# Modeling Cooperative Design Analyses

B. Tunçer and R. Stouffs
Delft University of Technology
Department of Architecture
Delft
The Netherlands

## ABSTRACT

The study of precedents plays an important role in design and design education. Architecture students prepare analyses of prominent precedents with respect to various criteria. Such design analyses are represented and communicated through abstractions. Collections of these abstractions are stored, related, managed, and presented in digital environments. Such web-based environments can serve as an extensible library of design precedent analyses. The use of an extensive library by a collection of students requires a flexible and extensible information model for relating and integrating the various contributions. We propose a methodology that establishes an information model for digital architectural analysis environments. This model facilitates a rich information structure of abstraction entities and their relationships, both structural and semantic, offering increased value for accessing and browsing this information. Specifically, a rich information structure allows one to access the information from alternative views to those that are expressed by the individual abstractions. In this paper, we start by discussing precedent-based learning, and describe the abstraction model currently used for precedent documentation and analysis. We then present our methodology for achieving a rich information structure. We end the paper with a description of an implementation of this methodology as an architectural analysis construction and presentation environment for a second year design studio.

## 1 PRECEDENT-BASED LEARNING

The study of precedents plays an important role in design and design education (Akin et al. 1997). While practitioners can rely on their own and their colleagues' experience in the process of a new design, students can only draw upon documented examples of success and failure from known architects. Especially in the early stages of design, it is common practice for architecture students to collect information on prominent buildings relevant to their design task. 'The rationale for banking on so called precedents is straightforward: it is wrong to wish to reinvent the wheel over and over again. … We should learn from our elders and adopt their successful solutions to problems similar to the ones we cope with' (Goldschmidt 1995, p. 70). Precedents as finished and complete design objects contain knowledge of design. A study of such precedents can yield, among others, heuristics used by the designer, design principles for various purposes and situations, and prototypes from building typologies.

Architecture students often prepare case studies for their design studio projects, gathering information about existing buildings with similar functionality to the subject of their project. They present this in the form of collages on paper, or as hyper-documents. By integrating the respective results into a common library, students can draw upon others' results for comparisons and relationships between different aspects or buildings.

There have been attempts at collecting and organizing these results into computational environments (e.g., Madrazo and Weder 2001, Madrazo 2000) as a collection of categorized and hyperlinked documents. The EDAT example (Akin et al. 1997) additionally offers the students a tool to present their work in the design studio and is extendable in different ways, e.g., for carrying out performance analyses on the stored test cases.

Such environments for precedent-based learning generally use an abstraction model or a document-based model. They use collections of abstractions for representing, storing and presenting design information. A drawing or text specifies a single abstraction; each abstraction expresses a different aspect of the design object, such as form, function, acoustics, structure, process, space, and organizational relationships (Schmitt 1993, p. 39). Abstractions are expressed as documents of various formats, e.g., drawings, diagrams, 3D models, images, simulations, and texts. Such computational environments treat the individual abstractions as entities or objects that are organized and related according to various categories and attributes. The purpose is to offer a flexible organizational framework and enable easy indexing and retrieval of documents.

There may be many different reasons for retrieving information on precedents in an electronic environment. Given a group of precedents of a specific building type, e.g., theaters, one may be interested in a particular theater hall because the works of that particular architect are of interest. Or, one may want to look at all foyers in order to get an overview of different circulation schemes used in theaters. Alternatively, one may want to deduce rules of thumb about designing theater halls with good acoustics by looking at theater halls that are considered to be examples of good acoustics. Such cases can be enumerated for pages. In general, information retrieval in such environments are based on keyword searches. Documents are indexed such that each document is represented by a set of keywords. This indexing can be done manually or automatically. Information retrieval actions within precedent-based learning environments generally fit one of the following two categories. Firstly, one may want to retrieve a specific known document that resides in the repository. If the retrieval query contains one or more of the document's keywords, the retrieval will be straightforward. Secondly, one may want to retrieve all documents pertaining to a certain concept or topic, including their links to other related documents. Such an overview of relevant documents may provide the necessary information in order to establish or verify a certain design aspect. The possibility of interpreting the entire document structure seeking information related to a concept of interest is an important requirement in such an environment.

