Chapter III

Towards A Methodology for the Development of Web-Based Systems: Models, Methods & Activities for the Conceptual Design of Large Web-Based Information Systems

Bernhard Strauch and Robert Winter
University of St. Gallen, Switzerland

E-commerce is changing the nature of business. To support 'buying and selling over digital media' for private and corporate Web users, companies need not only appropriate transaction systems, but also new information systems. While the systems development challenge for transaction systems is mostly restricted to separating access channel functionality from business transactions processing and developing systems support for new access channels, systems development needs for information systems are much more challenging since different media and different information source systems have to be integrated, novel forms navigation have to be supported, and information objects become more complex and more volatile.

Based on a review of related work from hypertext design, Web site / intranet design, and database-oriented CASE environments, this paper tries to identify the "essence" of a Web-based information system and proposes an adequate conceptual model. The model is intended to capture not only hierarchical document structure and hypertext semantics, but also dynamic page generation from databases and various approaches to explicit and implicit navigation. It becomes evident that Web-based information system can be regarded as supersets of traditional information systems, thereby requiring conceptual modeling to include various additional features. The proposed model comprises several classes of information objects, various types of associations, activities for the design, and quality checks. For

illustration purposes, the model is applied to an existing Web-based information system. Current Web-based information system development tools are analyzed with regard to the extent to which conceptual Web-based information system modeling is supported.

INTRODUCTION

E-commerce is changing the nature of business. To support 'buying and selling over digital media' (Kalakota & Robinson, 1999) for private and corporate Web users, companies need not only appropriate transaction systems, but also new information systems. The systems development challenge for transaction systems is mostly restricted to separating access channel functionality from business transactions processing and developing systems support for new access channels (e.g. PC via HTTP, mobile phone via WAP, POS terminal via TCP/IP). In contrast, systems development needs for information systems are much more challenging since different media and different information source systems have to be integrated, novel forms navigation have to be supported, and information objects become more complex and more volatile.

Early systems development was dominated by using authoring tools ('editors') to manually edit procedural program code. As a consequence, hand-written code mixing up data usage and functional aspects was difficult to maintain (Martin, 1992). Besides expensive quality control and communication problems among developers, the resulting code suffered from various implementation dependencies, thereby forcing developers to re-do large portions of the development process when technical details (e.g. file structures, access paths) change. A rather similar approach can be observed for early development of Web-based information systems (Rosenfeld & Morville, 1998). By using HTML authoring tools, complex code is created that does not only mix up appearance and contents, but also depends widely on implementation details. Moreover, the utilization of different authoring tools complicates communication between developers. As an example, the following problems usually occur with regard to navigation when different Web-based information systems have to be integrated:

- Navigation is interpreted and implemented in different ways depending on the authoring tool in use. Different tools use identical terms for different concepts or different terms for identical concepts.
- Navigation is not based on user requirements for optimal access to information objects or associations between information objects (e.g. Morville & Rosenfeld, 1999; Richmond, 1999). Instead, implementation details like various frame implementations dominate design.
- As a consequence, similar navigational concepts are implemented (and specified) in different ways so that explicit integration efforts are necessary to identify common structures and implement them consistently.

In order to enable efficient communication between developers and to provide a stable foundation for adopting new technologies, conceptual modeling of Web-

Figure 1: Class of systems

	Focus on Processing of Business Transactions	Focus on Informational Processes
Traditional Technologies (Host; Client/Server via proprietary protocols)	Traditional Transaction Systems	Traditional Information Systems
Internet Technology (Client/Server via HTTP)	Web-Based Transaction Systems	Web-Based Information Systems

based information systems is essential. We understand Web-based information systems as server components of distributed applications which use the HTTP protocol to exchange data between servers and clients ('browsers'). By this definition, the principal problem of Web-based information system development becomes apparent: Although conceptual modeling should be independent of all technical and application dependent details, even the relevant class of application components is defined by technical attributes (HTTP protocol, server functionality) instead of conceptual differences (see figure 1).

Traditional Information Systems vs. Web-based Information Systems

From a business perspective, Web-based systems can be classified as follows (Kaiser, 2000):

- **Business platform:** For e-economy business models like electronic auctions or process portals, certain Web-based systems become the backbone of their operations.
- **Sales and purchase channel:** For traditional business models like mail-order resellers, banks, insurance companies, or producing industries, certain Web-based systems are used to support an additional sales and / or purchasing channel (e-Commerce), while other channels and basic operations are supported by traditional systems.
- **Self service:** In any business, certain Web-based systems can be deployed internally to decentralize selected support processes (e.g. procurement, management of personnel data).
- **Information management:** In any business, certain Web-based systems support the creation, integration, analysis, and distribution of information, particularly for supporting management processes.

While the first three classes focus on the processing of business transactions, the last class characterizes Web-based information systems. To what extent do Web-based information systems conceptually differ from traditional information systems? While traditional information systems focus on querying, reporting, and analyzing structured data related to business transactions, Web-based information systems go beyond this functionality by integrating different media for knowledge representation and by hypertext functionality, thereby supporting not only the

creation, integration, analysis, and distribution of structured information related to business transactions, but also the storage and transfer of knowledge. While Web-based transaction systems do not conceptually differ from respective traditional systems, Web-based information systems allow for more advanced forms of information management than traditional information systems can (Kaiser, 2000). As a consequence, not only organizational, functional, and data views of business processes have to be specified during conceptual design of Web-based information systems. Moreover, complex knowledge components and their associations must be specified in order to enable flexible access and efficient navigation.

As an example, the data view of traditional information systems development and Web-based information system development is compared: A traditional data model comprises

- types of structured information objects (e.g. order, customer, product),
- certain types of relationships between information object types (e.g. generalization dependencies, references = existential dependencies),
- consistency constraints valid for all instances of a certain information object type, and
- eventually attributes of information object types.

All modeling is done on a type level.

Of course, all of these element types can be found in the data model of a Web-based information system. But a Web-based information system representing complex knowledge may also comprise the following additional element types:

- One-of-a-kind information objects (i.e. single object instances not belonging to any object type, e.g. a mission statement)
- Unstructured information objects (e.g. documents), and
- Additional types of associations between information objects (e.g. "part-of" relationships used for structuring a knowledge domain and "association " relationships used for navigation within a knowledge domain).

