MODELING BUILDING PROJECT INFORMATION

Bige Tunçer, Rudi Stouffs
Faculty of Architecture, Delft University of Technology

ABSTRACT: Building projects are expressed in a variety of documents presenting different abstractions of the building. Web-based project management systems are gaining ground as environments for organizing and managing these documents. However, such systems lack the possibility to distinguish and relate different components within these documents. By adopting a uniform language, e.g., XML, as a common syntax to represent these abstractions, documents can be interpreted and broken up into components in order to achieve a richer information structure. These components within and between abstractions can be related, and these relationships added to the representation. The result is an integrated structure of components and relationships, represented in a uniform way, offering new views not inherent in the structure as created by the original abstractions. This paper focuses on some of the representational issues involved in the process of interpreting, breaking up, and relating abstractions.

KEYWORDS: abstraction, document management system, representation, information structure, XML

1. INTRODUCTION

Building projects are expressed through a variety of documents such as drawings, diagrams, models, pictures, and textual information. These documents serve as a medium for communication among the project partners and between the different disciplines involved. From a collaborative perspective, each document reflects on the author's role as well as on the intended meaning. From a representational point of view, these documents present different abstractions of the project, e.g., geometry, structure, context, and functional organization (Schmitt, 1993, 39).

Short of imposing an integrated product model, a 'document-based' approach for managing these abstractions is commonly used. Treated as individual entities, the documents can be organized and related according to different categories and attributes. However, it is not possible to distinguish and relate different components within these documents. Doing so, however, would provide for a richer information structure offering new possibilities for accessing, viewing, and interpreting this information.

To overcome this drawback, we propose the adoption of a modeling language, e.g., XML, as a common syntax to re-represent these abstractions. In this way, documents can be interpreted and broken up into components. These components within and between abstractions can then be related, and these relationships added to the representation. The result is an integrated structure of components and relationships, represented in a uniform way.

This paper focuses on some of the representational issues involved in the process of interpreting, breaking up, and relating abstractions. We illustrate the potentials of this

framework with the representation of a number of abstractions belonging to a body of built architecture, specifically, Ottoman mosques.

2. BUILDING PROJECT INFORMATION

2.1 Current state of electronic document management systems

Electronic Document Management Systems (EDMSs) include functionalities related to the scanning, indexing, organizing, modifying, processing, storing, and retrieving of documents (Megill, 1999, 81). These documents can be of various formats, including text, raster or vector based graphics, and audio or video. These documents are augmented with links, attributes, and methods for viewing this information in a variety of ways. Recently, EDMS developments are gravitating towards the Web. In the form of Web-based project management applications, providing facilities for organizing and viewing documents, and redlining drawings and images, these have also found their way into the AEC industry (Smith, 2000). In a Web-based environment, viewing and editing documents has become more flexible through the introduction of URLs as links, and the possibility to invoke external methods for viewing, editing, redlining, and printing documents according to the file or mime-type.

From the authors' own experience, the Information, Communication, and Collaboration System (ICCS) project (Stouffs *et al.*, 1998) is a good example of such an approach. This is an EDMS that is designed to support communication and information exchange within the Swiss Architecture, Engineering, and Construction (AEC) industry. Here, the documents are organized according to a three-dimensional main organization, and can be further related through relationships (figure 1). Some of these relationships are recognized by the system; these specify the document hierarchy. Other relationships can be additionally defined by the users. Categories and attributes serve to further characterize the documents.

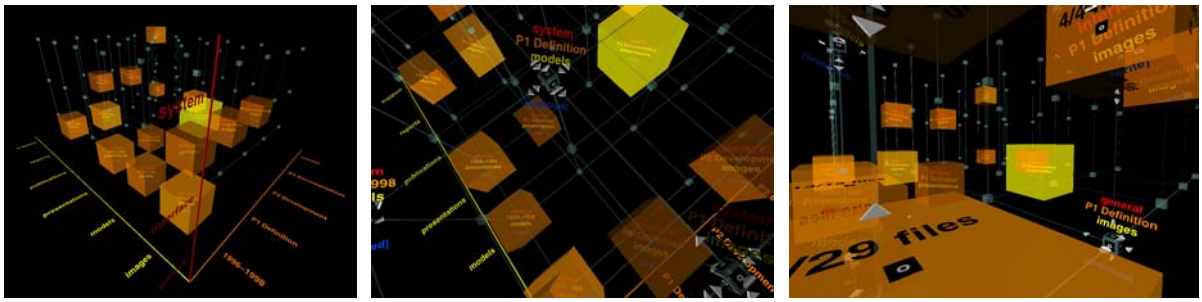

Figure 1. A 3-D representation of the organization of the documents in the ICCS project.