As a result, the use of an extensive library by a collection of students requires a flexible and extensible model for relating and integrating the various contributions. Specifically, there is a need for an information organization that enables a user to access information independently of the individual viewpoints of the authors of the information space. The approach described in this paper provides a methodology for modeling information in such a way as to provide a rigorous recipe when creating cooperative information environments for creating, managing, and presenting architectural analyses.

## 2 A RECIPE FOR DIGITAL ARCHITECTURAL ANALYSIS ENVIRONMENTS

### 2.1 **Rich Information Structures**

Information structures are created, at a minimum, by a collection of information entities, an organization of these entities, and a specification of the relationships between these entities. In the context of an architectural analysis, the individual abstractions and their relationships define the information structure. A dense information structure offers better support for searching and browsing this structure. Searches in a larger structure will offer more results while a denser structure can serve to distinguish entities by their relationships. Browsing a structure is also facilitated by its density as additional relationships offer more ways to move through the space. This density is directly defined by the authors of the information space. Therefore, we aim to support the authors with a methodology for increasing the structure's cardinality and its interrelatedness towards a richer structure: augmenting the structure's relatedness with content information, and expanding the structure through the replacement of abstraction entities by detailed component substructures.

In a syntactic manner, an abstraction can be considered as a compositional structure of data entities and relationships. While each abstraction touches upon a different aspect, abstractions relate through commonalities, similarities, and variations in vocabulary and compositional structures. When the abstractions are numerous and diverse, recognizing these relationships creates a tight network in which the individual abstractions no longer stand out. Such a network of abstractions can be said to embody a rich representation.

A rich information structure of abstraction components and their relationships, both structural and semantic, offers new possibilities for accessing, viewing, and interpreting this information. First, it allows one to access specific information directly instead of requiring a traversal of the abstraction component hierarchy. Individual components can be reached and retrieved more quickly when provided with more relationships. Second, components can be considered from a different point of view. The location of a component in the structure is no longer only defined by its place in the abstraction component hierarchy. Instead, components provide direct access to other related components, forming a part of the first component's context. Third, one can access the information structure from alternative views to those that are expressed by the individual abstractions. New compositions of components and relationships offer new interpretations of the structure and generate views not inherent in the structure as created by the original abstractions. Such interpretations can lead to new abstractions.

### 2.2 **Document Decomposition by Content**

A document management system commonly provides for an organization of documents with respect to categories or keywords. However, a categorization with keywords offers little information on the importance of a concept as specified by a document keyword, or on the portion of the document this keyword applies to. Furthermore, users may opt to simply ignore keywords which apply to only part of a

document. In this way, these document properties offer only a quantitative rather than a qualitative valuation of the document. Instead, by allowing the user to select portions of a document for assigning keywords, many more keywords that better fit parts of documents can be specified and associated with the appropriate document portions. This will make the documents inherently related by content.

Decomposing documents by content creates a richer information structure. Replacing documents with component structures automatically increases the number of information entities. Decomposition relationships between document components extends the network of relationships. Furthermore, a document decomposition enables the relating of keywords to document components, allowing for the specification of keywords that may otherwise be ill-suited to relate to the document at large. Document components that share the same keyword can be considered additionally related.

### 2.3 **Separation of Syntax and Semantics**

Document decompositions can be represented in various ways. We choose to consider a structural decomposition of a document as opposed to a semantic one, that is, document components are defined as subsets of the overall document and using the same representation. This approach to decomposing documents provides a uniform structure that is easily adaptable, unlike a semantic decomposition. In this structure, the semantics of the decomposition are separately specified by a categorization of the document components. This semantic keyword structure is derived from an analogy with the semantics of a system of architectural types. A structural document decomposition particularly applies to texts, images, and simple line drawings, as these lack any strong inherent structure. All composed of symbols from a relatively small vocabulary, i.e., characters, pixels, and line segments, in simple one- and two-dimensional patterns, they are represented in a similar structure and can be operated on in a similar way: divided into smaller parts and the parts organized into a hierarchical structure (figure 1).

