Integrating Architectural Abstractions

Bige Tunçer
Delft University of Technology, The Netherlands

Rudi Stouffs
Delft University of Technology, The Netherlands

Sevil Sariyildiz
Delft University of Technology, The Netherlands

Abstract
Building projects are communicated through project documents. A collection of these documents are stored, related, and managed within digital environments for various purposes. These environments are all concerned with the complexity of organizing an information space: how to organize the information and to relate the individual entities within this organization in order to support effective searching and browsing of the resulting information structure. We present a methodology to handle this complexity through integrating a number of design documents of different formats within a single information structure. When this integrated structure is highly intra-related, it provides support for effective searching and browsing of this information. To achieve such intra-relatedness, we consider a notion of types from architecture as a semantic structure for project document management in the AEC industry. We discuss specific techniques to support this use of types with respect to EDMS's and Web-based project management systems. We describe a prototype application, a presentation tool for architectural analyses, which combines these techniques.

Keywords
Complexity, Information Structure, Architectural Analysis, Flexibility, Effectiveness

1 Introduction

Information about a building project is generally communicated through design documents or abstractions. These documents or abstractions are of various formats, such as drawings, diagrams, 3D models, and texts, and convey different aspects of the project such as functional and organizational relationships, and structural aspects (Schmitt 1993). The organization and management of these abstractions is best supported in a computer medium.

Web-based project management systems are gaining ground as environments for organizing and managing design documents. However, a common problem of such systems is that they either offer only a loose organization of the design documents or, on the contrary, impose a rigid structure. We propose a methodology for increasing the effectiveness of such a system that does not impose a fixed frame of reference. This methodology integrates a number of design documents of different formats within a single information structure. When this integrated structure is highly intra-related, it provides support for effective searching and browsing of this information.

The first part of the paper describes the problem statement in the context of a document-based system. The second part of the paper introduces a notion of types from architecture and explores how this notion can be beneficially applied in the area of information management for the AEC industry. The third part of the paper discusses specific techniques to support this use of types with respect to EDMS's (Electronic Document Management Systems) and Web-based project management systems. The fourth part describes a prototype application, a presentation tool for architectural analyses, which combines these techniques.

2 Complex Information Spaces

Considering abstractions as document entities in an information environment on the Web, image archives, architectural analysis presentation tools, and EDMS's are all examples of environments for storing and relating large collections of abstractions or documents. These documents can be of various formats. While the main uses of these various environments are distinct, as archiving, presentation, and collaboration tool, respectively, these are all concerned with the complexity of organizing an information space composed of a large number of information entities and their relationships. The main question is how to organize the information and to relate the individual entities within this organization in order to support searching and browsing the resulting information structure. Specifically, there is a need for an information organization that enables an outsider to access this information effectively, independent of the viewpoint of the person who conceived the information structure.

A document-based approach, treating the individual documents as entities or objects that are organized and related according to different categories and attributes, offers a flexible organizational framework. This approach has both advantages and disadvantages. Since the semantics of the data is not encoded in the representational structure, it avoids a rigid organizational structure. This separation of syntax and semantics makes it flexible to input and organize the documents. However, the information structure defined by the documents and their relationships is rather sparse. This loose structure offers little support for searching and browsing. Our goal is to achieve a rich information structure that offers the flexibility of a document-based approach, though within a powerful organizational framework. To clarify this goal, we present two exemplar situations that could profit from such a rich information structure.

2.1 Project Document Management

Web-based document management applications are increasingly used to support the building process. Figure 1a offers an exemplar illustration of such applications (Lottaz, Stouffs and Smith 2000). Central to such applications is the consideration of documents or files as the primitive entities in the resulting information structure. Such applications can be augmented with powerful visualizations of the information structure (figure 1b), e.g., offering insight into the depth of a hierarchical organization or into the clustering of documents (Papanikolaou 2001). At the same time, however, the effectiveness of such visualizations is obviously limited by the adopted organization of the information structure. Treating documents as primitive entities does not enable a distinction, with respect to content and structure, of various parts within these documents, resulting in a sparse information structure.

A richer information structure can be achieved by increasing the number of document entities and/or creating a denser network of relationships between documents. For example, by extracting document property values (e.g., categories or attributes) from documents through indexing mechanisms, more explicit relationships between documents can be created. Properties recognized through document indexing, e.g., keywords, however, do not give any information on the importance of the concept described by this keyword for the document, or on which portion of the document it applies to. As such, it is a quantitative approach rather than a qualitative one. Instead, it would be more appropriate to decompose the documents according to content, and to define the document properties with respect to these subdocuments. This will not only enrich the information structure as defined by the documents, but also make the documents inherently related by content.