A new development in Web-based document management applications is the adoption of an object-oriented approach. The distributed nature of the Web has people looking for ways to make it easier to transport documents from one system to another. However, the registering of documents in an EDMS is a labor-intensive job. To facilitate these tasks, documents can be treated as objects, describing both the document's content and the operations to manage the object. New Web developments such as XML make it possible for a document to describe all its aspects and links within itself (Laqua, 1999). In this way, documents can be easily transferred and registered into different systems.

2.2 Drawbacks

An 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. A drawback of this approach is that it is not possible to distinguish and relate components within these documents. This results in an information structure, as defined by the documents and the relationships between them, that is rather sparse. Implicit relationships could additionally be recognized based on document property values (e.g., categories or attributes). Such property values can be extracted from the document through indexing mechanisms. Implicit relationships that are recognized from these document properties can also be made explicit by adding them to the document structure. 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.

2.3 Proposal

The goal is to provide increased value for accessing, viewing, and browsing the information structure that is defined by the collection of abstractions. This value can be achieved both by expanding the document structure, replacing document entities by detailed substructures, and by augmenting the structure's relatedness with content information. For this purpose, we propose the adoption of a representational language as a common syntax for describing these document substructures and their integration into a global information structure.

Linking the documents at a component level instead of only at the abstraction level provides for a richer information structure. Such a tightly related structure 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 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 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.

2.4 Advantages

A brief comparison of this approach with a product modeling approach offers the following results. Currently, product models are not widely used in the building industry, unlike EDMSs (Tolman and Ozsariyildiz, 1998). The complexity of building projects and the unstructured and interrelated nature of project data make it very difficult for the building community to agree on a global organization model. Once such a model has been established, all the parties involved in a project must concur to use the model from the very onset of the project. Furthermore, because of the complexity of such models, the learning curve is quite steep. Unless presented with simple interfaces, it will be difficult to achieve adopion by a large community.

At the same time, the power of a product model may not always outweigh the rigidity it introduces. There are situations where product models are not appropriate at all yet, such as a

project in an early stage (Tolman and Ozsariyildiz, 1998) or an analysis of an existing body. In an analysis, one usually does not want to go through the trouble of building up a complete product model.

The approach we propose is not as rigid as a product model. It is flexible, although a semantic guideline is included. It specifies a common syntax for the definition and creation of an integrated information structure that takes a middle way between a collection of documents and a full product model. Navigating this structure must yield, at a minimum, new views of the project information.

3. ENRICHING THE INFORMATION STRUCTURE

The proposed computational framework defines a common syntax for the representation of abstractions. Each abstraction is expressed as a hierarchical composition of components and relationships between these components. Relationships between components from different abstractions serve to collate these abstractions into an integrated structure, from which new views can be derived as collections of components and relationships. In this way, a rich representation can be achieved without imposing a fixed frame of reference.

Within a broader context, we are particularly concerned in this paper with documents that lack any strong inherent structure, i.e., text and images. Both composed of symbols from a relatively small vocabulary, i.e., characters and pixels, in simple and basic 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 in a hierarchical structure. When working with graphical models, the situation is quite different and one would need other, more sophisticated, forms of representation to apply this concept.

3.1 XML as the structuring language

XML (eXtensible Markup Language) is particularly suited for the purpose of decomposing abstractions in the form of text or images and integrating them into a single structure.

3.1.1 Advantages

XML is considered the future language for communication and information exchange on the Web (W3C, 1998; Tidwell, 1999; Veen, 1998). As such, it has the potential of becoming a universal standard for data interchange among applications. An XML document specifies a tree of objects; applications can extract, manipulate, convert, and exchange these objects. Whereas HTML is used for formatting and displaying data, XML represents the contextual meaning of the data; content and presentation are separate. XML is a meta-language that serves to define markup languages for specific purposes. It lets one define her own tags for structuring data. The resulting markup language can be shared among different users active in a same discipline. This way, XML and XSL (eXtensible Stylesheet Language) act as standard ways of organizing and displaying data within a discipline, and interpreting data between different disciplines. The XML structure ensures that the data is consistently organized and is both machine- and human-readable. Furthermore, if the structures agree, XML documents can be plugged into a larger context.

