriented concepts, the resulting design can be implemented on top of a conventional (i.e., non object oriented) platform. The key ideas underlying the method are also suitable to other modeling approaches (like HDM [Garzotto 91,93], EORM [Lange94], etc...); in fact they are compatible with other step-by-step methods (for example the one presented in [Balasubramaniam94]).
Using well-known object-oriented modeling concepts allows the formulation of complex designs with a concise yet expressive notation thus simplifying the construction of Computer Aided Hypermedia Development Environments (CAHDE).
The structure of this paper is as follows: in section 2 we briefly overview our approach for building hypermedia applications; in sections 3 and 4 we present the core of our method: specifying a hypermedia application as a navigational model derived from a conceptual schema, discussing modeling constructs and abstraction mechanisms. Finally we summarize the key aspects of our approach and discuss some additional issues.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

2 - Overview of our approach

We claim that building a hypermedia application is a four step process, where these steps are used in a mix of iterative, incremental and prototype-based styles of development. In each step a model is built or enriched; after building the last one, we have enough information to implement the hypermedia application. During each step we favor the use of a model-based (as opposed to method-based) approach; moreover as already stated, the order of steps in indicative and not prescriptive.

2.1 - Step 1: Conceptual Model Design

During this step a conceptual domain model is described using an object-oriented hypermedia design model (OOHDM) whose modeling primitives are: classes, relationships and sub-systems. Other hypermedia design models may be used in this step: HDM, EORM, etc... This modeling approach is similar to existing ones in the object-oriented field though enriched with some ideas (like attribute perspectives) and applied with rather different criteria (emphasizing object structure and relationships rather than object behavior). OOHDM can be considered a direct descendant of HDM, providing some higher level modeling constructs (sub-systems) and abstraction mechanisms (class hierarchies).

In terms of OOHDM the following activities are performed: definition of classes, sub-systems and relationships according to the domain semantics, building of part-of and is-a hierarchies, assignation of types to attributes; enrichment of relationships with cardinality information and addition of instance-specific information to the schema. Modeling constructs and abstraction mechanisms are similar to the ones found in object-oriented modeling approaches like OMT [Rumbaugh91]; class attributes may be multiply typed thereby defining different perspectives of the same information.

2.2 - Step 2: Navigational Design

During this step we describe hypermedia applications, by defining navigational structures that take into account the class (profile) of the intended users, and the set of tasks they are to perform using the system. Typical classes defined during this step are Nodes, Links, Indices, Guided Tours, etc.; we call these Navigational Classes. Navigational operations like following a link or selecting an item in an index are specified by defining the transformations on the Navigational Space, i.e. the set of accessible navigational objects. The Navigational Space plays a similar role as the "rhetorical space" in Sepia [Thüring et al, 1991].

Nodes represent logical windows on classes defined in the conceptual schema and links are derived from relationships in the schema. Different Navigational Models may be built for the same Conceptual Schema thus expressing different views of the same information base. This approach is highly compatible with modern design and implementation architectures with a shared database acting as an information server for client (hypermedia) applications [Fowler94].

2.3 - Step 3: Abstract Interface Design

The goal in this step is to specify how the user will perceive the navigational objects through the interface; this specification is to be done at a higher level than that of actual implementation environments. It is in this step that the interface metaphor and interaction styles are defined.

Hypermedia applications, like other interactive applications, require that the author must specify what are the perceptible objects the author intends to make available to the user, and how they behave in terms of the actions originating from the user (and from other perceptible objects). Perceptible objects will be in general built using primitive objects like buttons, text fields, graphic fields, etc. and will provide the interface for navigational objects as defined previously.

Some of the perceptible objects will be activated by the user, and such activation will cause transformations in the perception context (i.e., the set of perceptible objects), thus implementing the navigational style defined during Step 2. The dynamic behavior of the application can then be specified as the set of possible transformations to any given perception context – new objects appear in the perception context, while others disappear.

Navigational Classes (such as Nodes) provide information to be mapped (i.e., become contents of) perceivable objects. A clean separation between both concerns, navigational and abstract interface design, allows building different interfaces for the same model thus conforming with varying user preferences, user-interface technology or

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

implementation restrictions. As a summary, while during Step 2 we specify what objects are going to be navigated, during Step 3 we specify in which way those objects will be perceived.
We are now using Abstract Data Views (ADV) [Cowan93] as a design tool for specifying perceptible objects and their transformations. Abstract Data Views allows specifying the interface aspects of an object independently of the object itself. Using ADV's we specify interface aspects and behavior of Navigational Classes. After this step has been performed we have enough information to implement the application using a hypermedia system.

2.4 - Step 4: Implementation

By mapping the navigational and abstract interface models – the perceptible objects and their transformations – into concrete objects , i.e. those available in the chosen implementation environment, the author produces the actual hypermedia system to be run. In particular the model generated after performing Steps 1 through 3 can be implemented in a straightforward way on top of available hypermedia platforms such as Hypercard, Toolbook, KMS, Guide, Microcosm, etc.. Rules for performing this implementation can be systematically applied or incorporated in a CAHDE built on top of the target environment.
In this paper we will discuss in more detail Steps 1 and 2 of our method; a brief description of step 3 will also be presented, and we will only mention Step 4.

3 - Building an Object-Oriented Hypermedia Domain Model

An Object Oriented Hypermedia Modeling Schema (the Conceptual Schema) is built upon objects, classes, relationships and sub-systems. The schema consists of a set of objects and classes connected by relationships; objects are instances of classes, and thus, when a relationship holds between classes, it abstracts the object-to-object relationship. Classes may be related to sub-systems (abstractions of a whole hypermedia schema). Figure 1 shows part of the schema of a Tourism application. The notation is similar to Rumbaugh's OMT ([Rumbaugh91]) enriched with sub-system information as proposed in [Guilliam94, Wirfs-Brock90]. Boxes represent Classes with name and attribute information inside; arcs represent relationships; ovals represent sub-systems (further described in what follows); and the small diamond at the end of an arc indicates aggregation.

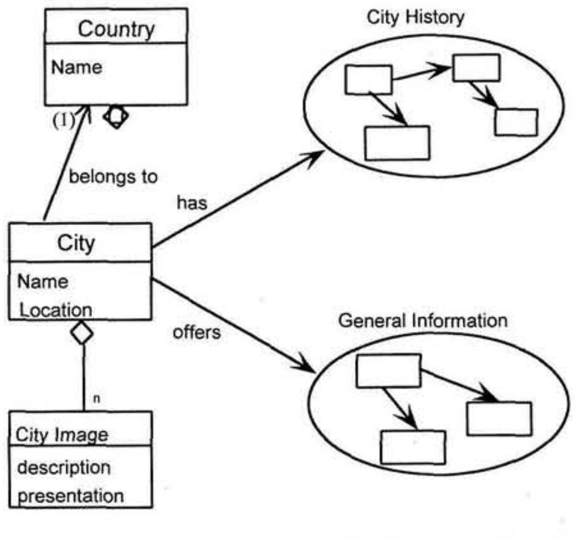


Figure 1: Conceptual Schema of a Tourism Application

In the example we have defined three main classes: City, City Image (in fact Cities are aggregations of City Images) and Country, and two sub-systems: "City History" and "General Information". The rationale for defining Sub-Systems is not application-dependent but rather domain-dependent. Sub-Systems stand for complete hypermedia schemata that are used as part of another hypermedia schema. They may be used when another hypermedia schema or application already exists or may be defined opportunistically as a modularization strategy.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Usually sub-systems contain entry points, i.e. classes in the sub-system that may be accessible from other classes outside the sub-system. Entry points are defined while defining navigational semantics. Sub-systems may also be nested, i.e. a sub-system may contain others.
As in other object-oriented approaches, classes are described with typed attributes and behavior. For the sake of conciseness we will focus on structural aspects and omit the discussion about behavior.
Relationships in our model express relations between domain objects intended to be navigated by the final user, and will be mapped to links in the Navigational Views. It should be noted that relationship definition is independent of any particular navigational semantics; when navigating from a city to the country it belongs to, the information could be presented in the same window or not. We will define the navigational semantic of relationships when deriving links during Step 2, and the way navigation will be perceived while defining interface aspects, during Step 3.
Relationships are also defined as classes thus including attributes and behavior, and are further organized in hierarchies. Cardinality constraints may also be specified when defining relationships. In Figure 1, cities belong to exactly one country.

3.1 - Attributes, Types and Perspectives

Class attributes are typed and represent intrinsic or conceptual properties of objects (a City name, its location). The type (or class) of an attribute will represent either an implicit relationship (when the type refers to other objects in the schema and as such it will be mapped to a hypermedia link), the kind of media used to represent the attribute or its rhetorical appearance in the final hypermedia application.
Each possible appearance of an attribute is called a perspective of the attribute. We use the same semantic for perspectives as HDM; some examples of this feature are : the attribute "location" in a "City" (Figure 1) might be viewed as a text or a bitmap, the attribute "presentation" in "City Image" might be viewed as a text, a bitmap, or a video. When multiple perspectives exist we use "[...]" and if one of them is the default one we mark it with a +. In the previous example we would write:
presentation: [Text, Bitmap, Video+].
Only the default perspective must be present in all instances, while the others may or may not be implemented. Note that, as explained in [Garzotto 91,93], perspectives will originate a class of hypermedia links not explicitly specified as relationships in the schema, namely, those connecting different perspectives of the same attribute (perspective links in HDM).

3.2 - Abstraction mechanisms: Aggregation and Inheritance

In our modeling approach we provide two abstraction constructs for dealing with complexity: Aggregation and the pair Generalization/Specialization. The first one is useful for describing complex Classes as aggregates of more simple ones and the second for building Class Hierarchies and using inheritance as a sharing mechanism. In addition, the notion of sub-system may be viewed as a third high-level abstraction mechanism.
Part-of relationships (e.g., book chapters, scenes of a film) are described using aggregation relationships. Aggregates in a class definition are similar to components in HDM. Implicit relationships exist between a complex object and its parts (and vice versa) and between the parts themselves, corresponding to the structural links in HDM. Judicious definition of aggregate structures is important in hypermedia because their specification can be useful when defining navigational views. Understanding the exact nature of an aggregate, i.e. what kind of object composition it represents is important for building good navigational structures (see [Odell94, Winston87]). Figure 1 shows an aggregation (of Images) for Class "City"; in Figure 2, "Country" is defined as an aggregation of "Political System", "Geography" and "People" Classes.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

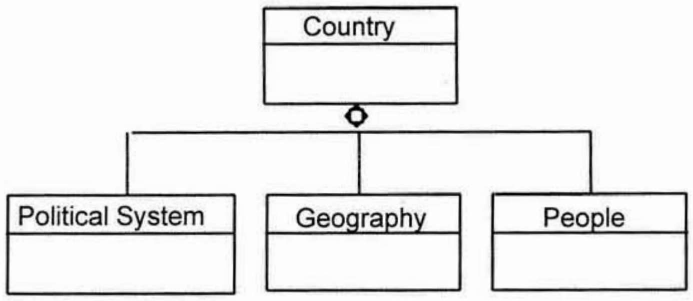


Figure 2: Country as an aggregate of other Classes

Classes inherit attributes, parts-structure, relationships and behavior of their super-classes. In Figure 3, the relationship between "Tourist Attraction" and "Town", reflects the fact that instances of each sub-class, "Tourist Place" and "Events", are located in a "Town" and that different kinds of "Tourist Places". share attributes defined in "Tourist Attraction". In our modeling approach, inheritance follows the usual semantics in object-oriented data models as defined in [Lucarella93, Rumbaugh91]. As previously stated, the use of class and aggregation hierarchies is a natural extension to HDM providing concise and easy to extend schemata. In Figure 4 we show the Conceptual Schema for sub-system "General Information" .

