(Re)presentation of architectural analyses

Two prototype applications

Bige Tunçer, Rudi Stouffs, and Sevil Sariyildiz
Delft University of Technology

Key words: Representations, Architectural analyses, Information structures

Abstract: We present a methodology for decomposing documents by content and integrating these into a rich information structure. This implies both expanding the document structure, replacing document entities by detailed substructures, and augmenting the structure's relatedness with content information. This paper focuses on some of the representational issues involved in the process of interpreting, breaking up, and relating documents. We describe a prototype application as a tool for building up, storing, and presenting architectural analyses in an educational setting implemented using XML, discuss a similar prototype application to be implemented using *sorts*, and compare these two different methodologies.

1. INTRODUCTION

In education, as in architectural history, theory, and design, complete and thorough analyses of architectural bodies or objects are indispensable. While practitioners commonly draw from a body of design experience of their own to support their current design, students must necessarily rely on the examples of success and failure from other architects. In the past, such precedent-based learning was implicit in the master-apprentice relationship customary in the educational system. Nowadays academics no longer have the possibility to maintain an extensive design practice and students, instead, draw upon a diverse set of precedents from various prominent architects. Thus, the study of important historical precedents or designs plays an important role in design instruction and in the students' design processes (Akin, Cumming, et al., 1997). In this study, students may benefit from a collaboration with peers, by selecting each a different aspect of a same

building, or a different building with respect to the same aspect. By integrating the respective results into a common, extensible, library, students can gain from other results in comparisons and relationships between different aspects or buildings. The complexity of such integration is best supported through a computer medium.

The Web offers many examples of architectural analyses on a wide variety of subjects, with varying degrees of sophistication (Tunçer and Stouffs, 1999). Commonly, such an architectural analysis consists of a collection of abstractions, each reflecting on a different aspect such as function, acoustics, structure, and organizational relationships (Schmitt, 1993). The abstractions may be described in different formats such as drawings, diagrams, models, pictures, and textual information, and individually contained in different documents. These documents can then be categorized and hyperlinked within an organizational structure in order to support navigation through the information space. More sophisticated examples rely on a database for storage and management of the information entities, and offer a more complex categorization of the documents and their relationships. While these studies present effective ways of accessing and browsing information, it is precluded within these analyses to distinguish and relate different components within the abstractions or documents. The result is an information structure, as defined by the abstractions and the relationships between them, that is rather sparse. If enabled, instead, the decomposition of abstractions would offer a richer information structure presenting new ways of accessing, viewing, and interpreting this information.

We propose a methodology for decomposing documents by content and integrating these into a tight structure. This implies both expanding the document structure, replacing document entities by detailed substructures, and augmenting the structure's relatedness with content information. The relationships between the resulting components make the abstractions inherently related by content. This paper focuses on some of the representational issues involved in the process of interpreting, breaking up, and relating abstractions. We describe a prototype application as a tool for building up, storing, and presenting architectural analyses in an educational setting implemented using XML, discuss a similar prototype application to be implemented using *sorts*, and compare these two different methodologies.

We illustrate the potentials of the representational framework with the representation of a number of abstractions belonging to a body of built architecture, specifically, Ottoman mosques. 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 1). These are Şehzade (İstanbul), Süleymaniye (İstanbul), and Selimiye (Edirne).

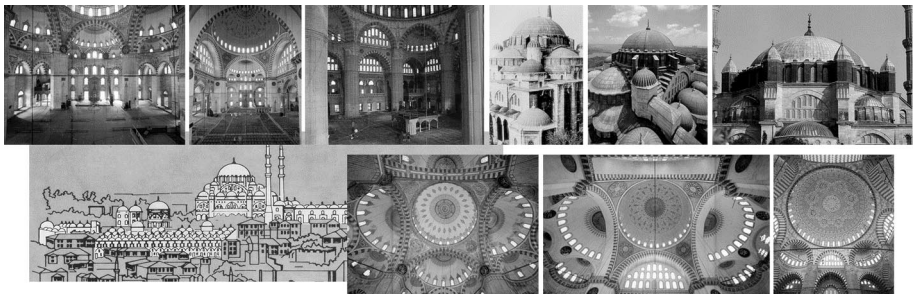

Figure 1. Sets of images from the mosques Şehzade, Süleymaniye, and Selimiye. Interior space, dome structure as seen from the outside, silhouette, and central dome(s). Images from Egli (1997), Stierlin (1998, 1985), and Erzen (1996).

2. DECOMPOSITION BY CONTENT

When using a common syntax to re-represent various abstractions, these can be interpreted and be broken up into components, 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. 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 abstraction 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 hierarchy. Instead, components provide direct access to other related components, forming a part of the first component's view. Third and most importantly, 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.

The input to the proposed system consists of a number of abstractions. Although the approach we propose is completely flexible because it does not impose any fixed frame of reference, only a common syntax for the representation, we do offer a semantic guideline in the form of a hierarchy of types. A type can be defined as a conceptual object that represents the characteristics from a group of similar objects (Tunçer and Stouffs, 2000). 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. The selected type hierarchy (figure 2) provides the information structure with the semantics of how the components within and between the abstractions are related. Each abstraction component is assigned at least one type and components relate through shared types and the relationships in the type hierarchy.