Related Work and Overview

While early work on the development of Web-based information systems concentrates on navigational, text-related issues (e.g. Diaz, Iskowitz, Maiorana & Gilabert, 1997), recent work also covers interface design and overall site design (e.g. Lyardet, Mercerat & Miaton, 1999; Lynch & Horton, 1999). In the following, navigational and text-related work as well as work on hypermedia design is discussed first. After that, more holistic approaches to the conceptual design of Web-based information systems are discussed.

Conceptual modeling of hypertext / hypermedia systems

Based on hypertext design (e.g. Isakowitz, Stohr & Balasubramanian, 1995), the design of Web-based information systems has been addressed since the mid 90ies. E.g., the HYDESIGN model (Marmann & Schlageter, 1992) proposes classes of (hypermedia) objects comprising 'atomic nodes', 'SBL nodes', and links between those nodes. Atomic nodes inherit general class properties and represent 'content'. SBL (**S**tructure **B**ehavior **L**ocality) nodes represent additional information: Structure defines how nodes and links can be connected, behavior defines the

dynamics of nodes and links, and locality defines a 'local' environment, i.e. a sub-module of the hypermedia network. By this approach, important conceptual constructs like navigation paths or aggregate objects (=sub-networks) are introduced into hypermedia design. When hypermedia design approaches (e.g. De Bra & Houben, 1992; Diaz et al., 1997; Isakowitz et al., 1995; Marmann & Schlageter, 1992) are used for conceptual modeling of Web-based information systems, however, important aspects like common standards of Web navigation, interface design, conceptually different classes of links, and overall site design are not taken into account. Since many problems addressed by hypermedia design like 'disorientation' or 'cognitive overhead' are also prevailing in large Web-based information systems, however, the basic approach of a hierarchical, recursive model of node classes and link classes is useful for Web-based information system design. But in addition to static links between particular documents, common standards of Web navigation imply dynamic links (Chen, 1997).

HDM (Hypertext Design Model), a conceptual model for hypermedia applications, separates information modeling, navigational modeling, and presentation modeling. Information objects are connected by semantic links, while the internal structures of information objects are represented by structural links between components (Garzotto, Paolini & Schwabe, 1993).

From both approaches, important features of information modeling and site design can be adapted, while navigational issues, unstructured information objects, and 'one-of-a-kind' information objects need to be complemented.

Holistic approaches to Web site and intranet design

ARANEUS (Atzeni, Mecca & Merialdo, 2000) separates database design and hypertext design, but neglects navigational issues in favor of information modeling. Although information objects are represented as networks of components, the central role of database design in ARANEUS imposes numerous restrictions for unstructured information objects and navigational design.

W3DT (initially called SHDT) was proposed as an early approach (and tool) for conceptual Web-based information system design (Bichler & Nusser, 1996; W3DT-Team, 1997). In W3DT, the development process comprises seven steps, guiding the designer from requirements analysis to implementation of the Web-based information system. For the first phase (requirements analysis), a collection of methods to determine user requirements is provided. The design phase is divided into information structuring, navigational design, organizational design, and interface design. Information structuring and navigational design is understood as an iterative mental process where a large knowledge domain has to be organized and structured into information objects, which are then linked and evaluated. Organizational design assigns information on maintenance and organizational integration to the information objects. When the design phase is completed, the information objects are implemented by HTML pages and gateway scripts, and the access structures of the model are implemented by links within the WWW site.

The W3DT research group has shown that it is possible to capture the navigational structure of real-world Web-based information systems. In W3DT's meta model, a Web-based information system is composed of pages, links, and layouts. While layout is not further specified, two different types of links and four different types of pages are differentiated. This simple classification, however, is not sufficient to capture the diversity of current HTML pages. Moreover, it is not differentiated between page templates and page contents.

A more holistic and advanced approach has been developed by Kaiser (Kaiser, 2000). Although intended as a methodology for the conceptual design of intranets, his results are also applicable to the conceptual design of Web-based information systems. Being the most important part of the methodology, the specification of a Web-based information system comprises not only the modeling of the information structure (represented by an entity-relationship schema), but also the modeling of navigational aspects (represented by a W3DT schema).

Summing up, most features of conceptual design of Web-based information objects have already been proposed in hypertext design, intranet design, or database design. A holistic approach that covers all systems aspects and information object classes, however, is lacking.

Overview

Following this introduction, we try to identify the 'essence' (McMenamin & Palmer, 1984), i.e. the implementation independent specification, of a complex Web-based information system. In section 3, an appropriate conceptual model is proposed that captures not only hierarchical document structure and hypertext semantics, but also allows for an implementation by dynamical page generation from databases and using various approaches to explicit and implicit navigation. For illustration purposes, the model is applied to an existing Web-based information system in section 4. In section 5, current Web-based information system development tools are analyzed with regard to the extent to which conceptual Web-based information system modeling is supported. The paper closes with some conclusions in section 6.

THE 'ESSENCE' OF A WEB-BASED INFORMATION SYSTEM

The benefits of conceptual modeling result from creating stability: Often contents change, but the general page structure remains unchanged. At other occasions, templates change, but contents remain stable. For conceptual modeling of Web-based information systems, therefore, content should be separated from navigational structure, navigational structure should be separated from layout (i.e., general page structure), and layout should be separated from content.

But if all implementation-related specifications are to be neglected, what is then the essence of a Web-based information system? As for traditional applica-

tions, organizational, functional, and data aspects of business transactions have to be specified. In addition, representation and access aspects of complex information areas have to be specified by means of

- (structured and non-structured) information objects,
- 'part-of' (i.e. hierarchical) relationships between information objects, and
- 'associative' (i.e. non-hierarchical) relationships between information objects.

Information objects may be one-of-a-kind instances that do not match any type definition, or may belong to some class of information objects. They may be administrated internally (i.e. within the same organizational context as the Web-based information system) or externally. They may be implemented by operating system files, file records, tuples of relational tables, or mail accounts.

Hierarchical relationships may be implemented by file system structure, HTML navigation, or by applets. Non-hierarchical relationships may be implemented by hyperlinks, mailtos, or downloads.

Information objects may be complex, i.e. may comprise several components. As a consequence, we have to represent

- components of information objects (e.g. character strings, pictures, textual tables) and
- hierarchical relationships between information objects and their components.

Information object components are usually implemented by operating system files or tuples of relational tables. Hierarchical component links may be implemented by physical inclusion, as reference using frame or border techniques, or using style sheets.