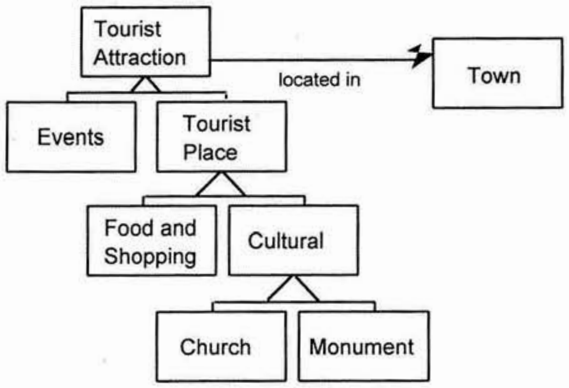


Figure 3: A Class hierarchy of Tourist Attractions

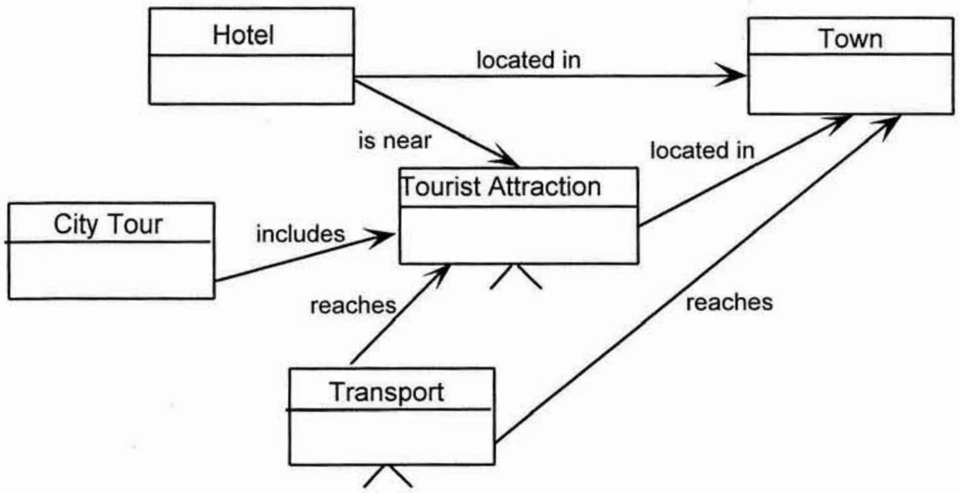


Figure 4: "General Information" Sub-System

In Figure 4, some classes, such as "Tourist Attraction" and "Transportation" have sub-classes (not shown to simplify the diagram). In Figure 5, we refine the definition of "Hotel"; in section 4 we will show Node definitions

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

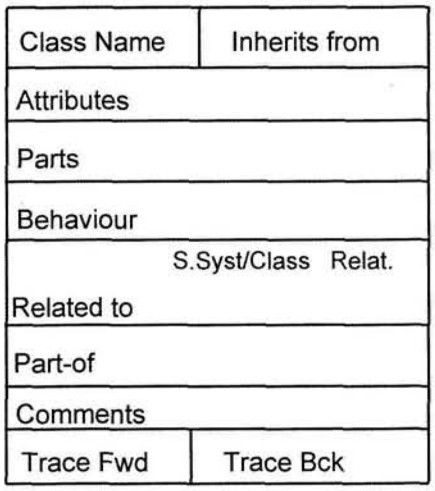
for classes “City Tour” and “Hotel”. Note that we could have defined a more abstract class “Lodging” including hotels, hostels, bed and breakfast, pensions, etc.

3.3 - Documenting the Model

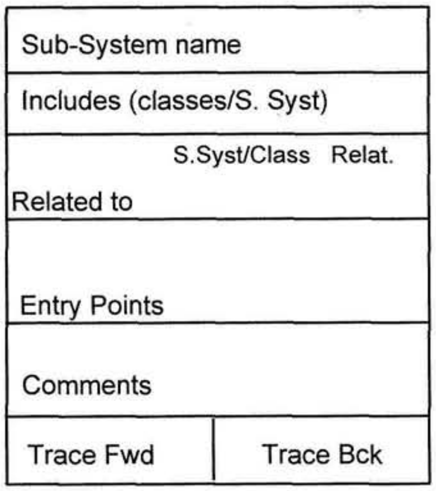
A very common problem when building large applications (in particular hypermedia applications) is that it is difficult to document the schema; to make matters worse Computer-Aided Software Engineering Environments are scarce in the hypermedia field.

To document the schema, we propose using cards similar to CRC cards [Wirfs-Brock90]. These cards are easy to manipulate and may contain, among other things, trace information, critical during maintenance.

We use Class, Relationship and Sub-system cards for documenting the model. Figure 5 shows Class and Sub-System cards templates and the Class card for “Hotel”. Cards include information about the artifacts they document and trace information. Backward trace allows answering questions such as: where does this artifact come from? (a real world entity or relationship for example). Forward trace is important in system evolution for analyzing the impact of changes in the model: (What Navigational Classes does this change impact?). Cards are easy to manipulate and can be readily automated in a hypermedia environment.



Class Card



Relationship Card

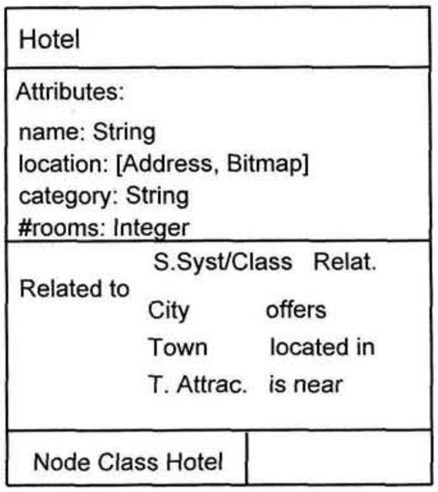


Figure 5: Class and Relationships Cards. An example: Class Hotel

In class card “Hotel” we have omitted comments and information about parts and behavior. The field on the bottom left indicates that Node class “Hotel” is derived from this Class; in a computer-based implementation, this reference could be implemented as a hypermedia link.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

4 - Navigational Design

Though we have already defined a conceptual schema, it may be rather abstract in terms of user needs, and as previously stated it does not include information about navigational aspects.
In our approach a hypermedia application is derived from the conceptual model by defining nodes, links, access structures, etc. that act as logical windows on the classes defined in the conceptual schema. Different hypermedia applications may be derived from the same schema, each one supporting the needs of a particular kind of user; in our example we can build a hypermedia application providing detailed access to accommodation resources like hotels, stressing information on transportation, etc. for use by a tourist visiting the city. In the same way we can build the touring view in which the user (a prospective tourist) will navigate through "Tourist Attractions"; in this view we will emphasize multimedia presentations, like videos, images, etc...
In this paper we will not further elaborate on implementation issues such as building a shared hypermedia database and considering each hypermedia application as a (complex and navigational) database view. However, the conceptual framework presented in this paper naturally leads to this implementation scheme.
Deriving a hypermedia application from a Conceptual Schema involves defining Nodes, Links, Access Structures, Navigational Contexts and transformations on the Navigational Space.

4.1 - Defining nodes

Nodes are the basic information containers in hypermedia applications and their structure depends on the application semantics, i.e., the particular interests of intended application users.
A node class is characterized by specifying the conceptual class(es) from which it is derived (its subject(s)), its attributes and the anchors it contains. Node attributes must be single-typed, i.e., when multiple perspectives exist in the conceptual class, different attributes or different node classes must be defined (one for each perspective). It should be kept in mind, however, that having different node classes for different perspectives does not necessarily imply that the user will actually perceive different objects, as multiple nodes may be mapped into a single perceptible object in Step 3. Node Classes may be grouped in inheritance hierarchies (see example below) that in general will mirror the class hierarchy in the Conceptual Schema.
In Figure 6 we show Node Classes "City" as defined for the navigational views "Tourist" and "Touring".

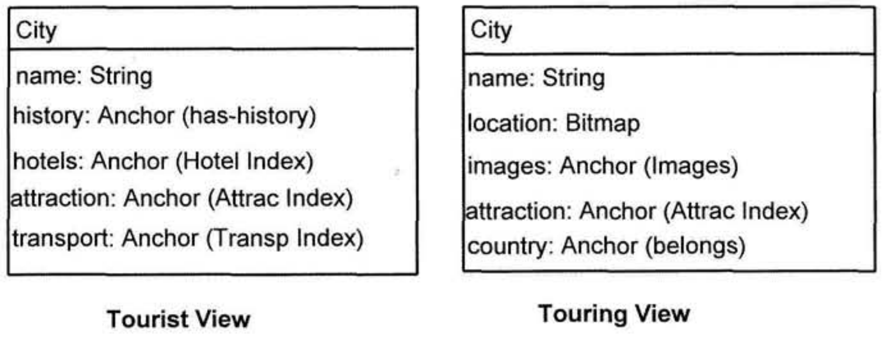

City
name: String
history: Anchor (has-history)
hotels: Anchor (Hotel Index)
attraction: Anchor (Attrac Index)
transport: Anchor (Transp Index)
Tourist View
City
name: String
location: Bitmap
images: Anchor (Images)
attraction: Anchor (Attrac Index)
country: Anchor (belongs)
Touring View

Figure 6: Node Class City for Tourist and Touring views

In Figure 6 the structure of nodes is derived almost directly from their corresponding conceptual classes; the node subjects are the conceptual classes with the same name. Node attributes are derived from some of the corresponding conceptual attributes and relationships (see Figures 1 and 4). Anchors allow accessing links (such as "has-history") or access structures (such as "Hotel Index"). In the "Tourist" view, we have omitted anchors for accessing "City History", and correspondingly we will omit defining links from "City" to "City History". In the same way, the "Touring" view includes city images and excludes information about hotels.
In Figure 7 we show node class "Hotel" (derived from conceptual class "Hotel").defined for the navigational view "Tourist".
Note that in this case we have filtered some attributes and relationships and added others. Links and access structures are further explained in the next sub-section.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Hotel
name: String
location: Address
category: String
town: Anchor (located in)
attraction: Anchor (is near)

Figure 7: Node Class Hotel for Tourist view

Nodes may be formed out of a combination of features from different conceptual classes (similar to derived entities in HDM) combined among them or with access structures.

4.2 - Defining links and access structures

In our model, links implement relationships defined in the conceptual schema. In other words, links are the *navigational realization of relationships*. Link classes are defined by specifying link attributes and behavior, source and target objects and cardinality. Link attributes express properties of the link itself and may be useful when defining n-ary links. In such cases the link may behave as a node (similar to the web center in HDM), acting as an intermediate object (between the link source and destination) during navigation. Taking into account link behavior patterns as discussed in [Lange94], link classes may be organized into hierarchies where abstract classes provide common behavior, and concrete ones add structure and eventually specialize behavior.
Implementing one-to-many relationships, like "offers" (Figure 1) requires some design decisions. In particular, though "offers" has been originally defined as relating "City" with the Sub-System "General Information", when refining this Sub-System "offers" must be detailed further.
In Figure 8 we show link class "is near " using a Link card. Such a link may be defined as one-to-one, which requires multiple anchors (one for each instance) or as one-to-many, which requires a chooser or an index as an intermediate object. The same holds for "structural" links like the one implementing the aggregate structure in "City". Anchors for bi-directional links may be derived directly from the link definition.

Link: is near
sources Hotel
targets Tourist Attraction
cardinality 1-to-m
attributes how-to-go: Text

Figure 8: Definition o f Link class "is near"

Access structures act as indexes or dictionaries and are useful for helping the final user find the desired information (the list of "Monuments" or "Restaurants" in a "City", a "Hotel Catalog", a guided tour through some selected "Tourist Places", etc.). Access structures are also modeled as classes and further characterized by a set of selectors, a set of target objects (usually objects in the schema) and a predicate on target objects. The predicate expresses which objects will be accessible in terms of their properties. Selectors usually stand for some of the attributes of the target objects and are organized according to a pre-defined data structure (an ordered list, a set of icons, etc.). In either case they must be explicitly mentioned in the definition of the access structure.
Behavior in access structures is useful for specifying the way in which the access structure will be used; for example a guided tour will include operations such as "Start", "Next", "End", etc. implementing different semantics as discussed in [Garzotto94].
Some access structures may be defined in the conceptual schema level, and may be used in any navigational view; for example the "Hotel Index" or the "List of Monuments" are useful in different views. Note that this is almost

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

always true for access structures allowing access to all instances of a class. In Figure 9 we show the Access Structure "Hotel Index" using the card formalism; we have not included a predicate on target objects because we want the whole set of hotels to be reachable. A predicate could have expressed, for example, the fact that we wanted to access those hotels located in a particular town or with fares within a particular price range. Note that this definition resembles conditional indexes as discussed in [Balasubramaniam94].

Access Structure: Hotel Index
Target Hotel
Selectors name (ordered)
predicate

Figure 9: Hotels Index

As discussed previously, access structures may be accessed from nodes, using specified anchors or may themselves be part of a composition node – for example, a node consisting of domain information plus access structures. In Figure 10, we can see the definition of a "Monuments Catalog" as a composition node. The value of the attribute "index" (an access structure) is fixed for a particular catalog and that the value of attribute "monum" varies dynamically according to the actual selection made within the index (as defined in the class behavior, which is not shown). We have encountered this kind of node in many actual examples; it seems to be a recurrent pattern in hypermedia applications, with a clear navigational semantics and easy to implement in hypermedia platforms.