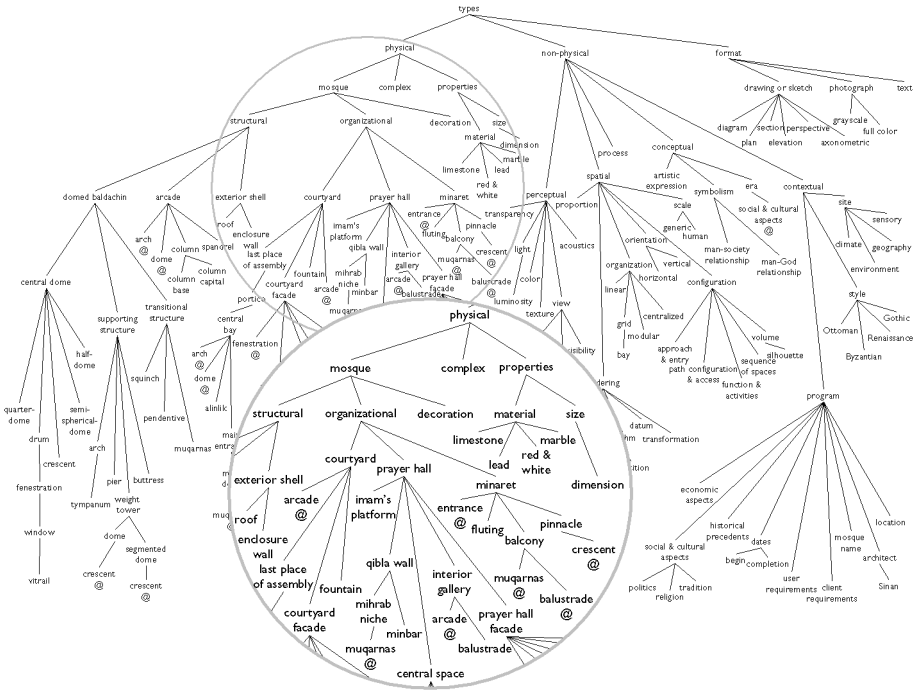

Figure 2. The type hierarchy defined for the case study. It splits into three branches at the top level: physical, non-physical, and format.

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

3. XML PROTOTYPE

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. 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 conversions can be automated.

3.1 Structure

The input to the application is a number of image and text abstractions. We are particularly concerned with texts and images in this application as these lack any strong inherent structure. 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.

The system is composed of two main hierarchies: types and components. The grammar of XML, i.e., the DTD (Document Type Definition), specifies the structure of both hierarchies in the system: their elements, their nesting and additional properties, and their attributes (figure 3). Both hierarchies are recursively defined.

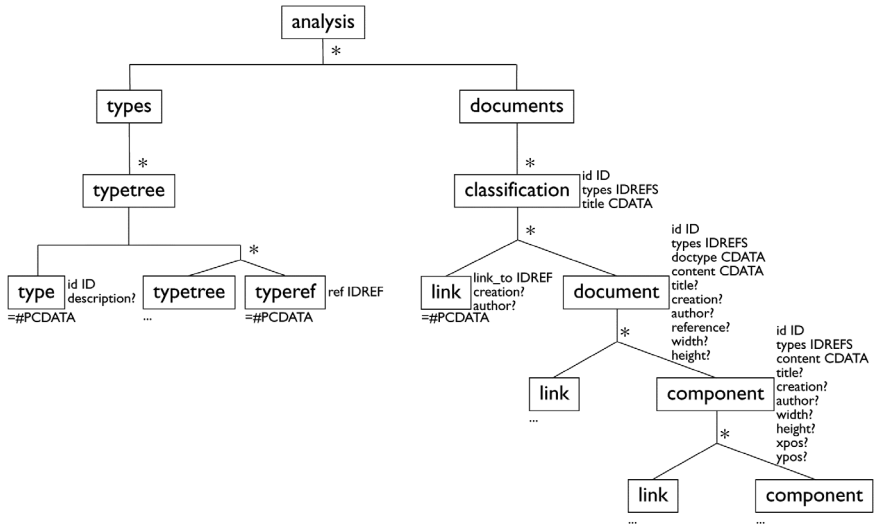

Figure 3. The grammar of the XML structures, the types and components hierarchies.

By distinguishing the components, some relationships within the abstractions have already been created. However, these relationships are purely syntactical. Semantic relationships are added through the hierarchy of types and the assignment of these types to the components. This type

hierarchy may be incorporated from an external framework or specifically defined corresponding to the subject of the analysis. This may require the hierarchy to be constructed across the viewpoints of different groups or users. 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 to components.