2.2 Architectural Analyses

Analysis plays an important role in design, research, and education. In education, it is common practice for architecture students to collect abstractions on prominent buildings relevant to their design task in the early stage of design. While practitioners can rely on a body of design experience of their own during the process of a new design, students can only draw from the examples of success and failure from known architects. Since it is wrong to reinvent the wheel over and over again, we should learn from our elders and adopt their successful solutions similar to the ones we cope with (Goldschmidt 1995). In the past, such precedent-based learning was implicit in the master-apprentice relationship common in the educational system. Nowadays academics commonly no longer have the possibility to maintain an extensive design practice, and instead introduce important outside precedents to students. As a result, the study of important historical precedents or designs plays an important role in design instruction and in the students' design processes (Akin et al. 1997).

While there is no doubt that the most effective outcome of such a study would be achieved when the student does the entire study herself, students also benefit from a collaboration with peers, in which they form groups to do an analysis of various aspects of a building or a group of buildings. By integrating the respective results into a common, extensible, library, students can draw upon other results for comparisons and relationships between different aspects or buildings.

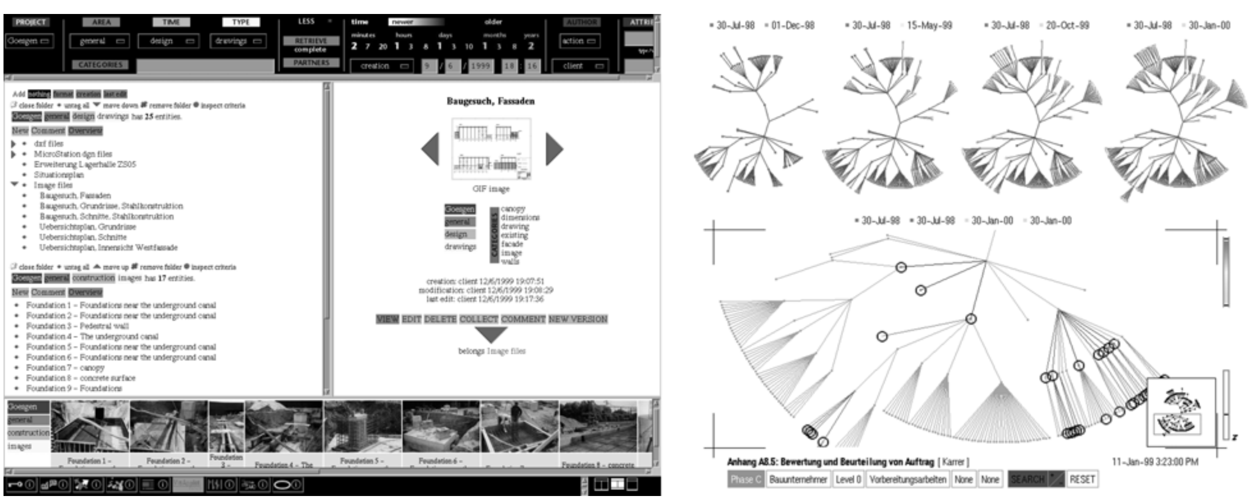

Figure 1. Web-based document management applications: a) Document view from the ICCS project developed at ETHZ, b) Disc view of a document structure from the ITvU project developed at ETHZ.

The Web offers many examples of environments to build up, store, and present architectural analyses on a wide variety of subjects (e.g., Madrazo 2000; Madrazo and Weder 1998). Commonly, these analyses consist of a collection of abstractions describing different aspects of the building. These abstractions exist in a variety of formats. These use keywords to categorize and hyperlink abstractions to support navigation through the information space. More sophisticated examples rely on a database for storage and management of the data, and offer a more complex categorization of the information entities and their relationships. A rich information structure that integrates the different aspects of the analysis, such that the analysis can be interpreted and used in ways other than the original abstractions present, would be particularly useful in education.

3 Types: A Concept From Architecture

Within a discipline, members commonly share a definition and classification of common concepts. This structuring of shared knowledge through common concepts gives insight into that particular discipline (Leupen et al. 1997). Architects generally classify building designs based on spatial and formal features. This classification features the concepts of type and typology.

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 (Madrazo 1995). Building types generally define classes of buildings that have common, often functional, characteristics. For example, we can define museums, offices, or libraries as building types. 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. It is fundamentally based on the possibility of grouping objects by certain inherent structural similarities." Type as a formal structure embraces a vast hierarchy of concerns from social activity to building construction. The relationships between all these aspects and the elements that make up the whole define the formal structure.

Types in architecture assist, besides the communication of shared knowledge, analysis of existing buildings, and design of new buildings (Leupen et al. 1997). In analysis, one gives names to aspects of buildings and describes how these fit into a composition, resulting in an "analytical typology" (e.g., Flemming 1990; Madrazo 2000; Mitchell 1990). In design, a reproducible system of design choices is stored in a "generative typology" (e.g., Achten 1997; Gero 1990; van Leusen 1994). Within a generative typology, a type can be considered as bearing a specific design experience for a specific situation; a design aid.

Studies that make use of typological classification have established a rich body of architectural knowledge. Exporting the notion of classification using types to other domains, such as project document management in the AEC industry, may also deliver important results. We discuss three points where the concept of types is of interest in such a context.