Separating semantics from syntax allows a semantic organization to augment the document structure without imposing a specific representational structure. This semantic organization can be specified as a compositional structure of descriptive keywords, in various ways (figure 2). Such a semantic structure assists in achieving a rich information structure. When keywords are organized in a structure, relationships between keywords define additional relationships between document components.

The separation of syntax and semantics ensures extensibility and flexibility of the overall representation and avoids the imposition of a fixed frame of reference. The semantics can easily be altered at any time without requiring an adaptation of the syntactic structure. Users can alter either the decomposition or the categorization without affecting the other. Furthermore, the user has full control on the effective positioning of any document within the categorical organization, by selecting either or both the number of keywords assigned and the level of decomposition. This flexibility avoids a rigorous and tedious process when using an application of this methodology.

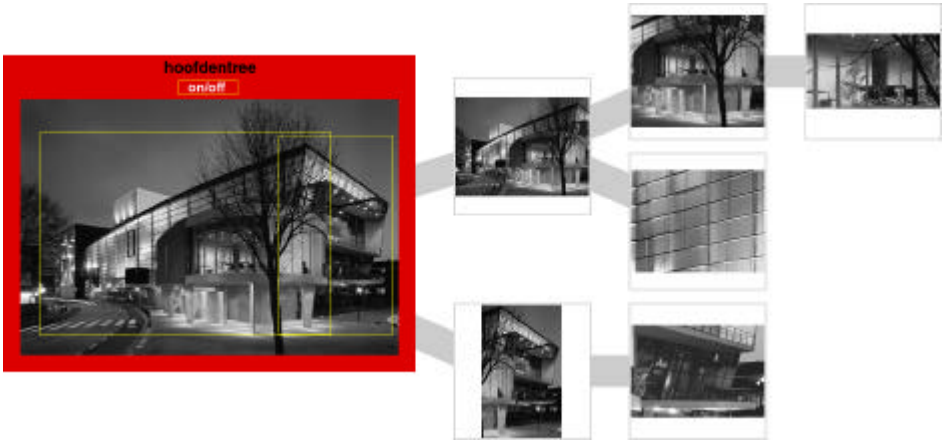


Figure 1: **An exemplar image decomposition hierarchy from the design studio application**

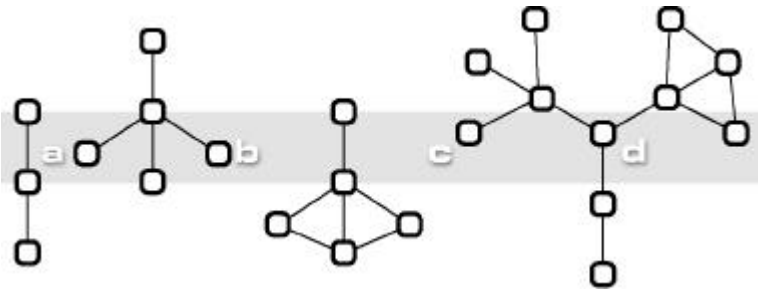


Figure 2: **Schematic diagram of four different semantic structures for descriptive keywords. a) a linear structure, b) a hierarchical structure, c) a network structure, d) a combination of the previous structures**