Why should information objects and their components be differentiated?

The easiest conceptual interpretation of a Web-based information system is a multi-stage 'assembly' of elementary information objects. We propose, however, to differentiate information objects from their components, thereby introducing an additional representation level.

Information object contents change often. Since conceptual models should provide some basic amount of stability, it is necessary to differentiate a stable and a volatile part in the 'assembly' structure of a Web-based information system. Information objects denote the borderline between the volatile and the stable portion of the network: While their contents change often (which has to be reflected in updating components and/or updating hierarchical component structure), their conceptual representation is stable so that the 'upstream' portion of the Web-based information system remains unchanged.

E.g., status, employee assignments, schedule, available documents, and other attribute-like properties of a project vary quite often, while its organizational embedding and its links to other projects remain quite stable. As a consequence, project documents should be regarded as information object components, while the project itself should be regarded as an information object.

In systems implementation, the differentiation of information objects from their components is reflected making the former accessible by hyperlinks, while the latter can not be separately accessed.

Information object components are the building block of every Web-based information system. Information objects can be identified as simplest information component aggregates that guarantee some degree of stability. Navigation areas are those portions of a Web-based information system which are so tightly coupled that some degree of navigation traffic is to be expected. Web-based information systems are those portions of the World Wide Web whose contents can be updated in some organizational context. The resulting conceptual structure of the World Wide Web can be interpreted as a five level hierarchy illustrated by figure 2. The Web-based information system level, the navigation area level, and the component level may comprise several sub-levels to reflect organizational responsibilities (Web-based information system level), relationships between information objects (navigation area level), and information object structure (component level).

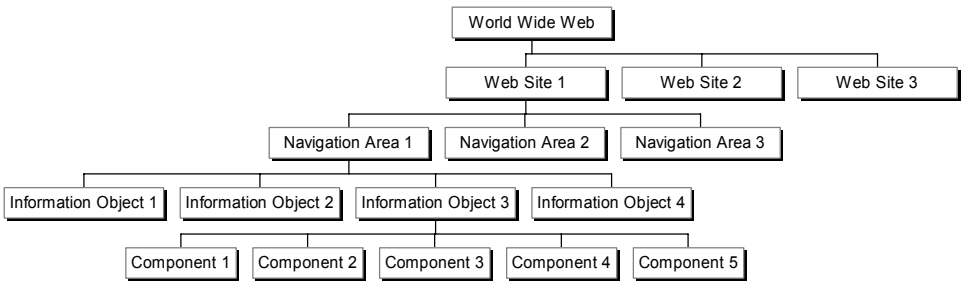
How can relevant information objects, components, and their relationships identified?

This paper focuses on methods to represent the essence of a Web-based information system. Of course, in order to represent information objects, components, and relationships, it is necessary to identify respective real-world objects first. The Object Type Method, a semi-formal approach to identify conceptual structures that is well grounded in linguistic theory, has been summarized in (Ortner & Söllner, 1989). Following this approach, identifying conceptual structures is achieved by reconstructing terms that professionals use by means of a set of basic construction operators (e.g. inclusion, association, aggregation). For details regarding the Object Type Method, we refer to (Ortner & Söllner, 1989) and the literature referenced in their paper.

CONCEPTUAL MODEL

Based on the discussion of related work in section 1 and the identification of conceptual components of a Web-based information system in section 2, we

Figure 2: Conceptual WWW hierarchy (Source: Strauch & Winter, 2001)



propose a conceptual model that is intended to cover all relevant components and their relationships, to imply some important quality rules, and to be a suitable foundation for an appropriate activity model which is presented in section 4.

META MODEL

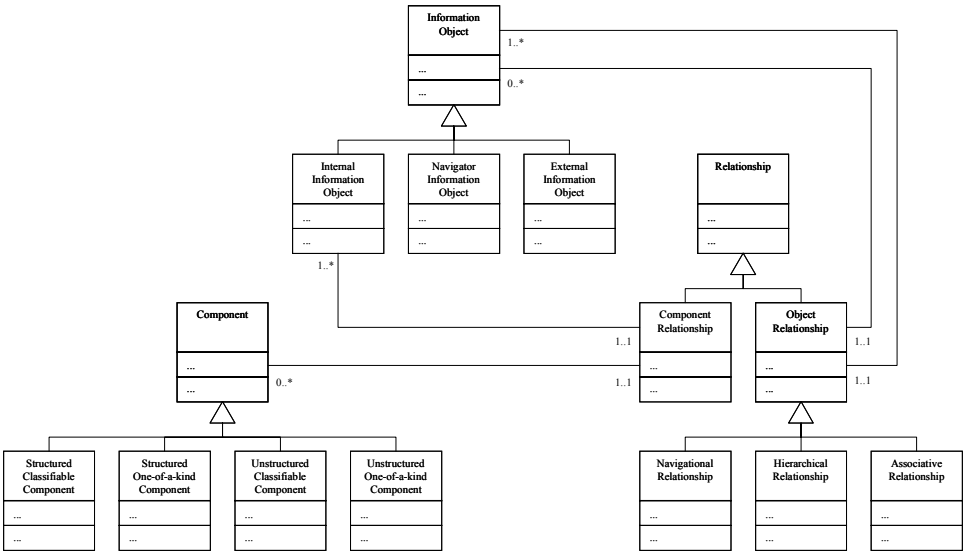
The meta model illustrated in figure 3 (UML class diagram (UML 2000)) represents all components of the conceptual model and their relationships. Basically, our meta model comprises information objects (with several subtypes), components of information objects (with several subtypes), and relationships between information objects (with several subtypes) or between information objects and their components. The components of the conceptual model and an appropriate graphical notation are presented in the following subsections.

Components of Information Objects

Structured classifiable components

This class of components represents information that can be regarded as an instance of some generalized class of information with certain attributes. E.g., data about a person's publications are structured because every publication has certain attributes (title, year of publication, etc.) and are classifiable because each data is an instance of a common class 'publication'. For structured classifiable components, structure and contents can be separated.

Figure 3: Meta Model



Unstructured classifiable components

This class of components represents information that can be regarded as an instance of some generalized class of information, but has no common attributes within that class. E.g., job descriptions may not follow some formal structure, but are nevertheless classifiable because each job description is an instance of a common class 'job description'. For unstructured classifiable components, structure cannot be separated from contents.