3.1 Separation of Syntax and Semantics

Types in architecture are highly conceptual. Types define classes of design objects that share common characteristics. The designs themselves are represented through design documents, e.g., texts, drawings, models. A design document is associated with a conventional notion of the respective type or class of objects. This association is understood both by the creators and by the users of this object, be it a building, a window, or a chair (Lawrence 1994). Type as a concept has no notion of representation. Instead, relationships between types play an important role. This results in semantic structure of types and relationships. However, it does not impose any particular structure on the design document depicting an instance of a type. The explicit linking between documents and types may be achieved simply through assignment.

An Electronic Document Management System (EDMS) offers a framework for a flexible organization of documents, treating the individual documents as entities or objects that are organized and related according to different categories and attributes. However, simply specifying one or more keywords for each document does not necessarily ensure a powerful organization that successfully assists users in retrieving documents of interest. A semantic network describing the document's composition, as in a product model, is too rigid (Tunçer and Stouffs 2000). Taking a middle way between a collection of categorized documents and a full product model is desirable. Separating syntax and semantics allows the semantic structure to augment the document structure without imposing a specific compositional structure. This separation provides extensibility and flexibility within a system without imposing a fixed frame of reference, as the semantics can easily be altered without an adaptation of the syntactic structure. Types can be imported as a network of concepts, organized according to their relationships and dependencies, and then associated with documents.

3.2 Semantic Structure

We can consider types in their most simplistic form as keywords. Keywords are commonly used as a means for the categorization of documents in EDMS's. An analogy with types adds a notion of intra-relatedness to keywords: a type is related to and dependent on other types. According to Johnson (1994), 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 abstract, conceptual, mathematical, semantic, or physical distance. Relationships between types result in formal and spatial organizations and ordering principles (Ching 1979). For example, relationships can be expressed in the form of a hierarchy. 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. Such a tight information structure provides support for effective searching and browsing of an information space (Tunçer and Stouffs 2000).

The semantic structure may also facilitate the assignment of types to document entities. When types or keywords are organized in a structure, these are more easily visualized and conceptualized. Effective visualizations allow efficient and fast access to data, and provide a better overview of data entities (Papanikolaou 2001). Effective visualizations that facilitate visual exploration and manipulation support the process of relating appropriate types or keywords to document entities.

3.3 Various Formalizations

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. This automation facilitates the process of relating and categorizing documents within an EDMS. It also supports the creation of a component view of a document. Recognizing instances of types in documents provides both qualitative and quantitative information about the importance of a type for a document. Furthermore, it enables a specification of exactly which part of a document a type applies to.

The recognition of document components corresponding to types further increases the intra-relatedness of documents in an EDMS. Going back to the concept of a tight information structure, an enumeration of the different kinds of relationships that exist between documents assists in establishing how the organizational structure supports effective searching and browsing of documents. In this organization, keywords or types, which define the semantic structure, are related within a network. These keywords are associated with documents. Documents that share a keyword are implicitly related. Furthermore, since keywords are organized in a network, their relationships add to the relationships between documents. The level to which this relatedness is considered is flexible. Finally, document decompositions create additional relationships in the form of document component hierarchies.

The result of these various relationships between documents is a tight information structure defined by the relatedness between documents offering 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 document 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 document hierarchy. Instead, components provide direct access to other related components, forming a part of the first component's view. Third, one can access the information structure from alternative views to those that are expressed by the individual design documents. 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 design documents. (Tunçer and Stouffs 2000).

The conceptual nature of types in architecture allows various depictions of types in different formats. When types are represented graphically and textually, one can browse or search a document-based system using any of the available representations of keywords. Such flexible representations are especially interesting for browsing information, when users do not have any specific query in mind (Gross 1995). In an architectural analysis, such uses are plentiful, as users are not only interested in individual design documents but in an interpretation of the entire structure seeking information related to a concept of interest. Graphical representations of keywords, or types, are of great use in such a context.

4 Techniques for Relating Types and Documents

We consider three techniques for achieving a tight information structure. These are the modeling of the type structure, the decomposition of the documents with respect to these types, and the use of recognition algorithms to assist in this decomposition.

4.1 Modeling and Visualizing the Type structure

The relationships between types constitute the semantic structure defined by these types. The form of this structure, however, is not predefined. It may be a linear structure, such as a chronological list of project phases. It may also be a hierarchical structure of types offering various levels of detailing. Furthermore, parts of the hierarchy may be reused as leaf nodes at various locations, resulting in a network structure, where elements can have more than one 'parent'. Elements within such a network may be further individually related, creating an even more complex structure. The structure's complexity can be extended or reduced according to individual cases. The overall structure may also constitute a combination of hierarchies and linear dependencies, describing different aspects or parts of a typology. In this case, the individual structures may be considered as different dimensions within the semantic model.

Elements of such a 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 attributes.