Monuments Catalog
Index: Monuments Index monum: Monument

Figure 10: A Composition Node – Monuments Catalog

Finally, in Figure 11 we show part of the navigational schema for the "Tourist" view. Small boxes represent access structures and dashed lines represent links that are automatically derived from node and access structure definitions (i.e., links connecting nodes and access structures). The navigational schema gives a snapshot of the navigational structure as previously defined.

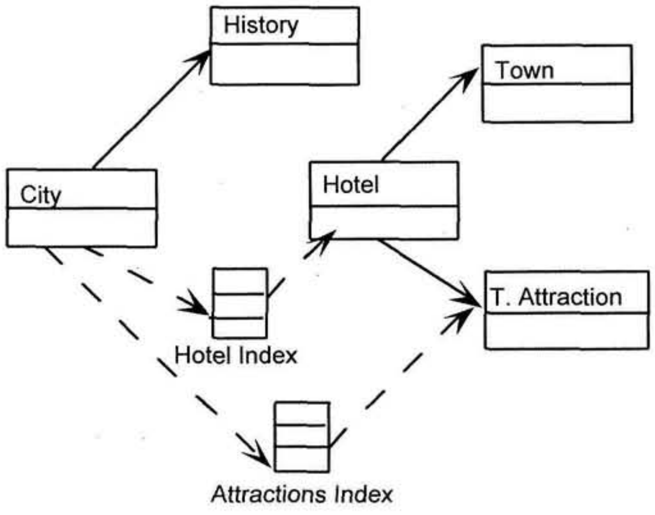


Figure 11 A Navigational schema

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

In the example above (Figure 11) we kept the navigational schema simple since we defined "Tourist Attraction" as an abstract node class mirroring the hierarchy in Figure 3. However, a more detailed design would need different access structures for Churches, Monuments, etc. In fact, "Attractions Index" can be defined as an index of indexes, where each of the indexes points to all instances of a particular kind of "Tourist Attraction".

4.3 - Navigational Contexts and the Navigational Space

As previously said, defining the Navigational Model (in the OOHDM terminology, a Navigational View) involves defining which are the objects that the user will navigate; those objects are: Nodes, Links and Access Structures. We complete the Navigational model by specifying Navigational Contexts and transformations to the Navigational Space.
To make the discussion concrete let us return to our previous example and suppose that each "Tourist Attraction" has a detailed aggregation structure or that some attractions (e.g., monuments) are further linked to nodes representing the artists that created them. To avoid disorientation we may want to specify that a user navigating through hotels is not able to navigate "into" "Tourist Attractions"; in other words, when the user navigates from a "Hotel" to a "Tourist Attraction" he can only see the top level of the hierarchy (a "snapshot") of the attraction, and is prevented from navigating into the aggregation structure.
On the other hand, we want to allow tourists wanting to directly explore "Tourist Attractions" by navigating from the corresponding index, to be able to do so. To solve this problem we define two Navigational Contexts: "Hotels" and "Attractions" and refine node class definitions, specifying attributes and anchors for each Navigational Context that makes sense for the node. In this way, when the user navigates to a "Tourist Attraction" in the "Hotels" context, the corresponding node will not contain anchors to its parts or to other nodes.
When the user explores the "Attractions" context, nodes will contain "full" information. In Figure 11 we show how the previous discussion is reflected in the definition of node class "Monument". Defining the class consists in defining which are the global "attributes" (i.e. those accessible in all Navigational Contexts), and which ones are "private" to a given context. In this case, we have only described the "Attraction" context; navigating to nodes of this class in other contexts gives access only to global attributes ("name" and "description").

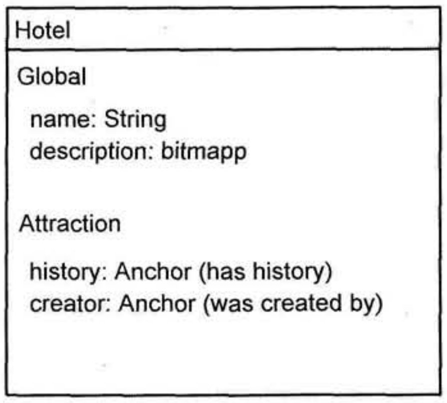

Hotel
Global
name: String
description: bitmapp
Attraction
history: Anchor (has history)
creator: Anchor (was created by)

Figure 11: Defining a Node Class according to Navigational Contexts

Note that the definition of Navigational Context does not take into account the user's profile but the navigation paths that may be followed. Navigational Context may be nested in a similar way as in [Casanova 91] and they solve the problems addressed in [Garzotto94a]. Note that link traversal may change the actual navigational context; we specify this fact as part of the link behavior. Reference [Barbosa 94] discusses Navigational Contexts in detail.
Each time the user follows a link, navigating from one navigational object to another, the Navigational Space, i.e. the set of available navigational objects, may change: for example we may want to specify that when following a link from "Hotel" to "Attraction" the node "Hotel" is no longer accessible; or we may want to define navigation on hierarchical structures such that when leaving a part (component) of the structure, by navigating to a different structure, the whole aggregate structure disappears (i.e., is no longer accessible). These transformations on the set of navigable objects, which are specified as part of link behavior, constitute the navigation semantics.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

5 - Defining the Abstract Interface Model – an Overview

Once the Navigational Model has been built we must define the appearance of navigational objects as well as the interaction style. Many authors consider this step as being an implementation concern, since modern Graphic User Interface libraries simplify the task of defining graphical interface objects. We claim, however, that this step must be faced in an implementation-independent way.
Given the definition of navigational objects and of the navigational style – i.e. which transformations occur to the Navigational Space and in which way objects will be navigated according to the navigational contexts they belong to – we now turn to the interface metaphor and the perceptible transformations occurring during execution of the application.
In our method defining an Abstract Interface Model consists in the specification of Interface Classes and their connections to navigational classes. In its simpler form, the Interface Model will include a Class (an Abstract Data View) for each node class (the View owner); its attributes will express the way in which each Node attribute and anchor is perceived by the user. In more complex cases, Interface Classes will correspond to combinations of Navigational Classes.
The dynamic behavior of the interface, i. e. the way in which user operations affect the perception context is specified as methods in the Abstract Data View. Methods may be associated with external events: the user selecting an anchor, or with "internal" ones: a node is being opened or closed. Interface behavior complements navigational behavior as specified in Step 2: during navigational design we specify the cognitive aspects of navigation while during Interface design we emphasize perceptive aspects.
Depending on the actual definition of the Abstract Data View classes, one may obtain a more "concrete" or more "abstract" interface definition. A "concrete" one would specify actual screen locations, window panes, formatting, etc..., whereas an "abstract" one would only specify which objects are perceptible, and among these which can be activated. In this latter case, activating a perceptible object will cause transformations to the set of perceptible objects (called the perception context), removing some from the current context and adding others.
An Abstract Interface Model may also include interface objects not previously defined that implement the interface metaphor and that communicate by message passing with other interface objects. For example, the notion of "focus of attention", which oftentimes is realized concretely as a cursor or highlight.
This object-oriented approach to interface specification is highly compatible with modern trends in User Interface Design and, and is similar to the way in which some commercial hypermedia environments such as Toolbook and Hypercard treat the user interface. In these systems, however, all objects are interface objects, thus merging navigational and interface objects. Once we have finished the interface model (that can of course be prototyped using modern user interface libraries) we have all the information we need to begin with implementation.

6 - Implementation

Being implementation highly dependent of the target environment we will only outline two possible implementation styles: using a hypermedia environment and using a general purpose object-oriented system.
In the first case the usual alternative will be mapping the interface objects to the system defined objects: cards, pages or frames. As we mentioned before many conventional hypermedia environments do not separate user interfaces from nodes, but provide a scripting language in which we will have to code both the interface and the navigational transformations. Nevertheless, provided we have clearly documented the products produced in Step 2 and 3, implementation is rather straightforward.
In contrast, it is more difficult to build different Navigational Views of the same Information Base, mainly because data objects and interface objects are the same objects, thus requiring the implementor to manage all the mapping between the two types of objects. Using an object-oriented hypermedia platform like MacWeb [Nanard91] may simplify the translation from model to implementation, reducing the "impedance mismatch" between concepts in both worlds.
The second alternative is the most interesting in some domains such as Management Information Systems. In this case the conceptual model can be implemented as a shared object-oriented repository (possibly on top of an object-oriented data base management system, see [Bancilhon92]), and different Navigational Views will be built as object networks that refer to the shared database for getting and storing information. Abstract Data Views can be easily implemented using this alternative, as well as other interface paradigms (like the model-view-controller). It is obvious that this alternative is the best for dynamic hypermedia applications, i.e. those in which nodes and link are

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

modified/added while the system is used: CASE environments and decision support systems are a good example of this kind of application.
As previously said the four steps of the life-cycle of a hypermedia application design and development are performed combining different process models: cascade, incremental, iterative, etc. In this sense our approach favors the use of a model-based approach over a method-based.

7 - Summary. Goals of our approach

We briefly summarize in this section the main goals of our method in relation to the problem of hypermedia design. First, our approach identifies three steps previous to implementation in which different problems are addressed:

1 - domain modeling and conceptual schema construction.

2 - definition of navigational structures and semantics.

3 - definition of perception aspects and dynamic behavior.

In each step we use high level modeling constructs: classes and abstraction mechanisms – aggregation, inheritance – to reduce complexity. Modeling constructs enrich conventional object-oriented approaches with hypermedia oriented abstractions like perspectives, anchors, access structures, etc...
We also propose a simple documentation framework: the use of Cards that not only allows documenting each artifact defined during the design process but mainly support a simple but powerful traceability model.
Clear separation of concerns from steps 1 to 3 allows building different hypermedia applications for the same domain model by deriving different Navigational Schemata from the same Conceptual Schema. Moreover, different interfaces may be built for the same hypermedia application. Some novel concepts are also introduced in this approach, the most important ones being the specification of Navigational Contexts, the transformations on the Navigational Space, and the notion of Abstract Interface Design. Finally, it is worth saying that though using rich modeling primitives, the resulting model can be implemented using a conventional hypermedia environment.

8 - Related Work

Our work is similar to other model-based approaches to hypermedia design like HDM [Garzotto93] and its descendants, in that it recognizes the importance of specifying a conceptual schema prior to implementation. It is based on an object-oriented model that builds hierarchical structures as aggregations of simpler ones and encourages the use of perspectives for presenting the same conceptual entity in different ways. The underlying model enriches HDM in that it provides higher level abstraction constructs (classes and objects) and mechanisms (generalization/ specialization). Our approach for building navigational views generalizes the use of Derived Entity types in HDM, which can play a similar role. Furthermore, HDM does not deal with abstract interface design.
Recently HDM has been extended with the notion of collections [Garzotto 94a], which are similar in spirit to Navigational Contexts. One important difference is that collections are composed of the equivalent to our "conceptual objects" and not of navigational nodes. Furthermore, these objects are the same in all collections they belong to, and navigational nodes can be particularized depending on the context.
A step-by-step methodology for designing hypermedia applications, named RMD has been recently proposed [Balasubramaniam94]. RMD is built on top of HDM and HDM2 [Garzotto94b]. It enhances HDM2 concepts with additional access structures (conditional indexes and guided tours) and proposes a seven step process for building hypermedia applications. Our approach is similar in that we identify several steps prior to implementation (like Navigational and Abstract Interface Design), though it is different in that it uses classes and objects during the whole process. It also formalizes navigational design as the process in which different views of the same domain are built.
Object-Oriented ideas have been used in the hypermedia field for some time now. However, with the exception of [Lange94], objects have been mainly used as implementation artifacts (See for instance [Marmann92, Nanard91]). Our modeling approach differs from those mentioned in that it addresses design aspects rather than implementation ones. Though being based on well known Object-Oriented modeling approaches it includes some novel features like the use of attribute perspectives.
Recently, [Lange94], has proposed extending an object oriented modeling technique (OMT, [Rumbaugh1]) with hypermedia links semantics. The resulting design model (EORM) can be compared with the underlying model in

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

our approach (OOHDM) in that it uses well known object-oriented concepts and mechanisms for representing application domains.
However, some key differences must be highlighted: First, the overall process is different because we divide the task of hypermedia construction so that during the task of domain modeling we only focus on objects, attributes and relationships in the domain without considering navigational semantics. Navigation aspects are addressed while constructing hypermedia applications, considering them as views on the conceptual schema. Second, the underlying O-O model provides an additional high level modularization construct, the sub-system. We also use attribute perspectives (the object-oriented equivalent of HDM perspectives or HDM2 slots) and consider aggregation as a "first-class" relationship, which provides a natural context for defining navigational semantics (similarly to entity structures in HDM). Finally, though not shown in this paper we use instance-diagrams to enrich the conceptual schema with instance peculiarities.
Our proposal is similar to current work in the object-oriented design field, aimed at separating the application (and presentation) views from the conceptual model like [Fowler94]. The key difference between our object-oriented method and those like Fowler's or Wirfs-Brock's [Wirfs-Brock90] is that ours emphasizes structure and relationships over behavior (during Step 1) and uses behavior for specifying Navigational and Interface Transformations.