Structured, one-of-a-kind components

This class of components represents information that does not belong to some generalized class of information, but comprises certain attributes. E.g., a form is a structured, one-of-a-kind component because every form has certain attributes (layout, boilerplate texts, field names, etc.), but cannot be assigned to some common meaningful class. For structured one-of-a-kind components, structure and contents can be separated.

Unstructured, one-of-a-kind components

This class of components represents information that neither belongs to some generalized class of information nor has certain attributes. E.g., a mission statement is an unstructured, one-of-a-kind component because only one such text exists within a Web-based information system, and no general statement structure is available. For unstructured one-of-a-kind components, structure cannot be separated from contents.

Information Objects

(Internal) Informational objects

Information objects are the simplest information component aggregates that guarantee some degree of stability. E.g., employee home pages are information objects because they are composed from personnel data (name, date of birth, etc.), images, links to organizational units, etc. that frequently change, but do not change as an aggregate as long as the employee is part of the organization.

Navigators

Navigators are application components that allow users to efficiently navigate within some portion of the Web-based information system. In contrast to ordinary information objects, navigators represent informational structure instead of the information itself. However, navigators are composed of several components so that they represent a subclass of information objects.

External information objects

External information objects are not part of a Web-based information system, but can be accessed from that Web-based information system, e.g. by hyperlinks. In

contrast to (internal) information objects, the structure of external information objects is unknown. At most, some basic structural information can be generated from the URL extension (HTML, IDC; HTX, ASP, NSF, CGI, etc.).

Relationships

Hierarchical relationships

Hierarchical relationships represent 'part-of' links between internal information objects. Navigators usually are based on hierarchical relationships. E.g., department information and open position information can be linked by a hierarchical relationship.

Associative relationships

Associative relationships represent associations between information objects that are not related hierarchically or by navigators. E.g., open position information and project information can be linked by an associative relationship.

Navigational relationships

Navigational relationships represent associations between (internal and external) information objects and navigators. In contrast to hierarchical and associative relationships that are used to represent information semantics, navigational relationships represent the application interface and are used to support the browsing process (Morville & Rosenfeld, 1999). E.g., department information can be linked to subordinate unit information as well as superior unit information by navigational relationships.

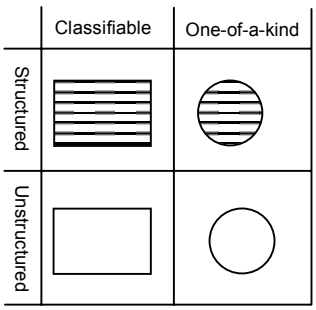
Component relationships

Component relationships represent the structure of internal information objects. In contrast to hierarchical and associative relationships, component relationships do not imply any hypertext functionality.

Notation

We denote the element of the conceptual model as follows:

Components of information objects



Information objects

(Internal) information object

Navigator

External information object

Relationships

Hierarchical relationship

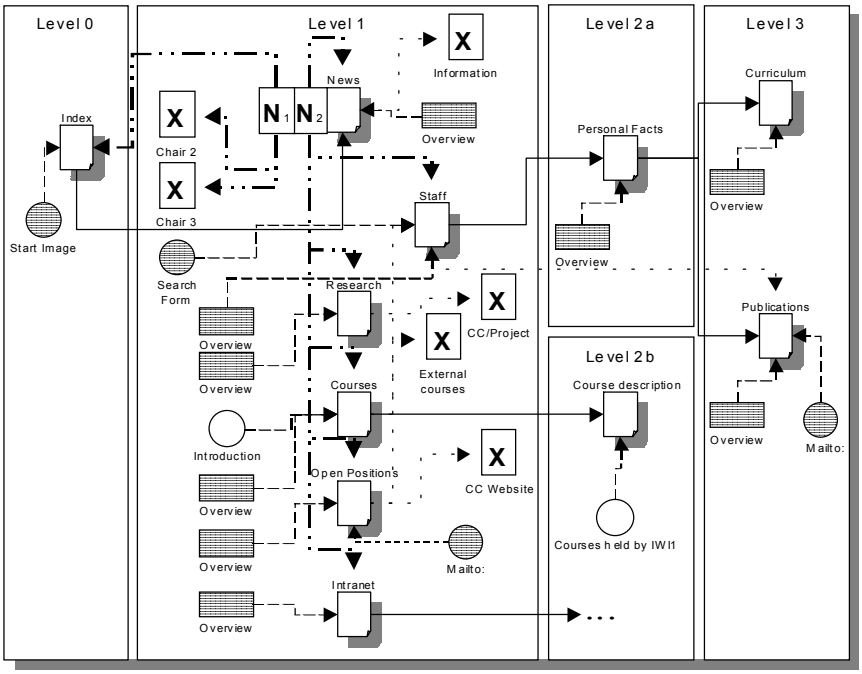
Associative relationship

Navigational relationship

Component relationship

Figure 4 illustrates the application of the proposed model to an existing Web-based information system of a university workgroup. The levels are related to hierarchical relationships within the information objects of the Web-based information system.

Figure 4: Conceptual schema of a Web-based information system (Source: Strauch & Winter, 2001)



Quality

The proposed conceptual model implies the following quality controls:
Within the Web-based information system:

- All hierarchical relationships must link an internal information object to one or more internal information objects
- All associative relationships must link an internal information object to one or more internal or external information objects
- All navigational relationships must link a navigator to one or more internal or external information objects
- All component relationships must link an internal information object to one or more components

Within the World Wide Web

- All URLs referenced by associative or navigational relationships must exist

ACTIVITIES FOR THE CONCEPTUAL DESIGN

For conceptual modeling of a Web-based information system, model elements, graphic notation, and quality controls have to be complemented by some basic methodology, i.e. at least by basic activities for the design and a recommendation of a certain sequence of design phases. Based on our experience with Web-based information system design and the discussion of essential Web-based information system elements, we propose the activity sequence described in this section.

Identification of Information Objects

Whether an information representation is an information object or should be regarded as a component of an information object should be decided by analyzing its update behavior: If frequent updates can be expected not only for its contents, but also for its relationships, it should be represented as a component. If updates will mostly be restricted to contents, it may be represented as an information object.