The strength of XML for our purpose is its ability to represent information structures: how various pieces of information relate to one another, in much the same way as a database might. Once a structure is agreed upon, existing documents can easily be converted to XML.

Using tools for scanning texts and images and recognizing keywords, concepts or patterns, such conversion can be automated.

3.1.2 Challenges

While XML has a lot of potential for exchanging and structuring information, there are also a number of challenges. For one, the fact that every user can define her own set of tags is both an advantage and a challenge in the application of XML. The flexibility that XML brings to creating documents can cause conflicts when sharing or blending documents. Namespaces prevent these collisions: an XML namespace is an agreed collection of names that are used in XML documents as element types and attribute names (W3C, 1999). Different domains can define their own namespaces, and many are in the process of doing so (Cover, 2000), including the AEC industry (aecXML, 1999). As with product models, though, it is very hard for a large number of people to agree on a single global structure. Unless this can be achieved, exchange of information may be limited to within an office or a discipline.

3.2 Case

Ottoman Mosques serve as a case study for this work. We have selected three mosques by the same architect, Sinan (1490-1588), that present three different typologies of classical Ottoman architecture in their spatial and structural characteristics (figure 2). These are Şehzade (İstanbul), Süleymaniye (İstanbul), and Selimiye (Edirne). We have chosen these as representatives of a large body of works. It is obvious from the case that the focus of this study is analysis. Analysis plays an important role in design and education. An 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 aspects or abstractions present, would be particularly useful in an educational setting.

Figure 2. Three rows of images from the mosques Şehzade, Süleymaniye, and Selimiye. a) interior space, b) central dome(s), c) dome structure as seen from the outside. Images from Egli (1997), Stierlin (1998; 1985) and Erzen (1996).

3.3 Process and architecture

The input to the proposed system consists of a number of abstractions. The output should be an integrated structure of components and relationships. In between, a number of steps need to be traversed: documents are to be broken up into their components, these components within and between documents related (figure 3).

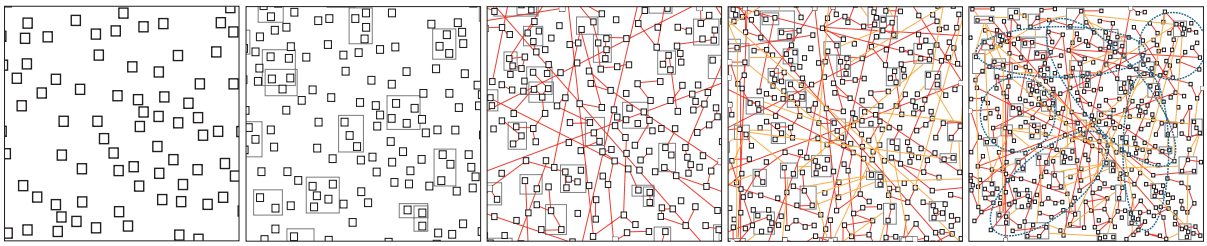

Figure 3. The integrated structure of a collection of abstractions. a) components, b) components grouped into meta-components, c) relationships between components, d) relationships between components and meta-components, e) abstractions distinguished within the network of components and relationships.

3.3.1 Distinguishing components

The first step is to break down the abstractions into a hierarchy of components. Images are broken down into sub-images, composed again through XML structures. Textual entities are treated the same way, i.e., translated into XML structures. This structuring into XML creates relationships between the components within an abstraction, and within the abstraction hierarchy. In other words, a document hierarchy is created where the sub-documents form the lower branches of the tree (figure 4b).

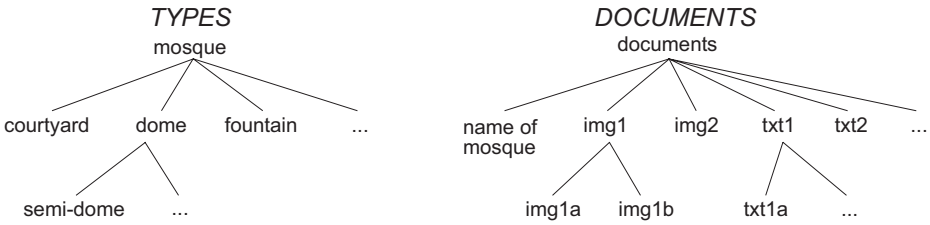

Figure 4. The two main hierarchy systems within the structure. a) the type hierarchy, b) the document hierarchy.