9 - Conclusions

We have presented an Object Oriented Hypermedia Design Method. Using this method it is possible to build a complex hypermedia application as a stepwise process beginning with domain analysis and proceeding with navigational and abstract interface design. The method is based in the construction of different models: Domain, Navigational and Interface model, using known object oriented concepts such as object structure and behavior, abstraction mechanisms such as aggregation and generalization/specialization which provide inheritance. Though not mentioned in this paper each model may be further enriched with instance-specific information when in a particular application, certain instances of a class exhibit exceptional features particular only to those instances.
The four steps outlined in section 2 provide a smooth path from high level domain modeling to implementation. Using Navigational Classes it is possible to provide a seamless transition from domain and application modeling to concrete hypermedia design. Using abstract interface specifications, it is possible to map the hypermedia objects defined in Navigational Classes into perceptible objects, which can in turn be mapped into concrete implementation objects.
We have used our approach to model many of the applications that had been already modeled using HDM, resulting in more flexible schemata (i.e., easier to extend and to reuse). We have also modeled more complex applications, such as a hypermedia-based software engineering environment, in particular a complete CASE environment for an object oriented software engineering methodology, and another CASE environment for our model and the associated methodology. Both examples are particularly interesting since they are evolving applications, i.e., ones in which nodes are added by the readers as they use the application. We noticed that using our modeling constructs and separating the conceptual schema from the navigational views we enhance modularity thus allowing easier evolution and maintenance of hypermedia applications. Once implemented, even using sophisticated object-oriented environments, the number of nodes and links depends only on the application complexity and the information base size.
Using well known object oriented analysis and design guidelines it is possible to obtain modular and well-structured hypermedia schemata in which connection patterns among instances reflect patterns defined in the class schema. In this way, it reduces the gap between design and implementation and helps during testing and evolution of the application.
Other aspect that have been addressed in our work is the need for a traceability model allowing to trace design decisions into an implementation, for helping during maintenance, reuse of existing artifact, etc. The use of a uniform formalism in which all key decisions in hypermedia application building must be recorded, together with a set of documentation aids (Class Cards) goes straight in that direction. As previously discussed designing and implementing very large evolving hypermedia applications is still an open field and using object oriented modeling techniques is a key approach for solving the problems we are facing with.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

10 - References

[Balasubramaniam94] P. Balasubramaniam, T. Isakowitz and E. Stohr: "Designing Hypermedia Applications", Proceedings of the 27th. Hawaii International Conference on System Sciences.

[Bancilhon92] F. Bancilhon, C. Delobel and P. Kanellakis: "Building an Object-Oriented Database System. The story of O2". Morgan Kaufmann, 1992.

[Barbosa 94] Barbosa, S. and D. Schwabe, "Navigation Modeling in Hypermedia Applications", Technical Report, Departamento de Informática, PUC-Rio, 1994

[Casanova 91] Casanova, M. A; Tucherman, L.; Lima, M.J.D.; Rangel Netto, J.L.M.; Rodriguez, N.; Soares, L.F.G.;, "The Nested Context Model for Hypertdocuments", Proceedings of the Hypertext'91, San Antonio, 1991.

[Cowan93] D. Cowan R. Ierusalimschy, C.J.P. Lucena and T.M. Stepien: "Abstract Data Views". Structured Programming, 14(1):1-13, January 1993.

[Fowler94] M. Fowler: "Application Views: Another technique in the analysis and design armoury", JOOP, Vol. 7 N 1., pp. 59-66.

[Garzotto91] F. Garzotto, P. Paolini and D. Schwabe. "HDM- A Model for the Design of Hypertext Applications", Proceedings of Hypertext'91 , ACM Press. pp. 313.

[Garzotto93] F. Garzotto, D. Schwabe, P. Paolini: "HDM- A Model Based Approach to Hypermedia Application Design", ACM Transaction on Information Systems, Vol. 11, #1, Jan. 1993, pp. 1-26.

[Garzotto94a] F. Garzotto, L. Mainetti and P. Paolini, "Adding Multimedia Collections do the Dexter Model", Proceedings ECHT'94, Edinburgh, Sept. 1994.

[Garzotto94b] F. Garzotto, P. Paolini and L. Mainetti: "Navigation in Hypermedia Applications: Modeling and Semantics". Journal of Organizational Computing (forthcoming).

[Gilliam94] C. Gilliam: "An approach for using OMT in the development of large Systems". Journal of Object-Oriented Programming, Feb. 1994, Vol 6, N. 9 pp. 56-60.

[Lange94] D. Lange: "An Object-Oriented design method for hypermedia information systems", Proceedings of the 27th Annual Hawaii International Conference on System Science, January 1994.

[Lucarella93] D. Lucarella: "Multimedia Object Retrieval Environment", Proceedings of Hypertext'93, pp. 39-50.

[Odell94] J. Odell: "Six different kinds of compositions", Journal of Object Oriented Programming, Vol. 5 N. 8, 1994.

[Marmann92] M. Marmann and G. Schlargeter. "Towards a better support for hypermedia authoring: The HYDESIGN model", Proceedings of ECHT'92, ACM Press, pp. 232.

[Nanard91] J. Nanard and M. Nanard. "Using Structured Types to Incorporate Knowledge in Hypertext, Proceedings of Hypertext'91 ACM Press. pp. 329.

[Rumbaugh91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W.Lorensen. "Object Oriented Modeling and Design", Prentice Hall Inc. 1991.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

[Wirfs90] R. Wirfs-Brock, L. Wiener and R. Wilkerson: “Designing Object-Oriented Software”, Prentice Hall 1990.

[Thüring 91] Thüring, M.; Haake, J.M.; Hanneman, J., “What’s Eliza doing in the Chinese Room? Incoherent hyperdocuments – and how to avoid them”, Proceedings of Hypertext’91 , ACM Press. , pp. 161.

[Winston87] M. Winston, R. Chaffin and D. Herrmann: “A taxonomy of part-whole relationships", Cognitive Science, Vol. 11, pp. 417-444, 1987.

[Wirfs-Brock90] R. Wirfs-Brock, L. Wiener and R. Wilkerson: “Designing Object-Oriented

Software", Prentice Hall, 1990.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Analysing the Quality of Hypermedia Applications:
A Design-Oriented Framework

Franca Garzotto [1], **Luca Mainetti** [1], **Paolo Paolini** [2,1]

(1) Hypermedia Laboratory - Politecnico di Milano °
(2) University of Lecce

Abstract
The software engineering community has defined a number of "quality models" to help establishing the quality of a generic software product. These models identify the key factors of software products that have a special interest for the end user; these factors are analysed by decomposing them into lower level attributes that can be directly analysed and possibly measured.
In this paper, we will specialise this approach to address the problem of hypermedia quality. We will provide a framework for *design-oriented hypermedia evaluation,* that defines a set of design objects, based on HDM (Hypermedia Design Model), and a set of fine-grained attributes of these objects that can impact on one fundamental quality factor, i.e., hypermedia usability. The approach will be exemplified by analyzing a commercial hypermedia application, and by elaborating some qualitative evaluations on its design.

1. Background and Motivations

Hypermedia applications today are no more relegated to research laboratories. There is an explosion of commercial hypermedia for a variety of tasks, and a huge number of public domain hypermedia, developed on the World Wide Web, are potentially available to millions of Internet users.
This scenario raises a number of issues for hypermedia producers, distributors, and end-user. How will vendors, or product managers, determine which applications to recommend to their customers? How will electronic publishers select the hypermedia titles to publish on the market? How do will product managers or product developers monitor their work and compare their products with the existing ones? How will end-users "avoid disorientation" in the universe of hypermedia applications, and distinguish between "good" or the "bad" products?
These questions are all related to the problem of *establishing the quality* of hypermedia applications, which, differently from other fields, has so far received to far a little attention by researchers and practitioners. New methods for quality evaluation need to be defined, or it is necessary to specialise the various techniques defined by Software Engineering [3] or by the Human Computer Interfaces community [13][16], in order to address the peculiar features of hypermedia.

This paper will try to contribute to this discussion, by addressing the problem of hypermedia quality from a *design* perspective. With the term "design", we refer to the "external level" of a hypermedia application, rather than to the design of implementation data structures and code. In our intended meaning, design includes all the aspects that are perceived by the end user, such as content organisation, behaviour of the various media, functionalities offered to the user, lay-out of the various hypermedia elements. The main thesis of this paper is that assessing the design quality is one of the steps towards

° Authors' Address: Department of Electronics and Information, Piazza L. da Vinci 32, 20133 Milano, Italy. Phone: +39-2-23993520; Fax:+39-2-23993411; e-mail: garzotto@elet.polimi.it

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

evaluating the overall quality of a hypermedia application. Design evaluation can be performed *before actually implementing* an application, based on a set of specifications or on a prototype version. It can also be performed *after implementation*, by abstracting the design specifications from a precise analysis of the final running application.

In our approach, design-oriented evaluation does not involve end users. As such, it is complementary to Human Computer Interfaces methods that evaluate quality factors (such as usability) by having a number of test users use the system to perform a representative set of tasks, and by surveying their behaviours or their comments.
In addition, we does not address problems related to the technical quality of the software (e.g., efficiency, maintainability, reusability, or portability); as such, design-oriented evaluation is also complementary to "classical" software engineering approaches.
However, the framework proposed here for analysing hypermedia design quality has been inspired by software engineering methods. We have adopted the assumption (commonly acknowledged for software metrics [3]) that in order to evaluate or to measure something, it is necessary: 1) to define which "entities" or "objects" are the subject of evaluation or measurement; 2) to define a language to describe them; 3) to identify the key factors that have a special interest for the selected objects. 4) to decompose these factors into lower level attributes that can be directly analysed and possibly measured.
The next section will define the "entities" of interest for assessing design quality; it will also introduce the terminology of the design model we use to describe them (HDM - Hypermedia design model), and will define a set of design quality attributes. Section 3 will exemplify these concepts, by analysing the design of a commercially available hypermedia application. Section 4 will draw the conclusions.

2. Design Constituents and Quality Factors

We have identified three main classes of "entities", or "design constituents", which contribute to the design of a hypermedia application and can be subject of evaluation: *structures*, *behaviours*, and *presentations*. Structures, behaviours, and presentations, can be described at two levels, *in-the-large* (i.e., at the global level) and *in the-small* (i.e., in detail).
Our focus in this paper will be on structures and dynamics. In section 3.1, they will be described precisely using the Hypermedia Design Model - HDM, that allows to standardise the description terminology[4]. Presentation objects will be discussed very shortly.

The key attributes that we have considered for evaluation of design constituents are *consistency*, *predictability*, and *self-evidence*. We will combinine these factors with the two "dimensions" under which hypermedia constituents can be described, and with each class of design objects, thus obatining a set more fine-grained attributes, discussed in section 3.2

3.1 Structures

The structures of an application concern the *organisation* of the pieces of information contained in the application.

[4] The primitives of HDM are introduced very synthetically, and the interested reader is referred to the bibliography [2][4][6] [7] for a more complete description.

Structures in-the-large

To define the *structures in-the-large* of a hypermedia application, we use the HDM concepts of *entity*, *entity type*, *collection, link*, and *link type*,

An *entity* groups together a number of information segments in a granule that corresponds to some real-world conceptual or physical "object" (e.g., the town "Venice" or the subject "Italian Landscape"). The constituents of an entity are called components, discussed later. The arrangement of the components, within an entity, can vary, according to the topology of the entity itself: they can be organised in a sequence, in a set, in a tree, etc. Entities that correspond to domain objects of the same class are grouped in the same *entity type* (e.g. "Town"), and all share the same topology.

A *collection* groups together a set of objects, called *members* of the collection [5]. Members of a collection can be entities, constituents of entities (i.e., components or single nodes) or other collections (*nested collections*). Objects can be organised into collections in order to represent a taxonomy, according to some objective criteria, or they can be grouped together simply to improve application readability and help the user to better find his way around in the application. A collection also holds a distinguished node, associated to the collection itself and called *collection-node*. The purpose of the collection-node is of illustrating the content of the collection itself, and also to provide access paths to the members of the collection.