The kinds and dimensions of a type structure results from the modeling of the semantics. The chosen model, however, also has an impact on how the resulting structure is visualized in order to facilitate an effective use of this structure in the process of augmenting the relatedness of project documents. Simple attributes can be presented in a 2D list view. When types have relationships and dependencies, this complexity initiates other ways of visualizing. These visualizations may be 2D or 3D, depending on which best fits the particular purpose (Stouffs 2001). A disc view in which the user can navigate, zoom, and pan seems to be very appropriate in the visualization of hierarchical structures (Papanikolaou 2001). A dynamic visualization for visualizing relationships in a network is very appropriate (Plumb Design 1998).

4.2 Decomposition View of Documents

The use of XML (eXtensible Markup Language) for the purpose of describing a decomposition of documents related with types has many advantages (W3C 2000). One of the strengths of XML for this purpose is its ability to represent information structures: how various pieces of information relate to one another. Once a structure is agreed upon, decompositions of existing documents can easily be expressed in XML. XML also serves to integrate such a decomposition of documents into an existing Web-based EDM environment. When decomposing documents in XML, the effect of this decomposition on the structure and representation of the EDMS can be kept to a minimum. Rather than having to replace a document entity by its composition hierarchy of document components, the XML decomposition can be linked to the document as an attribute, simply as text. By interpreting this document attribute, the decomposed document structure can be retrieved and presented. In this way, both the flexibility and the effectiveness of the EDMS are improved without altering the structure of the EDMS, nor imposing any fixed frame of reference.

Visualization approaches, as mentioned above, can also be integrated into an EDMS in order to improve on its expression. These can be plugged into the EDMS and can work on different levels, by interpreting the component hierarchy and displaying the relatedness of components from different perspectives.

4.3 Recognition of Components and Relationships

The process of document decomposition can be (semi-)automated using pattern recognition mechanisms and AI techniques. Within this paper, we are only concerned with text documents, images, and simple line drawings. Other formats will require similar, though different, recognition techniques. Image recognition mechanisms for images, shape recognition mechanisms for simple line drawings, and keyword or concept recognition mechanisms for texts can assist in presenting the user with suggestions about document components corresponding to given types.

When dealing with texts, neural networks and pattern recognition algorithms can pinpoint keywords in and extract key concepts from documents (Greenberg 1999). Determining which sets of text are related is achieved by identifying content patterns in one set and recognizing the same or similar patterns in other sets. For simple line drawings, shape recognition algorithms can be based on the matching of distinguishable elements in the drawing and the type descriptions (Krishnamurti and Earl 1992; Krishnamurti and Stouffs 1997). In order to automate the process of decomposing images, we propose a four-step approach. Starting with a collection of types whose instances may appear in these images, we proceed from the assumption that each type has an associated set of shapes and forms dependent on the current context that makes it possible to recognize this type within the images.

The first step is to determine the intrinsic structure (Barrow and Tenenbaum 1981) of the scene, reflecting on the spatial properties of this scene. Using image processing and manipulation techniques, the appearance of objects is enhanced and objects' edges accentuated, thereby, providing preliminary object description data such as edges, surfaces, surface orientations and distances. This is done to reduce the large amount of information available in an image and to extract the useful information necessary for the next step. We intend to use neural networks for the manipulation of image data.

The second step is to determine boundaries and regions of the geometry by segmenting and grouping the features in the intrinsic images. The resulting segmented images are formed by gathering the feature elements into sets likely to be associated with meaningful objects in the scene, i.e., edge segments corresponding to polyhedral edges. Some domain-dependent information may be used in this stage in order to determine the type of a boundary curve and to reduce noise. The form and shape information encoded within types plays an important role in providing this domain information.

The third step is the recovery of the geometry or shape of objects that make up the scene, from the line drawings resulting from the previous step. Information about regions and their adjacency, the relationships between boundary lines and vertices, and surface orientation information, enable the building of a geometric representation of the scene.

The last step is to interpret the geometry, matching it with a representation of instances of types that may be in the scene. These matches must subsequently be controlled and validated. The overlaps between the geometries of matches can be optimized. The neighborhood relationships of these geometries can be validated by relying on the relationships of types within the type hierarchy. Shape recognition and artificial intelligence techniques can further be used for the matching itself (Çiftçioglu et al. 1999), and for the control and validation of matches. As an example, neural networks are widely used for pattern recognition (Bishop 1995; Inoue and Urahama 2000).

5 Prototype Application

We are developing an application that will combine the described techniques in the form of a Web-based tool for the presentation of architectural analyses in an educational setting. The analysis presentation tool allows for a decomposition of documents by content using a hierarchical type structure. The input to the application is a set of design documents in the form of images, texts, and simple line drawings, and a type hierarchy. The output is an integrated structure of components and relationships. In between, a number of steps are traversed: documents are broken up into their components, and these components within and between documents are related through types. We are using XML for the purpose of decomposing documents and integrating these into a single structure. Ottoman Mosques serve as a case study for this work.

5.1 Structure

The prototype application specifies two information hierarchies: types and documents (figure 2). The type hierarchy specifies the semantic structure. The document hierarchy is defined by the collection of design documents and their decompositions. Both hierarchies are recursively defined.