3.3.2 Relating components

The second step is to relate components between abstractions. By distinguishing these components, relationships within abstractions have already been created. However, these relationships are purely syntactical. Unless content is taken into account when specifying relationships, the resulting structure will remain sparse. Instead of specifying such relationships by hand, *types* can serve to automate this process.

Types are the elements that provide the necessary semantics. In general, 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. It might even be said that type is the act of thinking in groups" (Moneo, 1978). In this context, a type can be defined as a building component, or groups of components, such as a dome, a structural pier, a semi-dome, or a decorative element (figure 5). A type can also be

considered as an aspect of a building, such as its location in the urban fabric, or its importance in the social context of the time it was built.

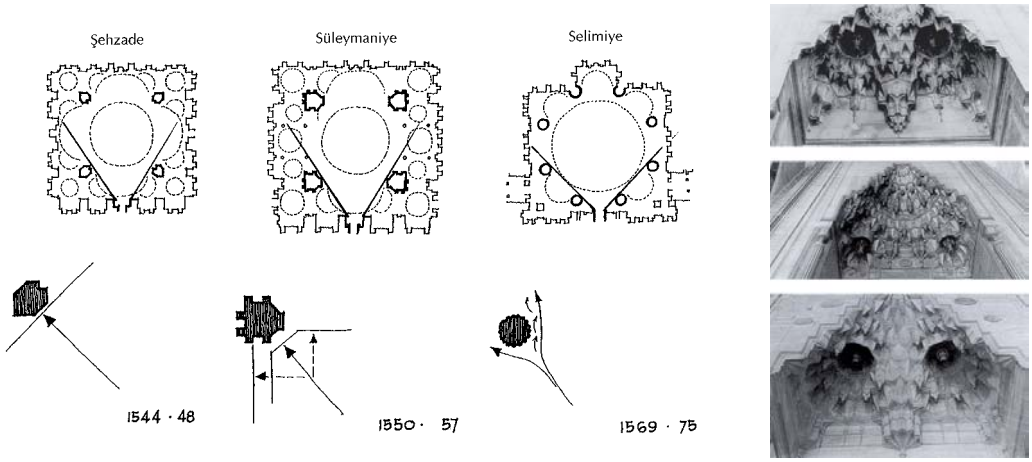

Figure 5. Types and their instances. Left, floor plans and the role of structural piers on the field of vision in three Ottoman mosques; right, muqarnas from the respective mosques. Images from Egli (1997) and Erzen (1996).

Each component has a type specified. Components that share the same type are naturally related. The types themselves are organized in a hierarchy (figure 4a), presenting an additional level of linking among the components, i.e., components are related not only if they belong to the same type, but also if the types assigned to them are related within the type hierarchy. The types and their hierarchy may be incorporated from an external framework as a fixed set if specified. Otherwise, every user or discipline adopts their own set of types and a hierarchy has to be constructed across the frameworks of different disciplines or users.

3.3.3 Relationships

The components are defined in XML as structures of tags and attribute-value pairs, and constructed in a hierarchical manner. This enables a simple matching of components between various abstractions. For the creation of relationships, we therefore suggest a semi-automated approach where some of the relationships between components are automatically deduced from the structures and their composition in the representation.

The different types of relationships between the elements in this organization are as follows (figure 6):

- Components within the document hierarchy: these belong to the same abstraction.
- Components that share the same type.
- Components that relate through the type hierarchy.
- Explicit relationships defined by users: these relationships form the non-automatic links between components.