## 2.4 **Architectural Types as Semantic Guideline**

Within a discipline, members structure shared knowledge through the definition and classification of common concepts. Architects generally classify building designs based on spatial and formal features. This classification introduces the concepts of type and typology. Types in architecture assist, besides the communication of shared knowledge, the analysis of existing buildings, and the design of new buildings (Leupen et al. 1997, p. 132). The concept of building types plays a central role in architecture, although there is no single definition of type and various approaches to the subject exist. Building types, e.g., museums, offices, and libraries, generally define classes of buildings that have common, often functional, characteristics. However, the functional classification is not the only aspect of building types. Generally a type can be described as the encoding of prominent features of a design object. Such features include function, form, and context. According to Moneo (1978), a type can be ‘defined as a concept which describes a group of objects

characterized by some formal structure.' This implies a grouping of objects by certain inherent structural similarities. These objects are not isolated from a spectrum of concerns from social activity to building construction. Type as a formal structure is defined by the relationships between all these aspects and the elements that make up the whole. This definition of types as a structure of aspects, elements, and their relationships makes it possible to formalize a building type as a network of components, concepts, and their relationships.

Relationships between types play an important role; a type is related to and dependent on other types. According to Johnson (1994, pp. 347-348), a relationship has first to do with identifying characteristics of elements. These make the elements recognizable as belonging to some family of elements. Second, a relationship relates to the distance between the elements, be it an abstract, conceptual, mathematical, semantic, or physical distance. Relationships between types result in formal and spatial organizations and ordering principles (Ching 1979). Types and their relationships can be represented in a graph system, as nodes and edges.

Type as a concept has no notion of representation. Nor does a typology prescribe a particular semantic structure. The structure may be expressed linearly (figure 2a), or hierarchically, offering various levels of detailing (figure 2b). When parts of the hierarchy are reused at various locations within the structure, a network structure results where types can have more than one 'parent' type (figure 2c). The structure's complexity can be extended or reduced according to individual cases. The overall structure may also constitute a combination of various dependency substructures, describing different aspects or parts of a typology (figure 2d). In this case, the individual substructures may be considered as different dimensions within the semantic model.

We can consider types in their most simplistic form as keywords. Keywords are commonly used as a means for the categorization of documents in document management applications. Elements of such a semantic structure do not necessarily need to be considered conceptually as types in the architectural sense. Types in this context are used to denote the dependency between elements. When these elements are related according to a semantic structure, they are more than simple keywords or attributes. As types are associated to documents, in the form of keywords, relationships between types induce additional relationships between document entities that otherwise do not exist. These additional relationships tighten the information structure already defined by the document entities and their relationships.

## 2.5 **Visualizations**

Imposing a semantic structure on keywords as types also facilitates the assignment of keywords to document entities and components. When keywords are organized in a structure, these are more easily visualized and conceptualized, facilitating a conceptual organization of documents with respect to this semantic structure. In particular, effective visualizations allow efficient and fast access to data, and provide a better overview of data entities (Papanikolaou 2001). Visualizations that facilitate visual exploration and manipulation support the process of relating appropriate keywords to document entities and components. For example, a hierarchical structure

of keywords allows for an effective overview of the entire structure in a single view that can be used when assigning keywords to documents and when creating new keywords within this structure. Even without any control mechanism to ensure the consistency of the positioning of new keywords in the hierarchy, the clarity of the structure enables the user to better determine which location may be appropriate for placing a new keyword in the hierarchy.

## 2.6 **Automation**

Types in architecture usually have various formalizations related to them. Formalizations of types make it possible to search for instances of types within documents of different formats. Since types are conceptual entities, with instances of these associated to design documents, the format of a document defines the respective type's formalization: as a keyword, an image, a sketch, etc. Formalizations of types in different formats can assist in automating the classification of documents by automatically recognizing instances of types within documents. Recognizing instances of types in documents provides both qualitative and quantitative information about the importance of these types for the documents. It also enables a specification of exactly which part of a document a type applies to. This automation facilitates the process of relating and categorizing documents.