Links (also called *webs* in HDM) are connections among nodes, entities, or collections. HDM defines various categories of links: *structural links*, *applicative links, index links, and guided tour links. Structural links* tie together the different nodes belonging to the same entity; they are, in general, implicitly defined by the structural features of the entities (tree, sequence, etc.). *Applicative links* connect together two "objects" - a node, an entity, or a collection - according to some intended relationship. Links that have similar meaning and connect objects of the same category or type are grouped under the same *link type*. *Index links* connect the collection node to the members, and vice versa; *Guided tour links* interconnect the different members. Index and collection links are collectively called *collection links*

Structures in-the-small

To define the *structures in-the-small* of a hypermedia application, we will use the HDM concepts of *slot, frame, node, component.*

A *slot* represents an atomic piece of information. It can be of a simple type, such as an integer, or of a complex type, such as, for example, a video synchronised with sound [8]. A *frame* is an aggregate of slots, put together in order to create a cohesive granule of information. Frames with similar slot structure are grouped in the same *frame type*. A *node* is a navigational unit. In HDM, a node is always associated to a frame that represents the node content. The type of a node is defined as the type of its frame. A node is, in general, interconnected to other nodes (while frames only aggregate slots, with no connections to other information granules).

A *component* is a group of nodes that constitute a logical unit within a given entity. Components grouping nodes of the same type belong to the same *component type*.

3.2 Behaviours

We use the term "behaviour" to indicate both two different aspects: *behaviour-in-the-large*, that refers to the global dynamics of an application, i.e., the way the user can move around across the different pieces of information, and *behaviour-in-the-small*, that refers to local dynamics, i.e., to the interaction of the user with each piece of information.

Behaviours in-the-large

In modern hypermedia applications, *behaviours in-the-large* amount 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. We will examine in this paper only the dynamic

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

features concerning *navigation* aspects, which are the most specific of hypermedia applications. If passive media only are considered, activating a link from its source leads to the activation of its destination and the de-activation of its source. This basic concept needs some extensions when active, time-dependent media are taken into account. If active slots were being played in the source node, for example, it must be defined the state in which the source itself is left (e.g., the original state, or the state reached at the moment of departure from the node) [9]. 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]).
Different categories of links induce different *navigation patterns*. The natural way of exploring an entity is to follow the structural links for *Structural Navigation. Applicative Navigation* is the most traditional (in the hypertext sense) style of navigation, and corresponds to traverse applicative links. It allows the user to move across different entities or collections. As far as navigation within a collection is concerned, there are two styles, that can be intermixed: *index navigation* and *guided tour navigation*, collectively called *Collection Navigation*. In *index* navigation, the collection node is used to access a member via collection links, and from a member it is possible, in general, to navigate back to the collection node. In *guided tour* navigation, it is possible to move from one member directly to another member (again following collection links) without need to traverse each time the collection node [14]. A typical combination of index and guided tour allows the random selection of a member, and from this the subsequent exploration of other members.

Behaviours in-the-small

Behaviours in-the-small concern the dynamics of slots and frames, and their interaction with the user. Slot dynamics and interaction are strictly related to the type of the slot itself. If the dynamics of an integer or a string slot is virtually not existent, the behaviour of and the interaction with multimedia value can have complex features. Zooming on pictures, playing control over video, sound, or animation, etc., are all examples of hypermedia interaction at slot level. The dynamics related to a frame consists in coordinating the presentation of its slots. If a single active slot is included in a frame (an audio track, for example), it may be played by default, or it might be played upon request, as an option to the reader. When several active media slots are involved (say multiple sound tracks, videos, animation, etc.), there are more complex issues concerning parallel presentations with proper synchronization, or well-sequenced presentations.

3.3 Presentations

The term "presentation" refers to how the content, the structures, and the behavioural features of the application are "shown" to the reader [11].

Presentations in-the-large

Presentations in-the-large are visual structures that present in-the-large structures or behavioural in-the-large features (such as navigation) to the reader. Examples of presentation-in-the-large objects, corresponding to in-the-large structures, are diagrams showing the high-level schema of an application (i.e., visualizing the application link types connecting the various entity types), or the structures of entity types, or the organisation of members within collections. Examples of presentation-in-the-large objects, corresponding to in-the-large behaviours, are visual elements showing the set of navigation links outgoing from nodes of a given type.

Presentation in-the-small

Presentations in-the-small are visual objects that are responsible for the lay-out of individual granules of information, and for visualizing to the reader the functionalities to interact with in-the-small structural

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

elements. Presentation elements in-the-small are for example visual pages, fields, and multimedia control buttons, etc.

3.2 Key Factors for Design Quality

As it happens for traditional software, the term *quality* is extremely vague also when applied to hypermedia products, and "like beauty, is very much in the eyes of the beholder" ([3], p.222). Rather than a general definition of hypermedia quality, what concern us here is what specific *design attributes* can impact on the *product attributes* that are of interest to the hypermedia user. We will focus here on a specific product attribute, *usability*, and we will discuss a set of design factors that can affect it.

Paraphrasing Jacob Nielsen's ([12], p.25), hypermedia usability can be defined "as the question of how well users can use the functionalities of a hypermedia application".
Usability is traditionally viewed as made up of a number of more "fundamental" components [12][1]: *learnability* (that refers to the speed of learning, before independent use of a product is possible), *comprehensibility* (that defines how much an application, its content and its functionalities, can be easily understood by the user), *memorability*
(that measures how easy is to remember a system, after some period of not having used it), *handling ability* (that refers to the efficiency of working with a hypermedia product when trained), *niceness* (that describes the user satisfaction when using the system).

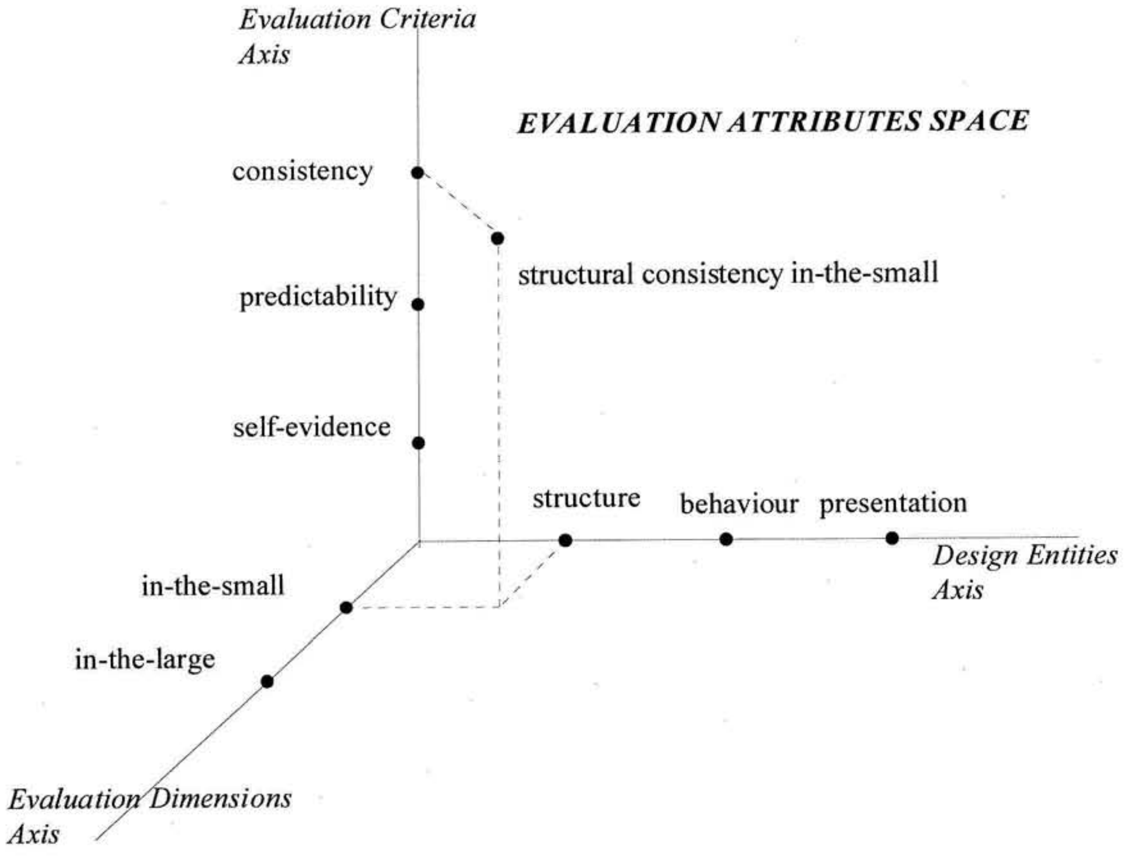
There are at least three "lower level" factors that are believed to impact on learnability, memorability, and handling ability [12][13][16]: *consistency*, *self-evidence*, and *predictability.*
Consistency can be synthesized by saying that "conceptually similar elements should be treated in a similar fashion, while conceptually different elements should be treated differently".
Self-evidence denotes the capability, for the reader, of guessing the meaning and the purpose of whatever is being presented to him for the first time.
Predictability expresses the possibility, for the user, of "guessing" the outcome of an operation (by analogy of what he has already experienced), and can be regarded as an extension of the concept of self-evidence.

There are no rigorously defined "theoretical" measures of these attributes. We can achieve more finer grained attributes, by applying consistency, self-evidence, and predictability - to the design objects discussed in the previous section. By combining these three evaluation parameters with the three classes of design constituents (structure, behaviour, and presentation) and with the two levels of design (in-the-large and in-the-small), we get eighteen finer-grained attributes. This situation is depicted in the following figure 1, where lower level attributes can be imagined as points on a discrete 3D space (see for example "structural consistency in-the-small).
The definition of a significant set of metrics to measure these attributes is the following step in order to make more rigorous and objective the evaluation process. This is a topic which we are currently exploring, and will be the subject of a future paper.
Even without a formal metrics, however, we believe that the lower level parameters that we have introduced, may form the basis to specify hypermedia quality in a measurable form, or at least can help to organise more systematically the process of hypermedia application analyisis, and of defining more precisely usability tests.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

To exemplify the use of the overall approach, the next session will provide an example of post-implementation design analysis, applied to a commercially available hypermedia applications[5].

Figure 1 - The Space of Design Evaluation Attributes



3. Examples of design-oriented analysis and evaluation

This section will analyze the design of a commercially available hypermedia applications in the tourism domain, called "Italia", by Touring Club Italiano and Opera Multimedia. We will analyze consistency, self-evidence, predictability, of its structure and behaviour, omitting, for lack of space, the evaluation of its presentation aspects.

[5] not developed by the authors of this paper

"Italia" is a truly enjoyable application, that allows the user "to discover and learn about the natural, artistic, and historic wonders of Italy" [15]. Its content is relatively complex, since it concerns over 250 places and 600 masterpieces in Italy, described by over 800 photographs (mostly full screen), hundred of pages of text, several short video clips, music, and animations across which the user can interactively navigate.

We will show that a design model, such as HDM, can be used to describe the design of an application that was developed with no knowledge of the model itself, and that the systematic description thus obtained can be helpful in detecting possible usability problems within the application.

"Italia" is organized in two main sections, that address different goals and have different access styles and interaction facilities. One section is called "Guide to Italy" and is more "information search" oriented. The user who wants to learn about Italy, or to plan a tour, can "visit" one of Italy's twenty regions, looking at regional features (wines, cuisine, art, nature). In addition, "Italia" offers, in the section "Discover Italy" a range of different tools and games containing video clips, slide shows, and animated sequences on topics of particular interest, that allow "discovery Italy" in a more playful (and simple) way. For lack of space, we will analyse only the "Guide to Italy" section of this application.

3.1 Structure in-the-large and in-the-small of "Italia"

The "Guide to Italy" section of "Italia" concerns artistic, historical, or natural places and regions of Italy[6]. We have modeled the organization of these contents by defining *two entity types* and a number of *nested collections* (i.e., collections of collections), closely interconnected each other, and accessible through a number of hierarchical indices. The only links available in "Italia" are structural links (to navigate within entities) and collection links (to navigate within and across collections).

Entity Types

Entities of "Italia" represent physical places located in Italy, that have some historical, artistic, or natural interest. The distinction between entities of the two types, called *"Minor Place"* and *"Major Place"*, depends from the amount of content associated to them and in the organization of such content, i.e., in the *structure* of the entity type.

An entity of type "Minor Place" is defined by a linear sequence of homogenous components of type Place. A component of type Place consists of a single frame of type F-Place. An F-Place frame contains a number of slots, including a picture slot, a text slot with a short descriptive text commenting the picture, and some text slots storing the name of the place or the monument, its region, the subject of the picture, etc.

An entity of type "Major Place" is structured as a tree of components. The root component has the purpose of allowing the user to select one of the three components History, Surroundings, or Tour of the Town. The History component contains one frame that consists of a sound slot (a voice explaining the town history), a title label with the town name, and an animation slot storing an animated sequence of images presenting monuments, places, or personages that are relevant for the town history. Surroundings has a sequence of frames, each one of type F-Place (described above). The component Tour of the Town has a frame type Map storing a picture slot with the map of a town, and a label slot with the town name. This component is connected to other components, each one with a frame of type Map showing a specific area of the town. Each of these components stores a sequence of frames of type Place.