(Re-)Construction of Hierarchical Structure

In contrast to associative relationships, hierarchical relationships create a consistent directed graph of information objects. Based on the hierarchy, levels of the conceptual Web-based information system schema can be identified (see figure 6). All hierarchical relationships link an information objects on level n to one or more information objects on level n+1. Since hierarchical relationships help to structure the knowledge domain to be modeled (i.e. re-constructed), conceptual modeling should start with building hierarchical relationships.

Figure 5: Identification of information objects (Source: Strauch & Winter, 2001)

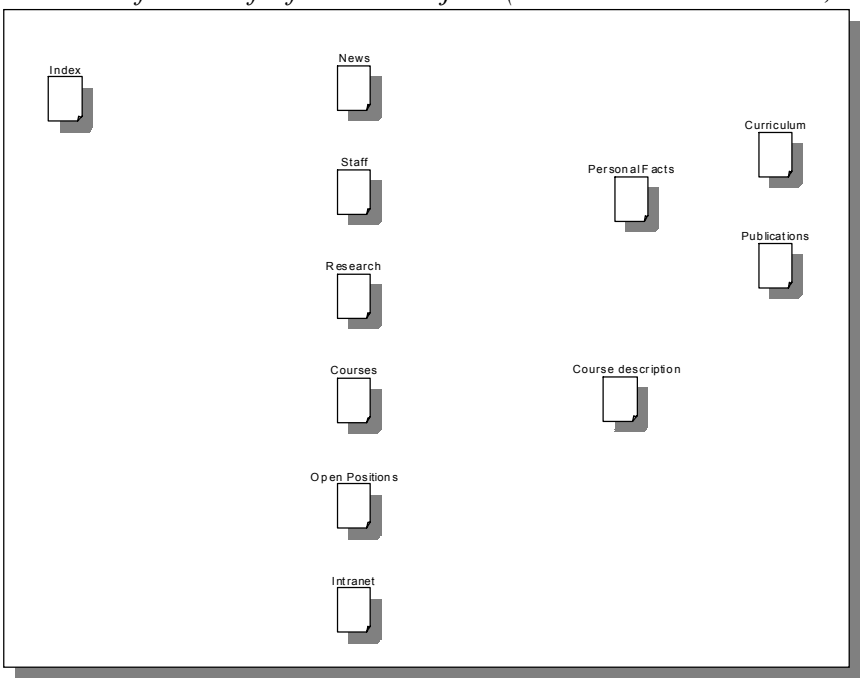


Figure 6: (Re-)Construction of hierarchical structure (Source: Strauch & Winter, 2001)

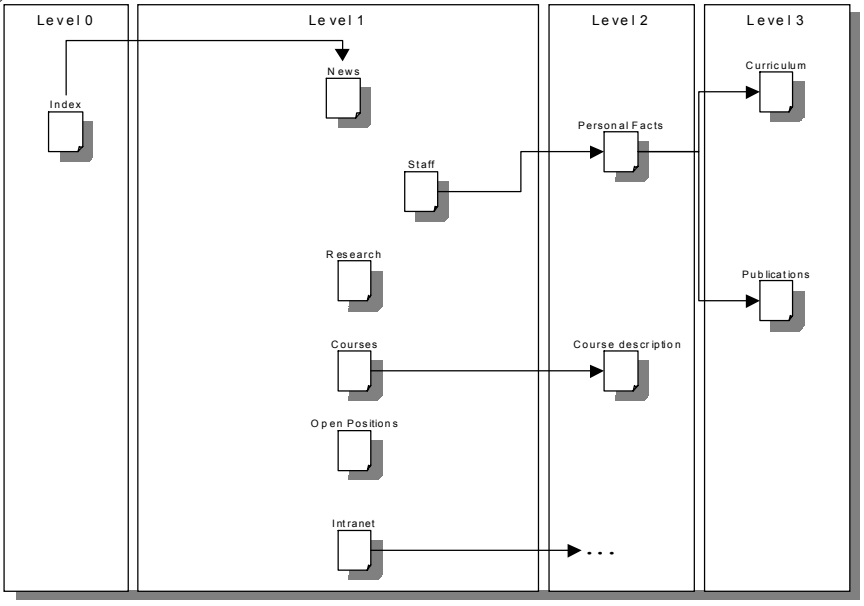
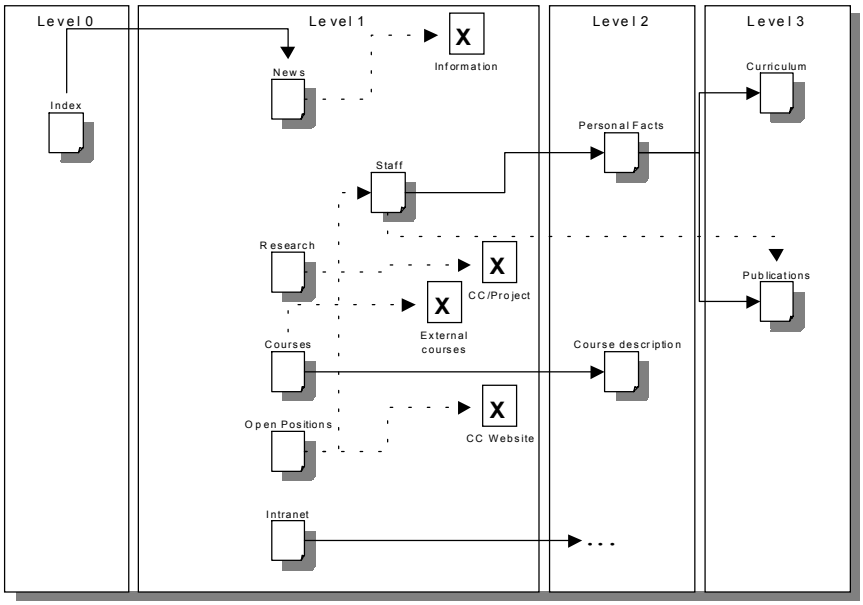


Figure 7: (Re-)Construction of associative structure (Source: Strauch & Winter, 2001)