The process of document decomposition may be (semi-)automated using pattern recognition mechanisms and artificial intelligence techniques. Image recognition mechanisms for images (e.g., Koutamanis 1995, Barrow and Tenenbaum 1981), shape recognition mechanisms for simple line drawings (e.g., Chase 1989, Krishnamurti 1981), and keyword or concept recognition mechanisms for texts (e.g., Greenberg 1999) will assist in presenting the user with suggestions about document components corresponding to a given categorization. Other formats require similar, though different, recognition techniques. While there has been a lot of research into the field of image and pattern recognition, especially in engineering, remarkably few practical applications of this research in the field of architecture exist. We do however expect these technologies to mature and be able to serve this purpose. These technologies will surely provide a considerable benefit in the uptake of a system utilizing document decomposition.

## 2.7 **Representational Structures**

The representation of a document decomposition requires the definition and recognition of the composing structure and relationships. Components and substructures can be recognized as instances of types. These may be grouped into more complex structures, creating structure to structure relationships corresponding to relationships between types. Substructures may also belong to more than one structure, in reference to the formal structure described by Moneo (1978). Types do not impose any particular representational decomposition on the document or abstraction depicting an instance of a type. Instead, different abstractions require different vocabularies that have their origin in the domains of the respective abstractions. These vocabularies may overlap but, more often, they will offer

alternative descriptions of related types reflecting on the function and context. A syntactical framework that offers representational flexibility is needed to define the vocabularies that express these structures. We propose the adoption of XML (eXtensible Markup Language) (W3C 1997) as a common syntax for describing document decompositions and their integration into a single information structure.

## 3 DESIGN STUDIO APPLICATION

We developed a first prototype for the presentation of architectural analyses on the web in order to illustrate the presented methodology (Tunçer et al. 2001). Ottoman Mosques served as a case study for this work. XML was adopted as a common syntax for the representation of document structures and their integration into a single information structure. Based on its results, we are currently developing a new system for the construction and presentation of a body of architectural analyses in the context of a design studio.

This design studio will start in September 2002 as part of a new curriculum to implement a three + two year bachelor and masters program. The design studio will be offered in the fourth semester to about 350 students. The central design theme of this studio is a "small public building", in particular, a theater. The students will be given a relatively complex functional program and will be requested to design and work out the materialization of this theater.

The students will begin the studio by analyzing selected precedents (historical and contemporary) of the relevant building with respect to various criteria (composition, program, construction, context, type, etc). Documentation of these precedents are presented to the students in the form of drawings, pictures, and texts. Until now, such documentation was commonly provided in the form of a book. In this studio, instead, this documentation will be available on the web within the same environment that the students will use for the presentation of their own analysis results. The result will be a common library such that students, in later design activities, can draw upon other students' results for comparisons and relationships between different aspects or buildings.

The students will be provided with a keyword hierarchy corresponding to a system of architectural types as a structure to hook up their contributions. In general, and depending on their knowledge of the domain, students can collaboratively define and extend this structure. Additionally, the students will be offered a tool for the decomposition of images and texts, such that the various components can be more accurately organized in coordination with the keyword hierarchy. These and other tools are integrated within a presentation environment for analyses. The user interface provides views for individual documents and all their related documents at one glance, and visual overviews of the entire document and keyword structures and their links. We are developing tools to create the keyword structure, and view it as a semantic map; to intuitively decompose documents and relate them with keywords; to generate pages to draw sections and views on a plan, relate the respective documents, and then to generate web pages from these, as entry pages to analyses.

### 3.1 **Representation**

The content of the system is provided as a number of abstractions from selected precedents and a type hierarchy. Abstractions may be decomposed into constituent entities, in correspondence to the adopted typology. Abstractions in the form of images can be broken up into smaller images using an image processing application. Abstractions in the form of text are immediately structured in XML. Currently we consider only text and image abstractions. The result is a component hierarchy, with the top-level elements corresponding to the various abstractions. The type hierarchy depicts the semantic structure for this component hierarchy, with each component entity assigned at least one type from the type hierarchy. The type hierarchy itself can be imported from an external source or collaboratively composed by the authors of the analysis. Both the type hierarchy and the collection of abstractions can serve as access points into the analysis.