[6]"Italia" is provided both in Italian and in English, within the same CD-ROM. The two versions are isomorphic and independent, and we will analyze the English one.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

Collections

Entities are organized in various collections, according to the combination of various three criteria: Region, Topic, and Subtopic. Topics are Art and Nature. Subtopics of Art are Art Periods: Prehistoric Times, Etruscans, Ancient Greeks, Ancient Romans, Middle Ages, Renaissance, Baroque, 18th-20th Centuries. Subtopics of Nature are Seaside Resorts, Mountain Resorts, Countryside, Parks, Lakes, and Spas. Collections have complex *nested structures*.

- *Lowest level* collections
  They are sequences of places organized by Region *and* Sub-topic (e.g., "Lake Places in Tuscany", or "Renaissance Places in Veneto"). The frame in the collection node is of type "F-region", and consists of a picture slot (a region map), and four text slots describing the region, its wines, and its local cuisine.
- *Second level* collections
  They have as members lowest level collections.
  There are two categories of second level collections, based on two different grouping criteria: *by Subtopic,* or *by Topic and Region.*
  Collections "by-subtopic" group lowest level collections that concern the *same subtopic*. Examples are the collections "*Renaissance Art* in Italy", grouping all collections of type "*Renaissance* Places in *R*", or "*Lakes* in Italy", grouping all collections of type "*Lake* Places in *R*" (R denotes a region). In collections of this kind, the frame in the collection node is of type "F-country", and consists of a picture slot (the map of Italy) and a text slot with a short presentation of Italy.
  A second level collection "by-region-and-topic", groups all lowest level collections that share the same Region, and concern a Subtopic of the same Topic. An examples is the collection "*Art* in *Veneto*" (grouping all collections of type "AS in Veneto" where AS is a subtopic of Art). In collections of this kind, the frame in the collection node is of type "F-region" (defined above).
- *Third level* collections.
  They have as members second level collections.
  There are several third level collections. One is called "Art in Italy", grouping all second level collections of kind "AS in Italy", where AS is a subtopic of Art (e.g., Renaissance, Baroque, etc.). "Nature in Italy", is defined in a similar way (for subtopic of Nature). In both cases, the collection node is of type "F-country".
  Furthermore, for each region R there is a third level collection "R by-Art&Nature" that groups the two second levels collections concerning the region R, and Art or Nature, and has a collection node of type "F-region".
- *Fourth level collection*
  There is one fourth level collection, called "Regions", whose members are the third level collections of type "R by Art&Nature"
- *Fifth level collection*
  The only fifth level collection - called "Master" - groups the two third level collections "Art in Italy" and "Nature in Italy", and the fourth level collections "Regions"[7].

[7]In addition to the above collections, there is simpler collection which is called "Index". It organizes all places (entities of type Minor or Major Places), relevant monuments (components of Places), and regions (third level collections of kind "Region R by Art&Nature"), in an alphabetic order.

Evaluation Considerations
Our evaluation of the structures of "Italia" is that, while entity types are relatively simple and therefore rather predictable and self-evident, collections very sophisticated but less predictable, due to the many levels of nesting and to the difficulty, for the normal user, to immediately grasp the many combinations of organization criteria.

3.2 Behaviour in-the-large of "Italia"

Let us now proceed with discussing the navigation facilities of "Italia". As already mentioned, there are no application links in this application, and therefore the only navigation patterns to be considered concern structural navigation and collection navigation.

Structural Navigation
Structural navigation within an entity of type "Minor Place" is extremely simple, and consists in moving backward and forward, via Next and Previous commands, along the sequence of nodes corresponding to the only component of the entity. Structural navigation within entities of type "Minor Place" is induced by the tree-like structure of this entity type, and is slightly more complex. From the node of the root component, the user can select one of the children components - History, Surroundings, or Tour of the Town, and access its first node. Within nodes of the Surroundings component, the user can navigate in a strictly linear fashion. From the node of component Tour of the Town (showing a town map), the user can select a child component concerning a specific area of the town, and access the sequence of nodes describing the selected area, in a linear fashion. In addition, the user can jump to the root node from any of the above visited nodes inside an entity.

Evaluation Considerations
Structural navigation in Italia is very usable. Structural links are consistent and predictable. Their presentation is intuitive and make self-evident their use. The lay-out of link placeholders (buttons) highlights the distinctions between links across different components with respect to links within a single component, and makes structural navigation quite self-evident. The richness of links provides quick ways of exploring the various entities. A minor burden is created by the purely linear way to scanning components of type Place: when a Place component stores a large amount of information and has many links, the user would probably expect to be able to directly jumps to the last or the first member. The lack of these direct links has the advantage of keeping the presentation simple, but it can be disappointing for the expert user, and can reduce the efficiency of use.

Collection Navigation
Collection navigation follows different patterns depending on the kind and level of nesting of a collection. The simplest collection is the Index, which allows the user to select any topic he is interested to, jump there, and return to the index at any time (and from any node of the application).
Navigation on all other collections is significantly more complex is , and complexity increases with the increasing nesting depth.
Within a lowest level collection (e.g., "Lake Places in Tuscany"), user can navigate both in index and in guided tour mode, according to three patterns: the user can directly access a member from the collection node, or can move from a member to the *next* and *previous* one, according to the order established by the collection, or can return from any member to the collection node. Direct access to all members is provided on the collection node, by showing the members (i.e., the places) on a map. Forward and

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

backward scan in the collection is provided by the left and right arrowhead buttons on the members, which are located on the bottom of the screen. If the current member is the last one of a collection, the right arrowhead button is visible but it is inhibited. For example, the user who is located on a Place (say, "Venice") within collection "Renaissance Places in Veneto", can move from Venice to Verona by following a "next member" link, return to the collection node which shows all the Renaissance places in Veneto, and directly select a different place (e.g., Vicenza).

Evaluation Considerations
Overall, navigation in lowest level collections is consistent, self evident, and predictable. We have found only a minor inconsistency in "Countryside in Veneto", where members are organized in a circular list, and going Next from the last member takes the user to the first member (instead of being prohibited, as it is in general).

The dynamics of navigation within second level collections is slightly different. Navigation within second level collections based on Subtopic, combines index and guided tour modes. From the collection node of "*Renaissance* Places in Italy", for example, the user can select a specific Region, e.g., Veneto, and access the lowest level collection "*Renaissance* Places in Veneto". From here, he can access the adjacent members by navigating forward or backward, or return to the collection node of "*Renaissance* Places in Italy". Differently from lowest level collections, members of second level collections of this kind are arranged in a *circular list*: once he reader has reached the last member collection, he can continue forward and access the first member collection (arranged in alphabetic order, base on the Region name). For example, from "*Renaissance* Places in Veneto" (Veneto is alphabetically the last Italian Region) the link Next takes the user to "*Renaissance* Places in Abruzzo", since Abruzzo is the first Italian Region in the alphabetic order.

Evaluation Considerations
The behaviour of second level collections is slightly inconsistent with respect to lowest level collections, and is not predictable nor self-evident at all for users who have experimented the navigation on lowest level collections.

Let us now consider navigation in second level collections based on Region and Topic, such as "*Art* in *Veneto*". From the collection node, the user can select a member, i.e., a lowest level collection, and from any member he can directly access any other one. For example, once the user is on the collection node of "*Art* in *Veneto*", he can select any lowest level collection in the set "Prehistoric Times in *Veneto*", "Ancient Romans in *Veneto*", "Middle Ages in *Veneto*", "Renaissance in *Veneto*", etc. From "Renaissance in *Veneto*", for example , the user can directly switch to "Prehistoric Times in *Veneto*".

Evaluation Considerations
Lack of linear scanning of members of second level collections of the second category is inconsistent with respect to navigation in second level collections of the first category, and is slightly disorienting when the user accesses a collection of this kind for the first time. This inconsistency is however compensated by a good presentation (which leaves the list of all lowest level collections always visible from the root and from all members, consistently in all collections of this kind).

The same navigation pattern above discussed is provided for all other collections at higher levels, with the exception of the Master collection. The collection node of the Master collection is only used to select

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

a collection among “Regions”, “Art in Italy”, “Nature in Italy”. There is no way to return to this node from its constituents collection (or from their members, whatever the level).

Evaluation Considerations
On the one hand, "Italia" collection navigation is extremely rich and powerful for an expert user. On the other hand, the use of the many collection navigation patterns may be not self-evident for an average user. The main problems are related to the depth of the collection hierarchy (four levels), the different dynamics of collections, i.e., the different navigation patterns available at different levels, and the fact that the user can reach the leaves of the collection hierarchy (i.e., the actual content about Places) in several different ways. Collection navigation is complicated by the fact that the user is allowed to jump across non adjacent collections in the hierarchy, skipping some levels, and sometime in a context dependent way. For example, from the collection node of “Renaissance in Italy”(second level), the user is allowed to jump to the collection node of “Nature in Italy” (third level) as well as to the collection node of “Regions” (fourth level). If the user is on the collection node of “Baroque in Veneto”, and selects Venice (member of a lowest level collection), he can activate a “jump” and go to the collection node of “Baroque in Italy” (third level collection).

4. Conclusions

The software engineering community has defined a number of "quality models" [3] to help establishing the quality of a generic software product. These models identify the key factors of software products that have a special interest for the end user; these factors are analysed by decomposing them into lower level attributes called "quality attributes", that can be directly analysed and possibly measured.
In this paper, we have specialised this approach to the problem of hypermedia quality in order to address the specific features of hypermedia. We have provided a framework for design-oriented hypermedia evaluation that defines a set of design objects and a set of fine-grained attributes of these objects that can impact on at least one fundamental quality factor, i.e., usability. Our current research is exploring a set of metrics to measure these attributes, in order to make more rigorous and objective the evaluation process.

Even in this preliminary phase, the approach discussed in this paper can be used to make more systematic the process of *analysing* a hypermedia application, and of elaborating informal considerations on the quality of its design . The results of the analysis can be used, both during the design/specification phase, and after its final implementation, to detect potential design weaknesses of an application.
In addition, our approach can be used when user testing is not feasible, in order to identify potential usability problems , or can be adopted before user-testing, to get some feedbacks and to better focus the usability tests.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

References

1. Bearne M., Jones S., Bearne J. S-F. M., “Towards Usability Guidelines for Multimedia Systems”, In *Proc. ACM Multimedia ‘95*, S. Francisco, CA, Oct. 1995
2. Cavallaro U., Garzotto F., Paolini P., Totaro D. "HIFI: Hypertext Interface for Information Systems". In *IEEE Software,* Nov. 1993
3. Fenton N.E., "Software Metrics: A Rigorous Approach", Chapman & Hall, 1991
4. Garzotto F., Paolini P., Schwabe D. "HDM - A Model Based Approach to Hypermedia Application Design" In *ACM Trans. Off. Inf. Syst.*, 11 (1), Jan. 1993
5. Garzotto F., L. Mainetti, P. Paolini "Adding Multimedia Collections to the Dexter Model". In *Proc. ECHT'94 - ACM Conference on Hypermedia Technology,* Edinburgh, UK, Sept. 1994
6. Garzotto F., Mainetti L., Paolini P., "Hypermedia Application Design: A Structured Approach". In *Designing User Interfaces for Hypermedia,* W. Schuler, J. Hannemann, N. Streitz (eds.), Springer Verlag, 1994 (in press).
7. Garzotto F., Mainetti L., Paolini P. "Navigation in Hypermedia Applications: Modeling and Semantics". In *Journal of Organizational Computing* - to appear
8. Gibbs S., Breiteneder C., Tsichritzis D., "Data Modeling of Time-Based Media". In *Proc. ACM SIGMOD*, Minneapolis (USA), May 1994
9. Hardman L., Bulterman D.C.A., Van Rossum G., "Adding Time and Content to the Dexter Model". In *Comm. ACM, 37 (2),* Feb. 1994
10. Hodges M., Russell M. Sasnet, "Multimedia Computing - Case Studies from MIT Project Athena", Addison Wesley, 1993
11. Kahn P., "Presentation Design Issues for Hypertext and Multimedia Publications", submitted for publication
12. Nielsen J., "Usability Engineering", Academic Press, 1993
13. Preece J., "Human-Computer Interaction", Addison Wesley, 1994
14. Trigg R.H., "Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment". In *ACM Trans. Off. Inf. Syst.* 6 (4),1988
15. Touring Club Italiano & Opera Multimedia, "The Italia CD-ROM, 1994
16. Shneiderman B., "Designing the User Interface", Addison Wesley, Mass. 1987

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

A Conceptual Framework for Hypermedia Design Methodologies

Manfred Thüring

empirica
Communications and Technology Research
Oxfordstr. 2
D-53111 Bonn
Germany
Tel.: ++49-228-985300
email: manfred@emp-d.uucp

1 Introduction

The design of hypermedia applications is a problem of high complexity because there are so many different aspects to be considered. The only way to reduce this complexity - and thus make the problem manageable - is a methodology that provides a structured view on hypermedia design and guides developers through the various steps of creating an application. Although this methodology should reduce complexity by structuring, it should also address all important aspects of the problem. In the following, I will outline a conceptual framework by proposing a number of topics that a methodology for commercial hypermedia design should address. I will discuss three major topics:

- the roles of different **stakeholders** involved in the development process,
- the relation of application **design** to application **marketing** and **usage**,
- the **products, procedures and criteria** that can be distinguished in hypermedia design.