The type hierarchy (figure 3) can be incorporated from an external framework or specifically defined corresponding to the subject of the analysis. The latter may require the hierarchy to be constructed across the viewpoints of different groups or users. As a result of the separation of syntax and

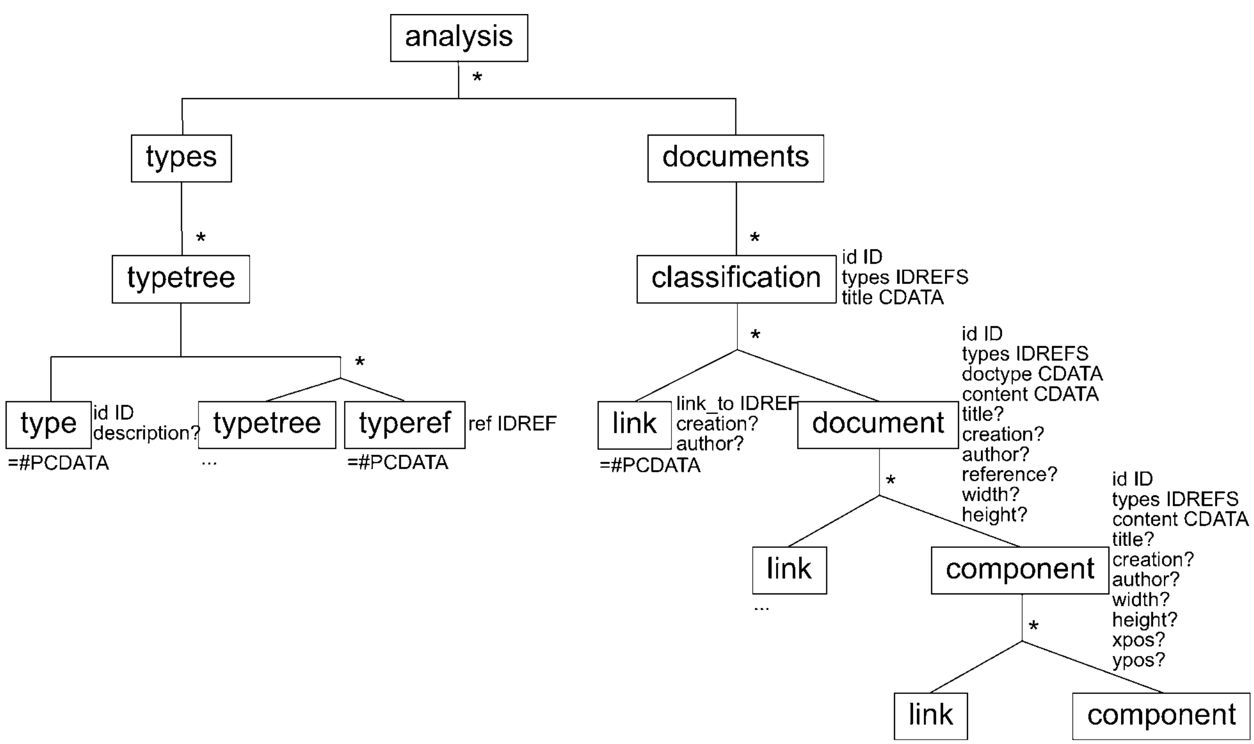

Figure 2. The grammar of the XML structures, the recursively defined type and document hierarchies.

semantics, this construction can easily be achieved, and altered even after documents have been decomposed. The structure is defined in XML by using the type name as the tag, and by nesting the elements according to the hierarchy. Each type is additionally identified by an ID, which is used for linking types to components. Below is a snippet of XML code for the definition of the type hierarchy:

```
<types>
  <typetree>
    <type id=”t166">types</type>
    <typetree>
      <type id=”t70">physical</type>
      …
    </typetree>
  </typetree>
</types>
```

Decompositions of abstractions are expressed in XML. Each component is identified by an ID, and the component hierarchy is defined by using the ID as the index, and by nesting the elements. Types are assigned to components by their ID’s. Below is a snippet of XML code for the decomposition of an image abstraction:

```
<document id=”d6" types=”t68 t66 t31"
doctype=”img” content=”sehzade17"
title=”plan and longitudinal section”
creation=”2000-05-03 15:35:03" refer-
ence=”3" width=”769" height=”1075">
  <component id=”d36" types=”t68t t31t
  t66t” content=”sehzade17-b” title=”plan
  highlighting different zones” cre-
  ation=”2000-05-04 12:49:06" width=”769"
  height=”489" xpos=”0" ypos=”494">
```