(Re-)Construction of Associative Structure

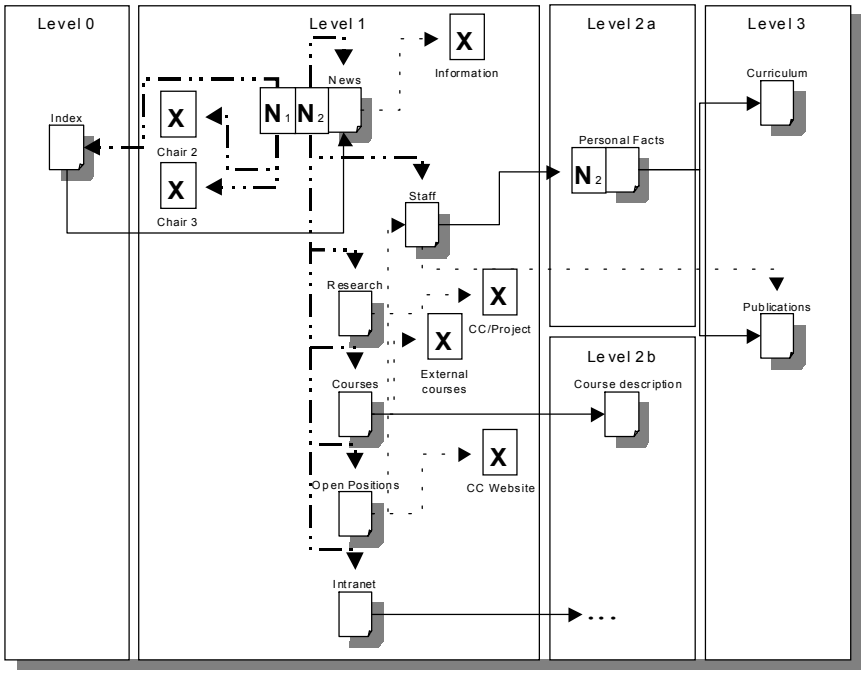
All semantic links that do not represent "part-of" relationships are represented by associative relationships in the next step. Associative relationships are bound to certain levels of the schema. External information objects are always linked to internal information objects by associative relationships. They should be modeled on the same schema level as the internal information objects they are linked to.

Design of Navigators

Although usually being based on hierarchical or associative relationships, navigators and navigational relationships represented a separate semantic concept. While the navigator itself represents application interface properties, navigational relationships represent its behavior. Using navigators, application users can easily navigate in a knowledge domain without having to follow hierarchical or associative relationships.

One or more navigators can be directly linked to an information object. If more than one navigator is linked to an information objects, an index number is introduced. A navigator is linked to that information object whose browsing shall initially create the navigation menu. A navigator it 'inherited' by a set of other information objects on the respective schema level and on the next schema level. By separating schema levels according to information object sets related to navigators, elements of complex Web-based information system schemas can be arranged.

Figure 8: Design of navigators (Source: Strauch & Winter, 2001)



Representation of Components

Component relationships and information object components can be represented either in the schema (see figure 4) or, for large schemas or complex information objects, in a sub-schema for the respective information object. Since relationships (including navigational relationships) are defined between information objects only, modifications of component structure do not affect the overall schema. Components could be reused by more than one information object.

TOOL SUPPORT

In this section, the state-of-the-art of tool support for the development of Web-based information systems is analyzed. There are two classes of tools available: On the one hand, WWW authoring tools could be used to design not only small sets of Web pages, but also complex Web-based information systems. On the other hand, traditional CASE toolsets could be used to design Web-based information systems as a special implementation of traditional, client/server information systems.

WWW Authoring Tools

In our opinion, the most advanced tools in this regard are Microsoft's Frontpage 2000, Microsoft's Visual Interdev, and Macromedia's Dreamweaver. Most other tools are just HTML coding tools and provide no support for conceptual modeling at all.

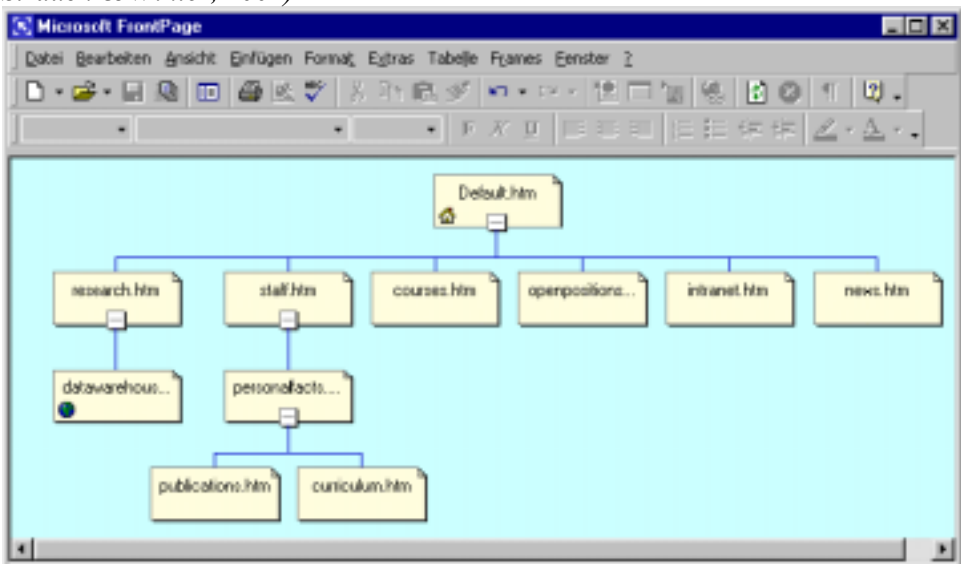
Microsoft FrontPage 2000

Frontpage 2000 provides two types of information objects which are internal information objects and external information objects. Internal information objects are directly corresponding to HTML files stored on the Web server. Component structure cannot be modeled. Navigators, however, are automatically generated. But these structures depend on Frontpage's navigational model and properties that have to be coded (no conceptual representation). For each Web-based information system, the designer has to decide globally whether borders should appear or not.

Hierarchical relationships between information objects can be modeled as 'navigational model' in Frontpage 2000 (see figure 9). Associative relationships, however, as well as navigational relationships in our definition and component relationships are unknown. When updating hierarchical relationships, deletions of entire sub-trees (including the respective information objects) may occur when a single relationship is deleted.

Summing up, just few basic concepts of conceptual Web-based information system design are currently supported by Frontpage 2000. It is necessary to extensively re-code Web-based information systems generated by Frontpage 2000 to allow for features like associative links, component reuse, non-standard navigators, etc. Obviously, respective Web-based information systems are not really based on a stable conceptual schema so that technical innovation will require significant development efforts.