The structure of both type and component hierarchies is specified by the XML grammar and encoded in the DTD (Document Type Definition). It specifies the kinds of elements, their properties and attributes, and their possible nesting. Both hierarchies are recursively defined. The type hierarchy is defined in XML by using the type name as the tag and by nesting the elements according to the hierarchy. An ID additionally identifies each type and is used for linking types to components. Components are also identified by an ID; the component hierarchy is defined by using this ID as the index, and by nesting the elements.

In this organization of types and components, various kinds of component relationships can be distinguished. Components are initially related by the abstraction hierarchy these belong to. By assigning types to components, components that share the same type are implicitly related. The type hierarchy further relates components, these relationships are derived from the nesting of the respective types in the type hierarchy. Additionally, explicit relationships can be specified between components.

The resulting XML structure forms a flexible source for further manipulation and traversal. Components can be flexibly categorized and grouped according to their relationships and attributes, offering various views of the information structure. Views can be traversed and linked using both explicit and implicit relationships. The documents are transformed and visualized using XML related developments (W3C 1997).

### 3.2 **Interface**

The interface allows the user to view both the type and document hierarchies and their relationships in an intuitive way. These views include both in-world and out-world views (Papanikolaou and Tunçer 1999). An in-world view presents a component (or type) together with its immediate neighbors within the hierarchy, and displays all other components that share a type with it (figure 3). Such a view allows one to browse the structure, interpret the relationships, and as such lead to interesting out-world views. While the types serve for the most part as binding elements in the structure providing relationships between the components, when traversing the information structure, the content as available in these components is the most

important aspect. As such, while the component's type, and its location in the type hierarchy, may be presented as properties of the component, the relationships are specified primarily as component-to-component relationships. This not only ensures that links are presented as shortly as possible, facilitating a swift traversal, but also shifts the focus onto the content, rather than the structure that surrounds it. Types further serve a role as index to the information structure.

In addition to the different in-world views, structural maps provide visual feedback to the users on their traversals and selected views by presenting the location of the currently viewed node within the hierarchy. Such views also give an overview of the scope and depth of the semantic structure guiding the analysis. Figure 4 presents some exemplar out-world views as clickable maps that offer an overview of the entire type hierarchy in relationship to the related documents.

### 3.3 **Tools**

We are developing and implementing a number of tools, using Java and SVG, in order to facilitate the development of keyword structures and the decomposition of images and texts, and to construct image maps that can serve as guides into parts of the information space. A first tool serves to create the keyword structure, and view it as a semantic map (figure 4b). This tool extends on an existing freeware application for building up and viewing network structures. Another tool assists the user in the decomposition of image abstractions. Image abstractions are decomposed by selecting rectangular areas from the images (figure 5), selecting sets of keywords from the type hierarchy (figure 4b), and attaching these to the image components. The same application also offers a tool for adding hotlinks to images, allowing for the development of image maps that can serve as a content map or index to a collection of related documents. The base image may constitute a plan of a building, markers can then be positioned on the image and related to the appropriate documents. From this information, a web page is generated containing the respective image map (figure 6).
When one moves the mouse pointer over a marker, a preview image of the related document appears. Markers can be clicked to browse to the respective document. Currently we provide for section markers, indicating where on a plan a section is taken, and in which direction (figure 6a), and view markers, defining where a picture or an elevation is located in relation to the plan (figure 6b).