With respect to hypermedia development one might argue that the first two topics are beyond the scope of a *design* methodology - and this argument is certainly true for developing applications in the context of pure research projects. For *commercial* development however, it is of crucial importance that developers are aware of the influence and potentials of different stakeholders as well as of the consequences and requirements resulting from marketing and the successful (or unsuccessful) usage of their product. I therefore claim that the design of applications which have to survive in a real market cannot ignore these issues.

2 Stakeholders involved in hypermedia development

Stakeholders can be best characterized by their particular perspective on application design and the first of these perspectives can be best described by the simple question:

- *"How should a hypermedia application be developed?"*

Obviously, this question represents the perspective of engineers who have to design and implement the application. Answers to that question are in the first place technology driven, e.g., the **functional properties** of the application are specified, different **methods** of how to build the application may be considered and **tools** that will adequately support a chosen method have to be selected.
This **engineering perspective** is of central importance for the design of any application - and it is probably the perspective which the majority of developers and researchers are most familiar with. Therefore, it has been dominating research and development until now contributing to technical progress and leading to considerable innovations in a variety of technologies that are basic for hypermedia design (e.g., multimedia databases, graphical user-interfaces, information retrieval, etc.). On the other hand, this perspective alone does not guarantee a successful hypermedia product. When we design an application we must be careful not to be carried away by our enthusiasm

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

of what we *can* do, neglecting the issue of what we better leave alone. In order to create an adequate application, what is technically *feasible* must be constrained by what is actually *usable*. This takes us to the second focus of hypermedia design which enhances the engineering perspective by an important aspect:

- *How should a hypermedia application be developed for particular kinds of usage?*

This question involves the actual user of the application and requires the anticipation of its future kinds of employment. Answers to that question should provide different user profiles, describe which tasks will be supported by the application and specify the organizational constraints encountered in the environment (e.g., the business organization) in which it will be used. **User profiles** should distinguish between different user types with respect to information needs, expertise, role or position, etc. **Task descriptions** should - at least - specify the materials that are used, the activities that are performed, the tools that are employed, the criteria that must be fulfilled and the results that are accomplished. The **organizational constraints** should address a number of heterogeneous issues, such as the physical setting in which the application will be used, the technical infrastructure into which it has to be integrated, the security measures and access rights which it must account for, etc.

The **usage perspective** extends - but does not replace! - the engineering perspective and leads us from technical feasibility to practical usability. Although this perspective has a well-known and elaborated background in terms of human factors research, its impact on professional hypermedia design is still comparably small. However, a number of new methods for task analysis, requirement specification and user-oriented evaluation have been proposed in the last few years and are currently gaining on importance (compare Nielsen, 1993). Integrating these approaches into the overall course of development will certainly increase the quality and acceptance of hypermedia applications - but again we can ask if this is sufficient in a commercial context. Features that may appear as desirable from a user's point of view may entail considerable costs which surpass their actual benefits. Therefore, what is well *usable* must be constrained by what is *marketable* for a specific segment of the hypermedia market. This consideration is reflected in the third focus on commercial hypermedia design which adds an economic aspect to our initial question:

- *How should a hypermedia application be developed for particular kinds of usage that are required in a specific market?*

When answering this issue it must first be decided if the envisioned application is marketable *at all* - depending on the demand, the degree of competition, the developmental costs, etc. This general decision determines if the application is targeted at a broad **mass market**, a **specific market segment** or even a **particular customer**. These considerations result into a number of crucial product features - or key selling points - that must be realized to ensure commercial success.

The **marketing perspective** adds another flavor to hypermedia application design and takes us beyond technical and usability considerations. It delimits the space of solutions that are technically feasible and highly usable to those solutions that are also commercially exploitable. These constraints arise from the knowledge of the market demands and of other products that can be regarded as competitors. Only if they impact on the actual design the final product will have a chance on the market. The different foci discussed so far are summarized in figure 1.

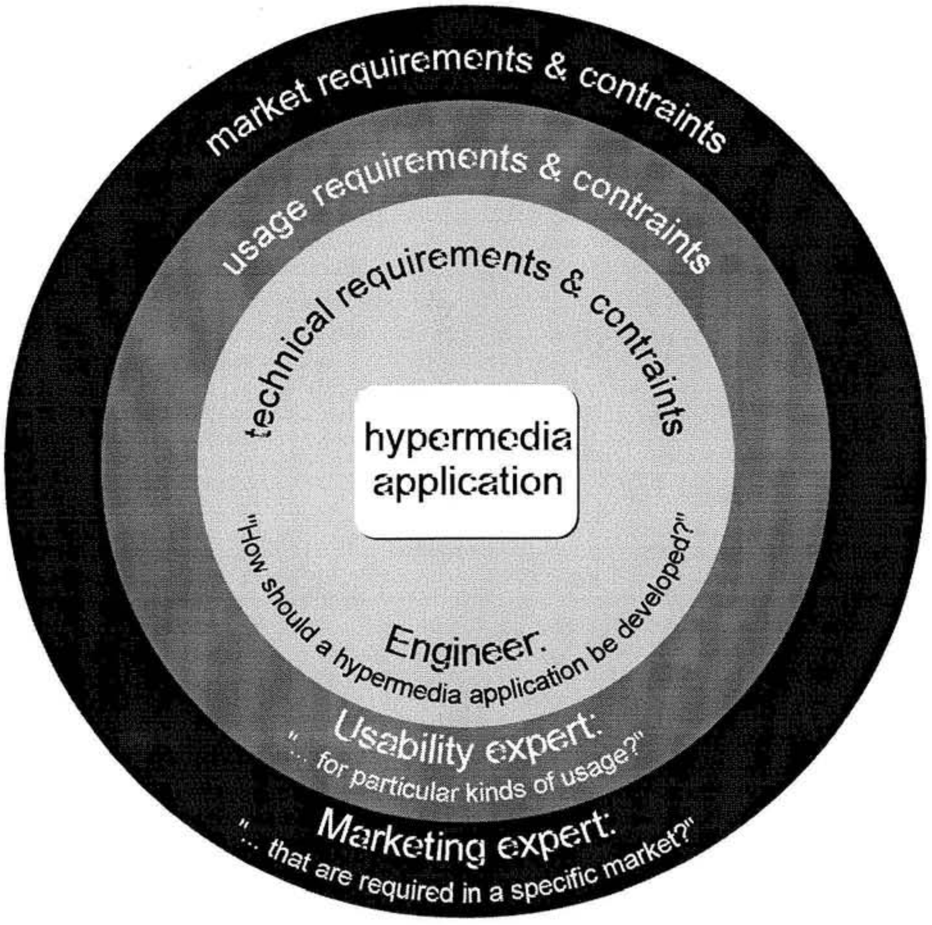

market requirements & contraints
usage requirements & contraints
technical requirements & contraints
hypermedia application
"How should a hypermedia application be developed?"
Engineer:
Usability expert:
"... for particular kinds of usage?"
Marketing expert:
"... that are required in a specific market?"

Figure 1: Stakeholders's perspectives, requirements and constraints

There are at least three types of stakeholders involved, i.e., the engineer, the usability expert and the marketing expert. Each of these represents a particular focus on the design problem and leads to three kinds of requirements that can be regarded as technical, usability and marketing constraints for the final solution. Note that this view postulates a **cooperative methodology** according to which the three types of stakeholders participate in the development process and - to different degrees - contribute to the product. While engineering issues certainly form the kernel of such a methodology, issues of usability and marketing must not be neglected. Instead, all three foci must be integrated into a common perspective and must impact on the course of design and its final outcome.

3 The relation of application design to application marketing and usage

All three perspectives should influence design in a commercial context. To specify in more detail how each of them might contribute to the success of a hypermedia application it is suitable to discuss their role in three phases that can be distinguished in the overall development procedure:

- preparation,
- development and
- introduction.

In the first phase, all information required for the application must be collected and the development must be planned in terms of products and deadlines. In the second phase, the actual development takes place resulting into the final product. In the third phase, the product must be introduced into the market, installed for customers and maintained during future use. To specify these phases in more detail, they can be matched against the three stakeholders' foci which leads to the 3*3 matrix in table 1. The cells of the matrix represent the contributions required from each type of stakeholder in each phase.

perspective / phase	preparation	development	introduction
marketing	(1) market analysis	(4) marketing concept	(7) marketing
usage	(2) user analysis	(5) usage model	(8) user service
engineering	(3) technical analysis	(6) application	(9) installation

Table 1.

A lot could be said to each of the cells in table 1, but in this paper I can only present a brief outline for each topic.

(1) The **market analysis** should describe which kind of market will be addressed (e.g., a specific economic sector, such as manufacturers of trucks). It should estimate the size as well as the saturation of the market and propose a set of key selling features that the application should have to be attractive for organizations that are targeted by the product. Thus the market analysis is the basic for deciding whether the application is worth to be developed with respect to commercial chances. If the decision is positive, key selling features serve as market requirements that must impact on the design of the product.

(2) The **user analysis** should be performed if the market analysis revealed that the application is worth a try. In this case, more detailed information about those organisations and users is required who are potential purchasers of the application. This encompasses a set of user profiles, tasks and organisational features (e.g., technical infrastructure, degree of spatial distribution, etc.) that are typically encountered in these organisations.

(3) The **technical analysis** should find out if the envisioned hypermedia application is feasible, i.e., if its functionality can be developed under the existing financial and timely constraints. If this is the case, it should decide about the methods and tools to be used as well as about the workplan.

Together the three analysis should lead to two results: (a) the decision if the application should or should not be developed and (b) the market, usage and technical requirements that must be met by the product. Therefore, they should be accomplished to a certain extent before the development phase is started. Nevertheless, their results should not be regarded as final since experiences during the future phases may lead to revisions with respect to all three kinds of analysis. A methodological background for these three types of analysis is proposed by Bødker et al. (1993) as well as Thüring et al. (1994). It includes three kinds of checklists (work-oriented, technical and market-oriented) which were developed in a research project on Computer Supported Cooperative Work, entitled EuroCODE. In that project, the checklist were used to describe user requirements, market consideration and informal technical specification which have proven useful at an early stage of design.

(4) The **marketing concept** should be built from the market analysis and should receive continuous feedback from application development. This requires to develop a more sophisticated strategy for targeting the product, for elaborating its major advantages (key selling features in comparison to competitors), and for preparing against potential factors limiting customer acceptance and uptake. The creation of such a concept is a task probably as complex as application design itself and therefore it cannot be described here in sufficient detail.

(5) The **usage model** should be based on the results of the user analysis, but in contrast to this analysis it should not describe a current state of affairs. Instead, it should outline how the behavior of users, the accomplishment of tasks and organisational features may *change* when the hypermedia application is integrated into working and business procedures. For

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

this purpose, it should provide a set of submodels specifying how the application should be used by particular persons and for particular tasks.

(6) Of course, the development of the **hypermedia application** is in the centre of the whole design process. In order to describe a methodology for this activity, it is necessary to distinguish between (a) different components or products related to the application, (b) a design procedure, e.g., including input/output relations, activities, methods and tools, and (c) design criteria which may be application specific or be derived from existing standards and/or other conventions (see below). As may be apparent from the brief outline of the previous analysis and models, the application is highly related to the preparation phase and considerably influenced by its results.

(7) The **marketing** of the hypermedia application will proceed according to the marketing strategy that has been developed in parallel to the application. Again, much could be said about this topic. In this context though, I will re-emphasise only one aspect: the marketing should stress the key selling factors of the application and provide arguments in terms of cost benefit considerations that are adequate to demonstrate how a user - and in particular a business organisation - will profit from the application, e.g., by greater speed to market, by saving costs or by increasing the quality of their products and services.

(8) The **user service** should provide supporting measures for actually introducing the application into organisations. It should include such services as training and support in case of problems. Depending on the impact of the application even support for "reengineering" current working activities and business procedures may be considered.