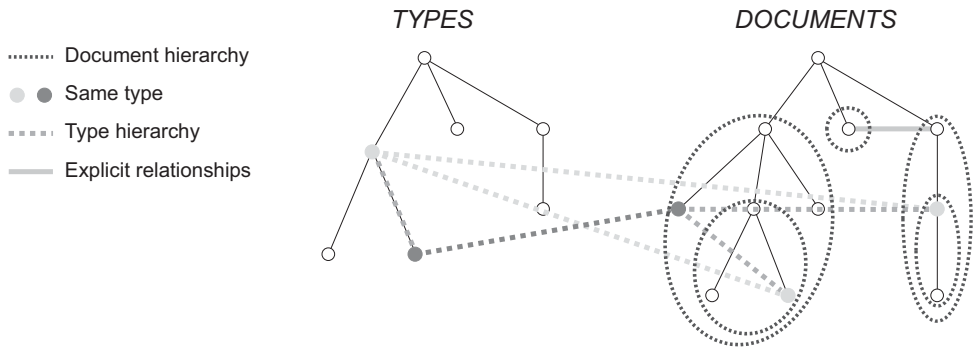

Figure 6. The different types of relationships within the structure.

3.4 Structure and implementation

The grammar of the representational language, i.e., the Document Type Definition (DTD) of the markup language, specifies the structure of both hierarchies in the system. This grammar defines the elements of these structures, their nesting and additional properties, and their attributes. Each document component is identified by an ID; types are identified by name. Types are linked to components as included elements, and explicit relationships between components are referenced through the component ID's. Both hierarchies are defined recursively.

3.5 Presentation

The resulting structure of related XML documents can most easily be presented through XSL. XSL involves the use of style sheets for formatting, and XSLT for creating transformations of the data. These transformations define the result tree: the information in the XML source document is evaluated, rearranged, than reassembled. In this way, a flexible information structure is achieved that can easily be added to, modified, and reordered.

In the presentation, two different types of views can be distinguished, an in-world view and an out-world view (Papanikolaou and Tunçer, 1999). The in-world view concerns a single component and its immediate context in the information structure, i.e., its types and its relationships to other components. The out-world view, on the other hand, considers an 'abstraction' as a collection of components and relationships. This collection may be presented as a single XML document with texts and/or images, and internal hyperlinks defining the structure of this abstraction. Such a collection can be assembled around a type, a concept, or an interesting relationship. The XML document should attempt to present this concept as clearly as possible.

More importantly, though, is the in-world view, as it 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 the binding elements in the structure, providing the relationships between the components, when traversing the information structure, the content as available in these components is the most important entity. As such, while the component's type, and its location in the type hierarchy, may be presented as properties of the component, the relationships should be specified primarily as component-to-component relationships. This will not only ensure that the links are considered as shortly as possible, tightening the information structure, but will also shift the focus onto the content, rather than the structure that surrounds it. Types can further serve a role as index to the information structure.

In order to characterize the in-world view through the context of the component and its relationships, the different categories of these links can be distinguished in the presentation, e.g., components that share the same type, components that share consecutive types, and components that are linked more than once. Furthermore, a structural map can 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 maps can be developed using Scalable Vector Graphics (SVG) and Java in relation to XML. They can show the component location in the document hierarchy, or the type location in the type hierarchy.

4. COMPUTATIONAL RICHNESS IN REPRESENTATION

While each abstraction in a building representation touches upon a different aspect, abstractions relate through commonalities, similarities, and variations in vocabulary, structures, and their relationships. When the abstractions are numerous and diverse, recognizing these relationships can create a tight network, where the individual abstractions no longer stand out. Such a network of abstractions can be said to embody a rich representation (Tunçer and Stouffs, 1999). In order to create a rich representation, more is needed than a collection of texts and images that are decomposed and integrated. Other data formats, such as drawings, models, and numerical analyses must also be taken into account. This requires a different representational language as well as a somewhat different approach to decomposing the abstractions and recognizing the relationships, dependent on the format.

An abstraction can be understood in a syntactic manner as a composition of components and relationships. The representation of an abstraction requires the definition and recognition of its components and relationships. Mechanisms for representing and relating abstractions rely on methodologies for recognizing patterns, variations, and regularities. Components can be recognized as instances of types specified by the user or suggested by the system. Components may be grouped into more complex components, creating type (component) to type (meta-component) relationships. Search and recognition mechanisms can assist the user in relating components within and between abstractions. Different mechanisms may be appropriate for different types of relationships (type relationships, compositional relationships, spatial relationships, etc.).

The recognition of relationships between components from different abstractions is complicated by the differences in representational languages and vocabularies. While the collation of vocabularies assists in the recognition of relationships between components, relating types in different abstractions may also provide insights in the process of collating these vocabularies. Mechanisms for relating components may rely on simultaneous approaches on both levels.