Figure 9: Web-based information system schema in Frontpage 2000 (Source: Strauch & Winter, 2001)



Microsoft Visual Interdev 6.0

Visual Interdev is intended for professional developers. It provides the same kind of navigational model as Frontpage 2000 and has the same limitations. Therefore, the same conclusions can be drawn.

Macromedia Dreamweaver 3.0

In Dreamweaver's conceptual model, two classes of information objects exist. On the one hand, there are information objects which represent a physical HTML file on the Web server. On the other hand, there are information objects like external links, mailto tags, and images. To some extent, these two classes correspond to information objects (files) and components (links, images) in our model (see figure 10). However, the latter group of objects is not differentiated so that meaningful quality checks or activities for the design cannot be applied.

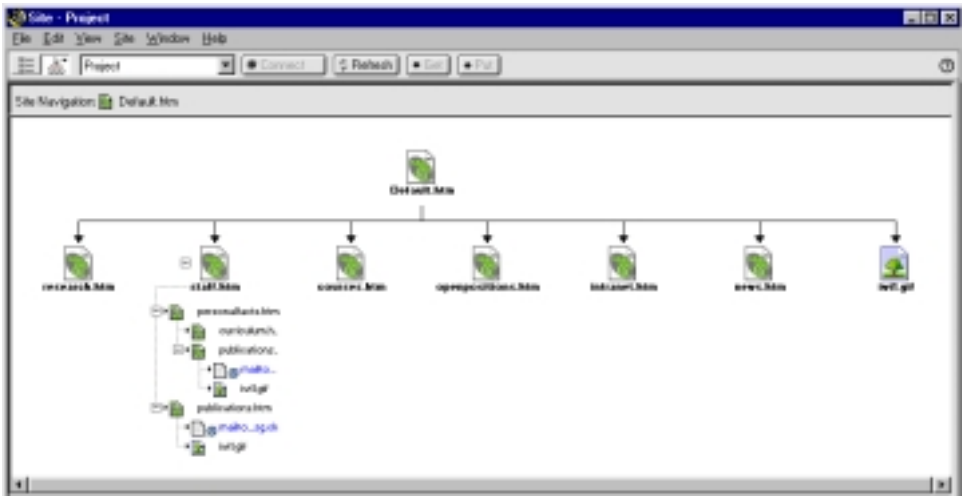
Between information objects, hierarchical relationships as well as associative relationships can be modeled. However, Dreamweaver does not provide a different notation for these two classes of relationships. Although associative links can be modeled, Web-based information systems can only be displayed as expandable trees (based on hierarchical relationships) and not as networks that also include associative relationships.

Dreamweaver does not support the design of navigators (and navigational relationships). The generator directly creates HTML hyperlinks from associative relationships.

Other Web design tools

Adobe PageMill 2.0 for Windows is basically a HTML editor with WYSIWIG capabilities, i.e. there are no real modeling features. However, the product overview

Figure 10: Web-based information system schema in Dreamweaver 3.0 (Source: Strauch & Winter, 2001)



of SiteMill for Macintosh (ADOBE SYSTEMS Inc., 1997) claims that the tool has some modeling features like a 'site view' and an 'external reference view'.

Aimtech's Jamba is a generator for Web pages that include multimedia objects implemented by Java code. Applets can be generated without any knowledge of the Java syntax by using a graphical interface for the assembly of multimedia objects to pages. Generated applications are stored in three different files:

- The project file (extension .jmb) comprises all information provided by the developer
- The HTML file (extension .html) comprises all applets and can be processed by Internet browsers
- The text file (extension .jtf) comprises information on interactive multimedia objects and is accessed by the HTML file

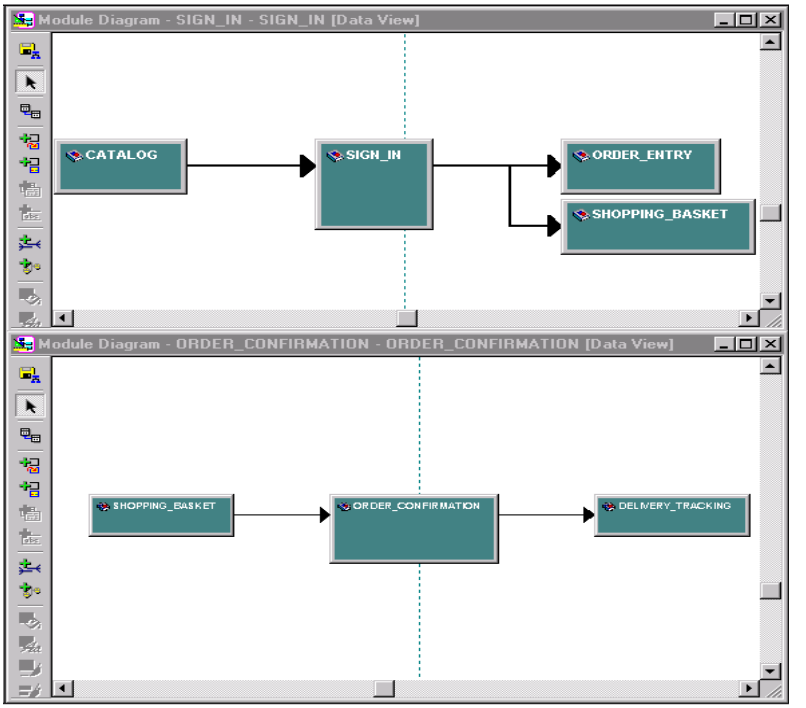
During page development, an object viewer can be used to simulate Web pages without having to use a browser. As building blocks for multimedia pages, a large number of buttons, graphics, menus, text fields, and links are provided. Jamba can be regarded as a comfortable Web page editor with Java generation capabilities and a large number of reusable components. It is, however, restricted to the development of single pages (not complete sites) without extended procedural features and without support for requirements specification, analysis, or design.