(9) The **installation** finally serves to integrate the application into the technical infrastructure of the customer. It may include additional services, such as customizing, maintenance and updating.

Taking a look at these nine brief outlines, one may question if all cells of the matrix should really be addressed in a *design* methodology. The major argument for claiming that this should actually be the case, is that each of the cells is closely related to the design product - either by providing requirements and constraints or by influencing its final (commercial) success. On the other hand, questions of marketing and usability may lie beyond the interests and expertise of many developers. Therefore, I vote for conceptualizing the design process as a **cooperative** activity in which three kinds of experts or stakeholders join their forces: the engineering expert, the usability expert (as "link" to potential users), and the marketing expert.

4 Products, procedures and criteria

A methodology for the design of hypermedia applications should distinguish between three general components:

- the **product** to be developed,
- the **procedure** of development, and
- the **criteria** to be fulfilled by both product and procedure.

For a methodology, each of these components must be specified in detail. In the next three paragraphs, I will outline some ideas in that respect.

4.1 The product

I propose to regard the design of a hypermedia application as the construction of three models:

- The *content model* represents the information units and their relations and is developed in terms of "authoring in the large" - creating structure - and of "authoring in the small" - creating content - (Garzotto et al. 1991). It should be based

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

(a) on input from the market analysis ("what is provided by competing products, what may they be missing?"), and

(b) on input from the user analysis ("which information and which media are currently used by persons who are potential future users of the application?").

- The *rhetorical model* determines which parts of the content model are presented and in which way the information is ordered. According to this view, the rhetorical model can be regarded as a set of filters - or guided tours - which select information and specify in which sequence it is presented. The rhetorical model should in particular use the information of the user model, i.e., it should provide a rhetorical structure for the major user profiles and tasks that have been identified.
- The *presentation model* should define the way in which the application is actually presented. This includes the general layout of the user interface (e.g., card-oriented versus browser-oriented) and the specification of the browsing semantics (e.g., effects resulting from activating a link). Beside specific requirements resulting from the marketing and usage model the presentation should be based on specific rhetorical criteria as well as consider existing guidelines of graphical interface design.

Similar considerations are made by Schwabe and Rossi (this workshop) who distinguish between "conceptual design", "navigational design" and "abstract interface design" and propose an object-oriented methodology. A more detailed - but also comparable - approach is described by Balasubramanian, Isakowitz and Stohr (this workshop) who define a "relationship management model" based on HDM and derive seven steps in which designers should proceed to produce the different parts of an application.

4.2 The procedure

Depending on the material that is available, the creation of the application may constitute a specific type of engineering problem:

1. "Turning text into hypertext": All material is available in a pre-structured format (e.g., an electronic document, a book, a manual, etc.).
2. "Synthesizing hypertext from heterogeneous sources": All (or most of) the material is available, but does not form a coherent entity (e.g., different books, articles, video clips must be aggregated into a hypermedia application).
3. "Designing from scratch": No (or only very few) materials are available: the designer has to create most of the content as well as build a complete new structure on her/his own.
4. "Reengineering an application": There is already a hypermedia application which has to be updated or thoroughly revised.

Each of these problems may require a different kind of procedure. There is, however, an order of activities implied by the models that were described above, i.e., the construction of the content model should precede the rhetorical model which in turn should precede the presentation model. On the other hand, this does not mean that the content model cannot be changed anymore when work on the rhetorical model has started. Instead, revisions must be possible whenever they are suggested by insights at a later stage of design.

Another procedural aspect concerns the phases outlined in chapter 3. The results of the market, usage and technical analysis lead to different requirements and constraints each of which should significantly influence the application. Hence, the three analysis should have reached a sufficient status before the development starts. Moreover, the creation of a marketing concept and a usage model requires that the application itself has reached a status that is detailed enough as a basis for marketing and usage descriptions.

A methodology which is consistent with this view is PHD (Pragmatical Hypertext Design; Schuler and Thüring (1994). PHD distinguishes between several procedural modules, e.g., planning, realization, evaluation and maintenance. The first and the last of these modules can include the activities of the preparation and introduction phase while the modules realization and evaluation can be regarded as part of development phase. One important feature of PHD is that it *recommends* an order of activities, but enables the designer to *deviate* from this order and return to earlier steps when necessary. Thus it does not enforce a particular procedure which must be followed under all circumstances, but accounts for the "opportunistic nature of design". Similar ideas are proposed by Nanard and Nanard (this workshop) who propose "to adapt the spiral model of software production to hypermedia production and to mix analysis, prototyping, specification, development and evaluation in a coherent framework."

Another feature of PHD is that its modules can be specified in more detail and thus allow for structuring activities that result from applying specific hypermedia design "languages", such as HDM (Garzotto et al. 1993). With respect to a methodological framework for design and the models proposed above, such a specification should address the input required for each module, the activities performed, the methods / tools employed and the output produced.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

This leads to a simple template which - when "instantiated" - represents methodological aspects of the development phases. An example is given in table 2.

	input	activities	tools / methods	output
content modelling	market, usage and technical requirements	authoring-in-the-large; authoring-in-the-small	"HYTEA tools and HDM" or "SEPIA and its construction kit"	content model
rhetorical modelling	requirements & content model	design of user / task specific reading structures	see above	rhetorical model (based on content model)
presentation modelling	requirements, content model & rhetorical model	definition of interface layout and browsing semantics	see above	presentation model (based on content model and rhetorical model)

Table 2.

4.3 The criteria

Criteria may concern the procedure as well as the product of developing an application. With respect to the **procedure**, criteria basically result from the workplan and the resources available for designing the application. Such criteria are deadlines for intermediate as well as final results, financial resources, adherence to organisation specific principles etc. More on such issues can be found in the literature on managing software projects. With respect to the **product**, two different kind of criteria can be distinguished:

- Application specific criteria result from the preparation phase and can be characterized as market criteria (e.g., key selling features), usage criteria (e.g., requirements resulting from user profiles and tasks) and technical criteria (e.g., the hardware platform to which the product is targeted).
- General criteria are application independent and may result from different sources, such as existing standards (e.g., for user interfaces), knowledge about human information processing (e.g., the notions of coherence (Thüring et al. 1991) and cognitive overhead (Conklin (1987) for hyperdocuments), rhetorical conventions (e.g., provide an overview of the application structure).

Unfortunately, both kinds of criteria are problematic. The first kind needs to be specified for each new application thus leading to a considerable effort and therefore running the risk of being neglected. The evolution of the second kind of criteria is still in the very beginning, i.e., there are no widely accepted general guidelines for hypermedia design. Both problems often lead to a situation in which the developers of an application have insufficient information about the criteria their application should fulfil. In this case, it is necessary to accompany the design process by developing adequate criteria and by capturing the design rational, i.e., the developers should formulate a set of design issues, answer them in terms of positions and justify these positions by arguments. This approach was proposed by Kunz and Rittel (1970) and an example of it is given by Hannemann, Thüring and Haake (1993) and by Schuler (this workshop).
The purpose of criteria in the design of hypermedia applications is twofold: (a) They serve as goals which the developers can pursue and thus guide their design activities. (b) They serve as points of reference for evaluating the application. Both purposes are consistent with those general criteria that were proposed by Garzotto et al. (this workshop) for the functionality, content, behavior, layout, consistency and symmetry of hypermedia applications. Such criteria should definitely support engineers in finding a technical solution of higher usability -and therefore with better chances for successful marketing.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

In summary, a methodology for hypermedia design ought to address which components should be distinguished for an application, which procedures should be performed to create these components and which criteria or issues, respectively, should be considered for these components.

5 Summary and conclusions

In the preceding chapters, three topics have been discussed which are closely related to different aspects of hypermedia design: (a) the roles of different **stakeholders** involved in the development process, (b) the relation of application design to application marketing and usage in the three general **phases of development**, and (c) the **products, procedures and criteria** that can be distinguished for hypermedia development. The major claims and some points for further discussion can be summarized by the some thesis and open issues.

Ad (a): Engineer, usability expert and marketing expert

Thesis: There are at least three different types of stakeholders involved in the development of commercial hypermedia applications: the engineer, the user (represented by the usability expert) and the marketing expert. Each stakeholder has a different perspective on application design and each of these perspectives must be considered in a design methodology which aims to support the development of products that are technically sound, highly usable and commercially successful.

Issue: How can the knowledge and the contributions of the different stakeholders be integrated into a *cooperative* methodology for hypermedia application design?

Ad (b): Preparation, development and introduction

Thesis: The three stakeholders' perspectives should influence the three general phases of design, i.e., preparation, development and introduction. Each perspective should play a central role in one of the phases: the usage perspective in the preparation phase, the engineering perspective in the development phase, and the marketing perspective in the introduction phase.

Issue: How can the interrelations between the contributions resulting from the three perspectives be specified in detail? Example: Which relations exist between different user profiles and the rhetorical model?

Ad (c): Procedures, products and criteria

Thesis: A methodology for hypermedia design should distinguish between (a) different components of the application in terms of hypermedia specific models (b) procedural considerations in terms of input/output relations, activities, methods and tools, and (c) design criteria in terms of application specific criteria and design guidelines which apply generally.

Issue: How can these components be integrated into a methodology specifying (a) which activities result in which application components, (b) which criteria should guide the development of each application component, and (c) which is the optimal sequence of activities to form an efficient overall design procedure.

Acknowledgements:

Many ideas on design in this paper are the results of discussion in the EuroCODE project. Therefore, I would like to thank all project members. Special thanks are due to Heike Kühn, and Andreas Stammler who have supported me in many ways and to Susanne Bødker, Ellen Christiansen, Jörg Haake and Jörg Hannemann who have deeply influenced my understanding of system design.

References:

[1] Balasubramanian, P., Isakowitz, T. & Stohr, T. (this workshop). RMD: A Methodology for the Design of Hypermedia Applications.

[2] Bødker, S., Christiansen, E., Grønbæk, K., Madsen, K.H., Mogensen, P., Robinson, M., Kühn, H., Robinson, S., Thüring, M., Hinrichs, E., Sørgaard, P. & Hennessy, P. (1993). The EuroCODE Conceptual Framework: Preliminary. Deliverable D1.1 of the ESPRIT Project 6155 CSCW Open Development Environment (EuroCODE).

[3] Conklin, J. (1987). Hypertext: An introduction and survey. IEEE Computer Magazine, 20(9), 17-41.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08

[4] Garzotto, F., Mainetti, L. & Paolini, P. (this workshop). Preliminary Considerations on the Evaluation of Hypermedia Applications.

[5] Garzotto, F., Paolini, P. & Schwabe, D. (1991). HDM - A model for the design of hypertext applications. Hypertext '91 Proceedings, p.313-328.

[6] Garzotto, F., Paolini, P. & Schwabe, D. (1993). HDM - A model based approach to hypertext application design. ACM Transactions on Information Systems, 11(1).

[7] Kunz, W. & Rittel, H. (1970). Issues as elements of information systems (Working paper 131). Berkeley, CA: University of California, Center for Planning and Development Research.

[8] Hannemann, J., Thüring, M. & Haake, J. (1993). Hyperdocument presentation: Facing the interface. Arbeitspapiere der GMD, Nr.784. GMD: Darmstadt.

[9] Nanard, J. & Nanard, M. (this workshop). Some Thoughts and Experience on Approaches and Tools for Designing Hypertext Structure.

[10] Nielsen, J. (1993). Usability Engineering. Boston etc.: Academic Press.

[11] Schuler, W. (this workshop). A design space for hypermedia interfaces.

[12] Schuler, W. & Thüring, M. (1994). Pragmatical Hypertext Design (PHD). Arbeitspapiere der GMD 813. Sankt Augustin: Gesellschaft für Mathematik und Datenverarbeitung mbH.

[13] Schwabe, D. & Rossi, G. (this workshop). From Domain Models to Hypermedia Allications. An Object-Oriented Approach.

[14] Thüring, M., Haake, J. & Hannemann, J. (1991). What's Eliza doing in the Chinese Room? Incoherent hyperdocuments - and how to avoid them. Proceedings of the 3rd ACM Conference on Hypertext (Hypertext '91), San Antonio, Texas, December 15-18, 1991 (pp. 161-177). New York: ACM Press.

[15] Thüring, M., Keil, K., Kühn, H., Stammler, A. & Wiemer, K. (1994). The EuroCODE Framework: Draft. Deliverable D1.3 of the ESPRIT Project 6155 CSCW Open Development Environment (EuroCODE). Bonn: empirica, Gesellschaft für Kommunikations- und Technologieforschung mbH.

Center for Digital Economy Research
Stern School of Business
Working Paper IS-94-08