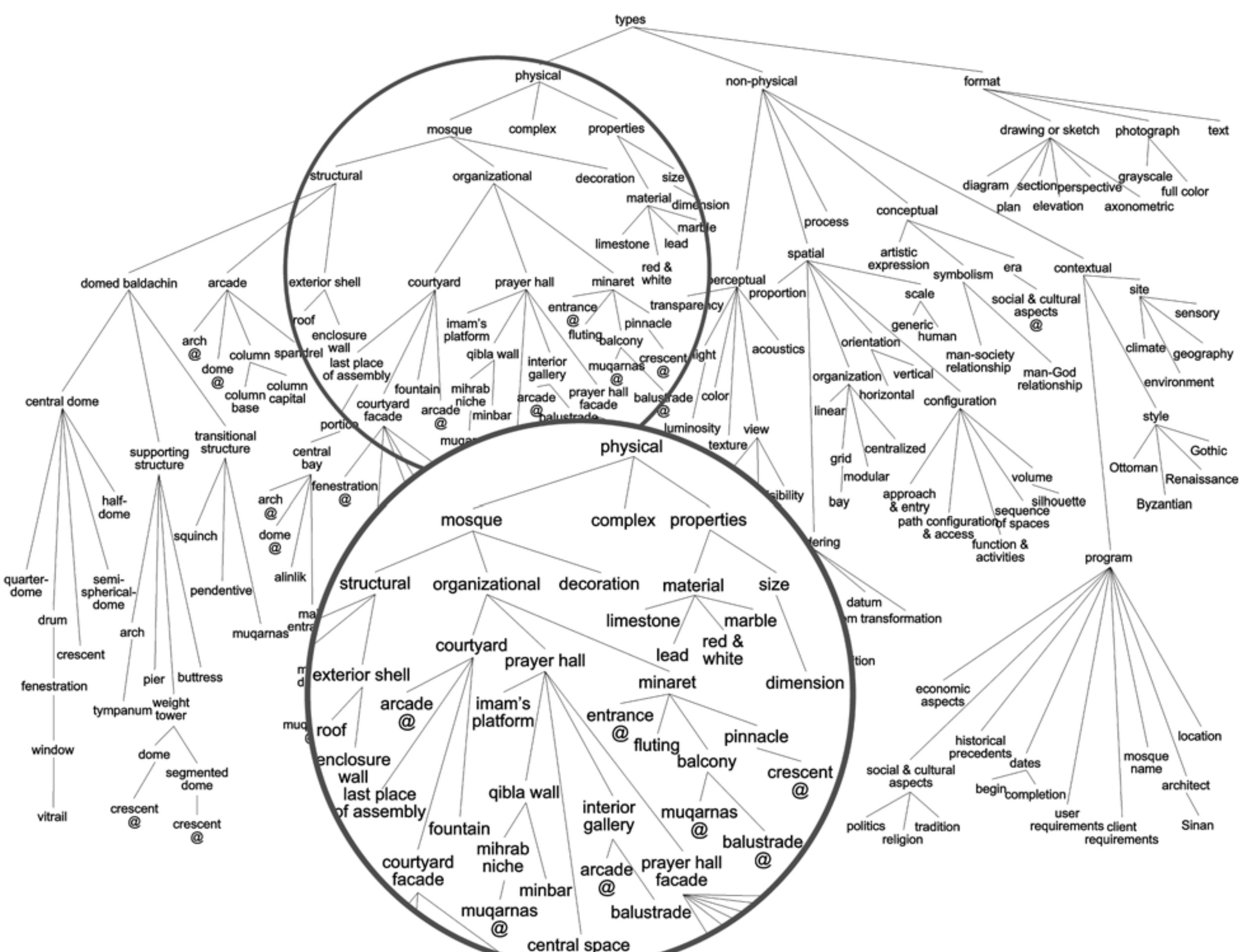

Figure 3. The type hierarchy from the case study, defining keywords that characterize physical and conceptual aspects of the selected buildings. Keywords that are defined elsewhere in the hierarchy are marked by ‘@’.

```
    <component id="d54" types="t48t"
    content="sehzade17-b-2"
    title="courtyard" creation="2000-05-08
    10:00:42" width="423" height="489"
    xpos="15" ypos="494">
    </component>
  </component>
</document>
```

In this organization, the abstraction hierarchy initially relates components. Additionally, components that share the same type are implicitly related. The type hierarchy further relates components; these relationships are derived from the nesting in the type hierarchy. Finally, explicit relationships between components can be specified as references to the component ID's. These are transferred to the XML structure as IDREFS tags (figure 4).

The resulting XML structure offers a flexible source for further manipulation and traversal. Components can be selected 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 XML documents are transformed and visualized through related developments such as XSL, XSLT, Xpointer, and XLink.

5.2 Interface

The system allows the abstractions to be broken up into components through an intuitive interface. Images are decomposed by selecting rectangular areas from the image (figure 5), selecting a set of keywords from the type hierarchy, and attaching these to the image component. Texts are decomposed by selecting a piece of text and attaching keywords to it. Image recognition mechanisms for images, and keyword or concept recognition mechanisms for texts could be used to present the user with suggestions about relevant components.

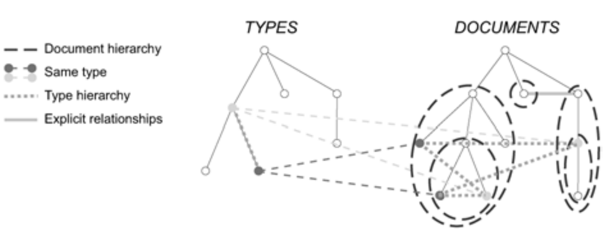

Figure 4. The different kinds of relationships within the structure.