Recognizing these relationships creates a complex network of components and relationships that specifies an integrated structure of the object represented by the abstractions. The representation of this structure specifies a language and vocabulary for this structure. From a representational point of view, this meta-language is a composition of the languages of the original abstractions. New abstractions can be considered as defined by new languages that form part of the meta-language. Such abstractions are important to the user in order to comprehend and interpret the resulting integrated structure. Rather than reducing the rich structure into simpler abstractions, this ability to define new abstractions must be understood as queries to the structure that are unrestricted by the original composition into abstractions. Slicing the integrated structure for a new abstraction relies on the specification of a corresponding vocabulary. According to this vocabulary, types and their instances will be

included or excluded from the section, resulting in a subset of the components and relationships. These define the new abstraction.

The final goal of the project is to derive at a specific implementation, yet from general principles (Tunçer and Stouffs, 1999). It is not the intention to develop a global system that can deal with all abstractions belonging to all sorts of building projects, but to define the representational framework for achieving an integrated structure of components and relationships from a collection of abstractions. The definition of abstractions and mechanisms can then be interpreted and implemented for different building projects or architectural bodies.

5. CONCLUSION

One can debate the advantages and disadvantages of product models versus EDMSs. However, a fact is that, today, product models are mostly absent from the practice while EDMSs gain widespread acceptance. Even when considering only this point, it is worthwhile to look at how an EDMS can be improved to yield a more powerful document structure both at the component level and at the global information level. 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. 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. In such a context, a rich information structure where the views one can derive are not defined by the original documents is particularly worthwhile. We believe that XML as a structuring language is specifically suited to define and develop this information structure when dealing with text and images. Such a system would serve well in an educational context for students to present and learn from existing buildings or bodies of buildings.

REFERENCES

aecXML. (1999). AecXML: A framework for electronic communications for the AEC industries, IAI aecXML Domain Committee.
http://www.aecxml.org/technical/

Cover R. (2000). The XML Cover Pages: Extensible Markup Language (XML), Organization for the Advancement of Structured Information Standards (OASIS), Billerica, Mass.
http://www.oasis-open.org/cover/xml.html

Egli H.G. (1997). *Sinan: An interpretation*, Ege Yayınları, İstanbul.

Erzen J.N. (1996). *Mimar Sinan: Estetik bir analiz*, Şevki Vanlı Mimarlık Vakfı Yayınları, Ankara.

Laqua R. (1999). What Is Happening to EDM, white paper, Gateway Consulting Group.
http://www.gatewaygrp.com/What%20is%20Happening%20to%20EDM.pdf

Megill K.A. and Schantz H.F. (1999). *Document management: New technologies for the information services manager*, Bowker-Saur, London.

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

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

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

Smith S. (2000). Team up on the Internet, *CADALYST 17* (2), 52-58.

Stierlin H. (1998). *Turkey: From the Selçuks to the Ottomans*, Taschen, Cologne.

Stierlin H. (1985). *Soliman et l'architecture ottomane*, Office du Livre, Fribourg, Switzerland.

Stouffs R., Kurmann D., Tunçer B., Mieusset K.H. and Stäger B. (1998). An information architecture for the virtual AEC company, *Product and Process Modelling in the Building Industry* (ed. R. Amor), Building Research Establishment, Watford, UK, 479-486.

Tidwell D. (1999). XML and how it will change the Web, IBM. http://www-4.ibm.com/software/developer/library/xml-web/

Tolman F.P. and Ozsariyildiz S.S. (1998). A product modeling approach to competitive tendering in the building and construction industries, *Product and Process Modelling in the Building Industry* (ed. R. Amor), Building Research Establishment, Watford, UK, 515, 522.

Tunçer B. and Stouffs R. (1999). Computational richness in the representation of architectural languages, *Architectural Computing: from Turing to 2000* (eds. A. Brown, M. Knight, and P. Berridge), eCAADe and The University of Liverpool, Liverpool, UK, 603-610.

Veen J. (1998). Designing for the future with XML, Wired Digital. http://hotwired.lycos.com/webmonkey/98/19/index0a.html?tw=authoring

W3C. (1998). Extensible Markup Language (XML) 1.0, World Wide Web Consortium Recommendation 10-February-1998. http://www.w3.org/TR/REC-xml

W3C. (1999). Namespaces in XML, World Wide Web Consortium 14-January-1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/