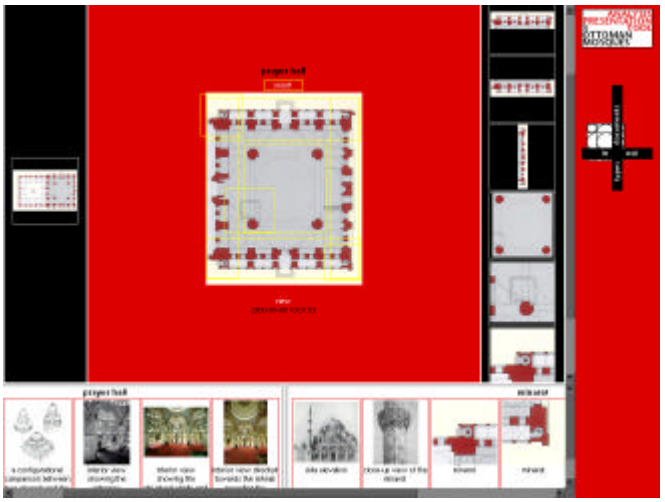

Figure 3: **A snapshot of an in-world view from the prototype application**


Figure 4: **Three snapshots from out-world views of the type hierarchy. The focus of this figure is on the graphical representation of the type structures, not on the types themselves. a) a 2D list view, b) a 2D dynamic tree view, c) a 3D dynamic network view**


Figure 5: **A snapshot from the image decomposition tool showing two rectangular areas drawn on the loaded image in order to create two image components**

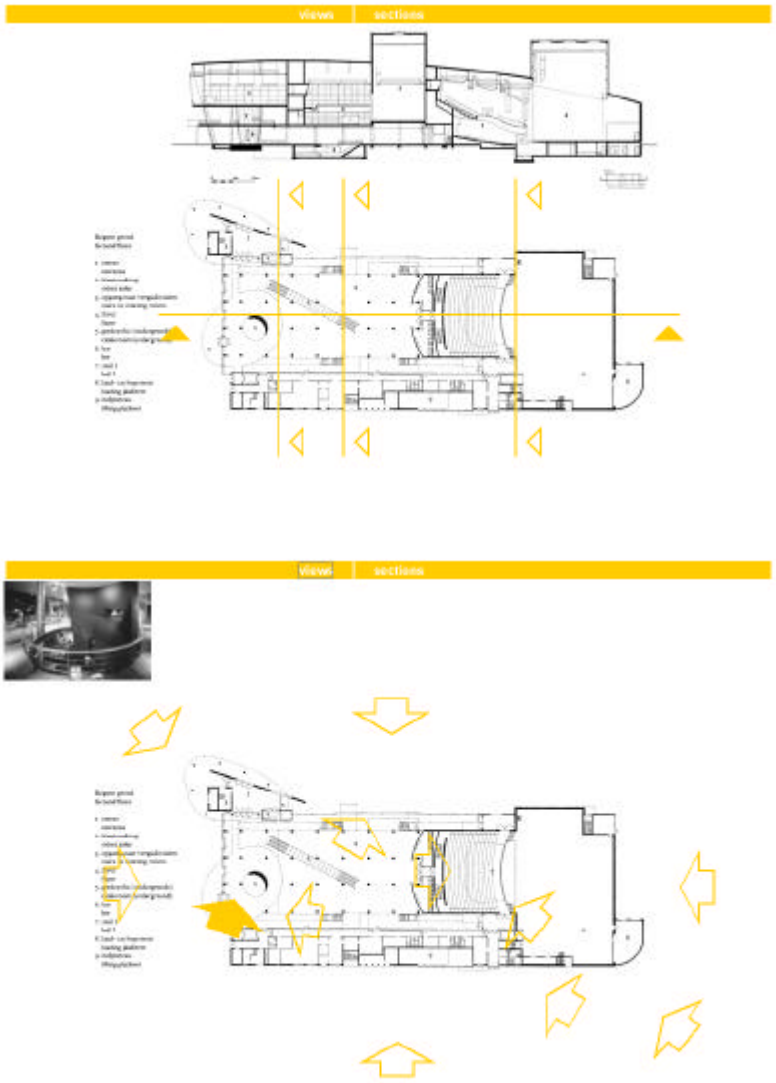


Figure 6: **Generated web pages containing image maps that serve as a content map or index to a collection of related documents. a) image map with section markers, b) image map with view markers**