CASE Toolsets

As an example for advanced, state-of-the-art CASE toolsets, Oracle's Designer is regarded. To implement a conceptual design, Designer offers a Web site generator in addition to client/server generators for Oracle's 4GLs, for Visual Basic, for C++, and others. Based on a long tradition of database design, structured analysis, and more recently integrated process modeling, CASE toolsets like Designer offer an adaptable application development repository, advanced graphical design tools, simulation capabilities, workgroup support, a scalable environment, powerful generators, open interfaces to other development tools, lots of reports and quality checking tools, lots of utilities, etc. Various systems views can be (more or less) concurrently developed by deploying different design tools, and the information systems life cycle can be completely covered by using requirements specification tools, systems design tools, implementation tools, and maintenance / reverse engineering tools. In figure 12, the data view (in contrast to the display view) of a module subtree in systems design is illustrated. Data links and component structure are represented in that schema, while navigational links (that are widely based on data links) are represented in the display view of that module subtree.

In most cases, advanced CASE toolsets are built around a relational database. As a consequence, the concept is more or less restricted to structured information objects: While form-and-menu based information systems of any complexity can be developed very efficiently, non-trivial navigational structures, documents / images, and one-of-a-kind information objects cannot be covered effectively.

Figure 11: Module data view of information system schema in Designer



CONCLUSIONS

Based on a short discussion of related work and an analysis of the essence of a Web-based information system, a conceptual model has been proposed that combines advantageous conceptual modeling features of several approaches to hypermedia design, intranet design, and Web site design. Application to a real-life example shows that useful quality checks can be specified, complex navigational constructs and information object structures can be designed, and schema elements can be usefully arranged based on the proposed model. An analysis of both WWW authoring tools and CASE toolsets, however, shows that neither tool type is currently able to support such a rich conceptual models. While WWW authoring tools are generally better in designing the special features of Web sites (e.g. navigation), CASE toolsets are better in supporting large schemas, providing workgroup support, and supporting several views and phases of systems development.

In order to prevent current Web-based information system development to create the legacy systems of the future, it is necessary to base implemented systems on a stable conceptual foundation. To provide a dependable specification for Web design software vendors, approaches to conceptual modeling from scientists and practitioners have to be integrated, thereby taking into account findings from hypertext design as well as experience from CASE practice. Hopefully, specific discussion of Web-based information system modeling issues will help such specifications to emerge.

REFERENCES

ADOBE SYSTEMS Inc. (1997). Adobe PageMill Details - SiteMill for Macintosh. Available on the World Wide Web at: http://www.adobe.com/ prodindex/ pagemill/siteben.html. Accessed November 27, 1997.

Atzeni, P., Mecca, G. and Merialdo, P. (2000). Design and Maintenance of Data-Intensive Web Sites. Available on the World Wide Web at: http:// www.dia.uniroma3.it/Araneus/publications/rt25-97.zip. Accessed August 11, 2000.

Bichler, M. and Nusser, S. (1996). SHDT-The strucured way of developing WWW-sites. In Dias Coelho, J. (Ed.). *Proceedings of the 4th European Conference on Information Systems*, 1093-1101.

Chen, C. (1997). Structuring and visualising the WWW by generalized similarity analysis. In N.A., *Proceedings of the 8th ACM conference on Hypertext HYPERTEXT'97*, 177-186.

De Bra, P. and Houben, G. J. (1992). An extensible data model for hyperdocuments. In N.A., *Proceedings of the ACM Conference on Hypertext ECHT'92*, 222-231.

Diaz, A., Iskowitz, T., Maiorana, V. and Gilabert, G. (1997). RMC: A Tool To Design WWW Applications. Available on the World Wide Web at: http:// www.stern.nyu.edu/~tisakowi/ papers/www-95/rmcase/187.html. Accessed November 27, 1997.

Garzotto, F., Paolini, P. and Schwabe, D. (1993). HDM-A model-based approach to hypertext application design. *Transactions on Information Systems*, 11, 1-26.

Isakowitz, T., Stohr, E. A. and Balasubramanian, P. (1995). RMM: A methodology for structured hypertext design. *Communications of the ACM*, 38, 34-44.

Kaiser, T. (2000). Methode zur konzeption von Intranets. [Method for the Conceptual Design of Intranets]. *Doctoral Dissertation*, University of St.Gallen, Switzerland.

Kalakota, R. and Robinson, M. (1999). E-Business: Roadmap for Success. Boston: Addison-Wesley.

Lyardet, F. D., Mercerat, B. and Miaton, L. (1999). Putting Hypermedia Patterns to Work: A Case Study. Available on the World Wide Web at: http:// pelican.info.unlp.edu.ar/ht992/patternsAtWork.html. Accessed May 10, 1999.

Lynch, P. J. and Horton, S. (1999). Web Style Guide: Basic Design Principles for Creating Web Sites. Available on the World Wide Web at: http:// info.med.yale.edu/caim/manual/contents.html. Accessed May 10, 1999.

Marmann, M. and Schlageter, G. (1992). Towards a better support for hypermedia structuring: The hydesign model. In N.A., *Proceedings of the ACM Conference on Hypertext ECHT'92*, 232-241.

Martin, J. (1992). Application Development Without Programmers. Englewood Cliffs: Prentice-Hall.

McMenamin, S. M. and Palmer, J. F. (1984). Essential Systems Analysis. New York: Yourdon Inc.

Morville, P. and Rosenfeld, L. (1999). Designing navigation systems. Available on the World Wide Web at: http://Webreview.com/wr/pub/98/02/20/arch/index.html. Accessed April 28, 1999.

Ortner, E. and Söllner, B. (1989). Semantische datenmodellierung nach der objekttypenmethode [Conceptual Data Modeling Using the Object Type Method]. *Informatik-Spektrum*, 12, 31-48.

Richmond, A. (1999). Navigation Architecture of The WDVL. Available on the World Wide Web at: http://www.stars.com/WebRef/Navigation/WDVL.html. Accessed April 28, 1999.

Rosenfeld, L. and Morville, P. (1998). Information Architecture for the World Wide Web–Designing Large-scale Web Sites. Sebastopol: O'Reilly & Associates.

Strauch, B. and Winter, R. (2001). Conceptual Web site modeling. In Rossi, M. and Siau, K. (Eds.). *Information Modelling in the New Millennium*. Hershey: Idea Group Publishing.

UML. (2000). OMG Unified Modeling Language Specification. Available on the World Wide Web at: http://www.omg.org/technology/documents/formal/unified_modeling_language.htm. Accessed January 22, 2001.

W3DT-Team. (1997). W3DT-WWW Design Technique. Available on the World Wide Web at: http://wwwi.wu-wien.ac.at/w3dt/. Accessed November 26, 1997.