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 6a). The in-world view allows one to browse the structure and interpret relationships, and as such lets the user be guided to interesting out-world views. Types mainly serve as binding elements in the structure providing semantic relationships between components. When traversing the information structure, the content as available in these components is of most importance to the user. As such, while the component's types, and their locations in the type hierarchy, may be presented as properties of the component, its relationships are given primarily as component-to-component relationships. This not only ensures that the links are presented as shortly as possible, tightening the information structure, but it also shifts the focus onto the content, rather than on the structure surrounding it. Types further serve a role as index to the information structure. Access to the analysis is provided through the collection of abstractions and from the type hierarchy.

In addition to the different in-world views, structural maps can provide visual feedback to the users on their traversals and offer selected views by presenting the location of the currently viewed

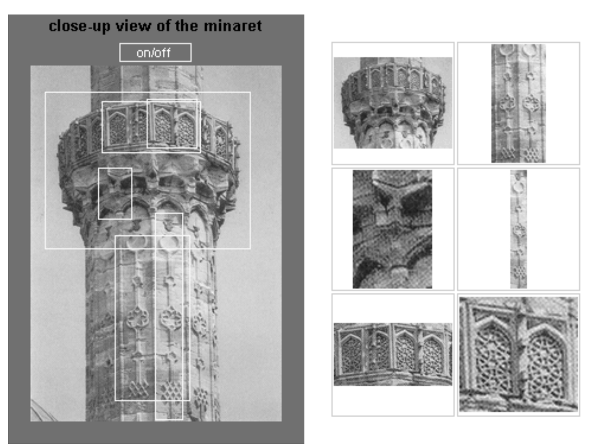

Figure 5. An example from the prototype application showing image decomposition.

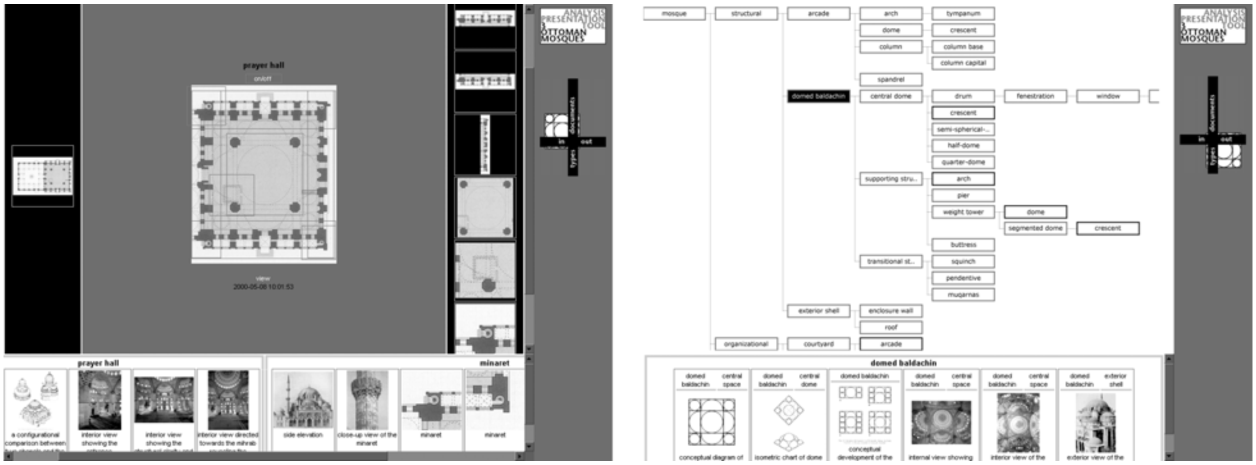

Figure 6. *Two snapshots from the prototype implementation. a) An in-world view, b) An out-world view.*

node within the hierarchy. Such maps can be developed using SVG, X3D, and Java in relation to XML. A out-world view is presented as a clickable map that offers an overview of the entire type hierarchy in relationship to the related documents (figure 6b).

The presented approach provides the users with a simple interface and easy mechanisms for the presentation of an analysis of design precedents, and possibly their own designs. The system is designed in a way that the project grows as users add abstractions from different buildings, even from their own designs. Since all the information is integrated within a single environment, users will benefit from the different studies collected in the analysis, and can draw new conclusions across studies and presentations.

6 Conclusive Remarks

Complexity is a necessary characteristic of information models if they are intended to yield more than a few predefined viewpoints to the information. 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. The most important question is how to achieve such complexity in a simple approach. Hereto, we have described a methodology and its implementation as a tool for the presentation of architectural analyses in an educational setting. Our next step is to undertake an exemplar integration of this methodology into an EDMS in order to augment its capabilities to confirm the applicability of this methodology in this context. Though we have not attempted this yet, we are confident this will be successful mainly because of the advantages of using XML for document decomposition.