## 4 CONCLUSION

Analysis plays an important role in design and education. An information structure that integrates the different abstractions or aspects of an analysis, such that the analysis can be interpreted and used in ways other than the original abstractions present, would be particularly useful in an educational setting. As the web gains more importance in all fields, including cooperation in educational projects, providing software that makes it possible for team members scattered over diverse sites to share and manage information while maintaining a comfortable, easy-to-use interface becomes crucial. It seems to us that enriching the information structure both by

detailing the components and by tightening the structure through content relationships would provide a more powerful structure in such a system. Especially in analysis, one is not just interested in one or more specific documents to be processed or built upon, but in interpreting the structure seeking information related to a concept of interest. Targeting a largely unfamiliar audience, the indeterminacy of viewpoints provides the possibility to anticipate individual requests from the audience. Unexpected viewpoints derived from the information can also invoke new interpretations of existing information, which in turn can lead to creative discoveries. In such a context, a rich information structure where the views one can derive are not simply defined by the original documents is particularly worthwhile.

## 5 REFERENCES


Akin, O., M. Cumming, M. Shealey et al. (1997) An electronic design assistance tool for case-based representation of designs, *Automation in Construction* **6**, pp. 265-274.

Barrow, H.G. and J.M. Tenenbaum (1981) Computational vision, *Proceedings of the IEEE* **69(5)**, pp. 572-595.

Chase, S.C. (1989) Shapes and shape grammars: from mathematical model to computer implementation, *Environment and Planning B: Planning and Design* **16**, pp. 215-242.

Ching, F.D.K. (1979) *Architecture: Form, Space and Order*. Van Nostrand Reinhold, New York.

Goldschmidt, G. (1995) Visual displays for design: Imagery, analogy and databases of visual images, in Koutamanis, A., H. Timmermans and I. Vermeulen (eds.), *Visual Databases in Architecture*. Avebury, Aldershot, UK, pp. 53-74.

Greenberg, I. (1999) Facing up to new interfaces, *IEEE Computer* **32(4)**, pp. 14-16.

Johnson, P.A. (1994) *The Theory of Architecture: Concepts, Themes and Practices*. Van Nostrand Reinhold, New York.

Koutamanis, A. (1995) Recognition and retrieval in visual architectural databases, in Koutamanis, A., H. Timmermans and I. Vermeulen (eds.), *Visual Databases in Architecture*. Avebury, Aldershot, UK, pp. 15-42.

Krishnamurti, R. (1981) The Construction of Shapes, *Environment and Planning B: Planning and Design* **8**, pp. 5-40.

Leupen, B., C. Grafe, N. Körnig, et al. (1997) *Design and Analysis*. OIO, Rotterdam.

Madrazo, L. and A. Weder (2001) AALTO on the Internet: architectural analysis and concept representation with computer media, *Automation in Construction* **10(5)**, pp. 561-575.

Madrazo, L. (2000) Computers and architectural design: going beyond the tool, *Automation in Construction* **9(1)**, pp. 5-17.

Moneo, R. (1978) On typology, *Oppositions* **13**, pp. 23-45.

Papanikolaou, M. (2001) IT in virtual enterprises, in Engeli, M. (ed.), *Bits and Spaces: Architecture and Computing for Physical, Virtual, Hybrid Realms.* Basel, Birkhäuser, pp. 172-175.

Papanikolaou, M. and B. Tunçer (1999) The Fake.Space experience - exploring new

spaces, in Brown A., M. Knight and P. Berridge (eds.), *Architectural Computing: from Turing to 2000*. eCAADe and The University of Liverpool, Liverpool, UK, pp. 395-402.

Schmitt, G. (1993) *Architectura et Machina: Computer Aided Architectural Design und Virtuelle Architektur*. Vieweg, Braunschweig, Germany.

Tunçer, B., R. Stouffs and S. Sariyildiz (2001) (Re)presentation of architectural analyses: two prototype applications, in de Vries B., J. van Leeuwen and H. Achten (eds.), *Computer Aided Architectural Design Futures 2001*. Kluwer Academic, Dordrecht, The Netherlands, pp. 495-505.

W3C (1997) *Extensible Markup Language (XML)*. (last accessed: March 19, 2002) http://www.w3.org/XML/