There has been a lot of research into the field of image recognition, especially in engineering. Remarkably, there are very few practical applications of this research in the field of architecture. With the advances in Web technologies, many institutions are placing their slide and image archives on the web (de Jong and van der Voordt 2000; Gross 1995). One can expect to have (semi-)automatic recognition mechanisms to be in place for the indexing of these images for effective and efficient retrieval. The functionality of such mechanisms in these environments should be pretty straightforward. We are hoping to have a practical contribution that would be of immediate use in this respect.

Acknowledgements

The images from the ICCS (http://iccs.arch.ethz.ch) and ITvU (http://caad.arch.ethz.ch/research/ITvU) projects appear courtesy of the Chair for Architecture and CAAD, Swiss Federal Institute of Technology Zurich (ETHZ).

References

Achten H. (1997). *Generic Representations*. Ph.D. diss. Eindhoven: Eindhoven University of Technology.

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

Barrow H.G., and J.M. Tenenbaum. (1981). Computational vision. *Proceedings of the IEEE* 69(5):572-595. New York: The Institute of Electrical and Electronic Engineers.

Bishop C.M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon.

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

Çiftçioglu Ö., S. Durmisevic, E. Durmisevic, and I.S. Sariyildiz. (1999). Artificial intelligence in building design. In *ISAMA 99*, eds. N.A. Friedman and J. Barrallo, 113-120. San Sebastian: The University of the Basque Country.

de Jong T.M. and T.J.M. van der Voordt. (2000). Retrieval and reference. In *Ways to Study: Architectural, Urban, and Technical Design*, eds. T.M. de Jong, Y.J. Cuperus and T.J.M. van der Voordt, 29-35. Delft: Faculty of Architecture, Delft University of Technology.

Flemming U. (1990). Syntactic structures in architecture: teaching composition with computer assistance. In *The Electronic Design Studio*, eds. M. McCullough, W.J. Mitchell and P. Purcell, 31-48. Cambridge, Mass.: The MIT Press.

Gero J.S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine* 11(4): 26–36.

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

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

Gross M. (1995). Indexing visual databases of designs with diagrams. In *Visual Databases in Architecture*, eds. A. Koutamanis, H. Timmermans and I. Vermeulen, 1-14. Aldershot, UK: Avebury.

Inoue K. and K. Urahama. (2000). Learning of view-invariant pattern recognizer with temporal context. *Pattern Recognition* 33: 1665-1674.

Johnson P.A. (1994). *The Theory of Architecture: Concepts, Themes and Practices*, 347-348. New York: Van Nostrand Reinhold.

Krishnamurti R. and R. Stouffs. (1997). Spatial change: continuity, reversibility and emergent shapes. *Environment and Planning B: Planning and Design* 24: 359-384.

Krishnamurti R. and C.F. Earl. (1992). Shape recognition in three dimensions. *Environment and Planning B: Planning and Design* 19 : 585-603.

Lawrence R.J. (1994). Types as analytical tool: reinterpretation and application. In *Ordering Space: Types in Architecture and Design*, eds. K.A. Franck, and L.H. Schneekloth, 273. New York: Van Nostrand Reinhold.

Leupen B., C. Grafe, N. Körnig, M. Lampe and P. de Zeeuw. (1997). *Design and Analysis*, 132. Rotterdam: OIO.

Lottaz C., Stouffs R. and Smith I. (2000). Increasing understanding during collaboration through advanced representations. *Electronic Journal of Information Technology in Construction* 5(1): 1-24.

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

Madrazo L. and A. Weder. (1998). AALTO on the Internet: architectural analysis and concept representation with computer media. In *Proc. EuropIA'98, Cyber Design vs. Real Design*, eds. C. Branki and K. Zreik. Paris: EuropIA Productions.

Madrazo L. (1995). The Concept of Type in Architecture: An Inquiry into the Nature of Architectural Form. Ph.D. diss. Zurich: Swiss Federal Institute of Technology.

Mitchell W.J. (1990). The Logic of Architecture: Design, Computation, and Cognition. Cambridge, Mass.: The MIT Press.

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

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

Papanikolaou M. and B. Tunçer. (1999). The Fake.Space experience - exploring new spaces. In *Architectural Computing: from Turing to 2000*, eds. A. Brown, M. Knight and P. Berridge, 395-402. Liverpool, UK: eCAADe and The University of Liverpool.

Plumb Design. (1998). Visual Thesaurus. http://www.visualthesaurus.com/

Schmitt G. (1993). Architectura et Machina: Computer Aided Architectural Design und Virtuelle Architektur, 39. Braunschweig, Germany: Vieweg.

Stouffs R. (2001). Visualizing information structures and its impact on project teams. *Building Research and Information* 29(3): 218-232.

Tunçer B. and Stouffs R. (2000). Modeling building project information. In *Construction Information Technology 2000*, ed. G. Gudnason, 937-947. Reykjavik, Iceland: Icelandic Building Research Institute.

van Leusen M. (1994). *A System of Types in the Domain of Residential Buildings*. Ph.D. diss. Delft: Delft University of Technology.

W3C. (2000). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6-October-2000. http://www.w3.org/TR/REC-xml