The different abstractions are decomposed into their constituent entities defining the hierarchy of components. The abstractions in the form of images are broken up into sub-images by determining the important components, in correspondence to the types, and by cutting them up using an image processing application. The abstractions in the form of text are immediately structured 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.

In this organization, relationships defined by the abstraction hierarchy initially relate the 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 types 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.

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 XML documents are visualized through related developments such as XSL and XSLT, also using XPointer and XLink.

3.2 Interface

The system is Web-based and allows the abstractions to be broken up into components through an intuitive interface. The images are decomposed by selecting rectangular areas from the image, selecting a set of keywords from the type hierarchy, and attaching these to the image component. The 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 the relevant components.

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 neighbours within the hierarchy, and displays all other components that share a type with it (figure 4a). The in-world 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 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 are specified primarily as component-to-component relationships. This does not only ensure that the links are presented as shortly as possible, tightening the information structure, 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. Access to the analysis is provided through the collection of abstractions and from the types hierarchy.

In addition to the different in-world views, structural maps 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 SVG, X3D, and Java in relation to XML. An 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 4b).

The presented approach provides the students with a simple interface and 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, students will benefit from the different studies collected in the analysis, and can draw new conclusions across studies and presentations, including their peers'.

4. SORTS

From a representational point of view, the components and relationships recognized within an abstraction can be said to form a language (Tunçer and Stouffs, 1999), with the vocabulary of the language dependent on the representational format of the abstraction. When abstractions are collated into a single information structure, this structure defines a meta-language that is a composition of the languages of the original abstractions. Consequently, new abstractions can be considered as defined by new languages that form part of this meta-language (figure 5). Slicing the

structure for a new abstraction, then, relies on the specification of a corresponding vocabulary. According to this selected vocabulary, components and relationships will be included into the section, or excluded from it. The resulting structure defines the new abstraction. However, any ability to define new abstractions should not be conceived as the reduction of a rich structure into simpler abstractions once again. Instead, these new abstractions constitute the result of queries to the structure that are unrestricted by the original composition into abstractions.

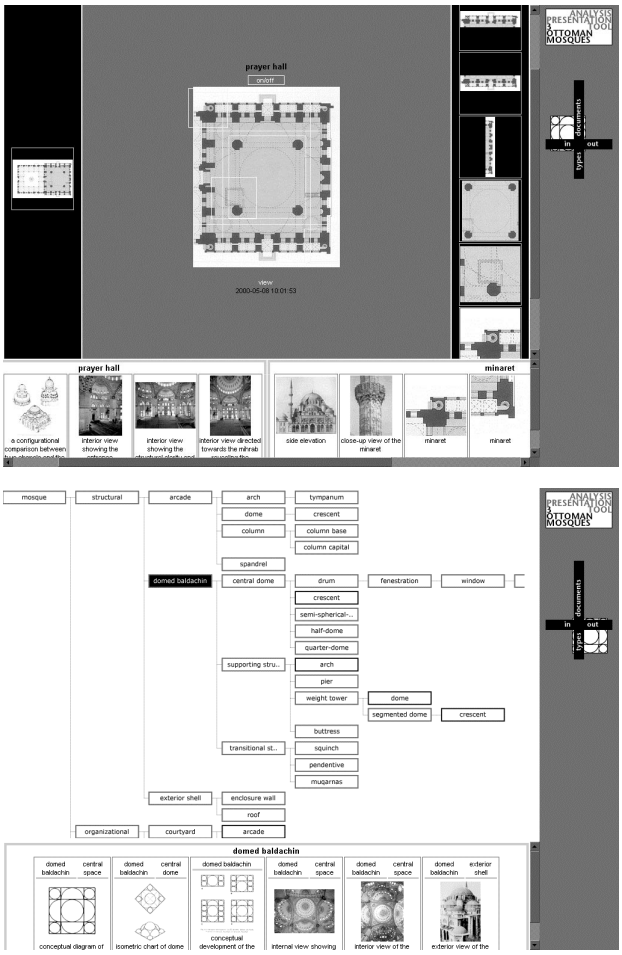

Figure 4. Two snapshots from the prototype implementation. a) an in-world view, b) an out-world view.

Both collating abstractions into a single structure and slicing new abstractions requires a comparison and mapping of the respective languages and vocabularies and the (information) structures expressed in these. *Sorts*,

an approach to representational flexibility (Stouffs and Krishnamurti, 1997), provides support for this. Similar to XML, *sorts* specifies a common syntax, allowing for different vocabularies and languages to be created, compared, and related. Different sorts can be separately conceived for each abstraction and, subsequently, compared for similar components. Building on these similarities, and using an iterative process of development and comparison, a common sort can be arrived at that allows for all abstractions to be represented into a single integrated information structure. Once such a sort has been achieved, the individual abstractions can be mapped onto this sort and their results merged into a single composite structure. New abstractions can be extracted from this structure in a similar process, by conceiving an appropriate sort and mapping this onto the unified sort. No longer restricted by the original abstractions, querying an architectural design or analysis depends on an appropriate expression of the query as a sort, and interpreting the resulting information structure.

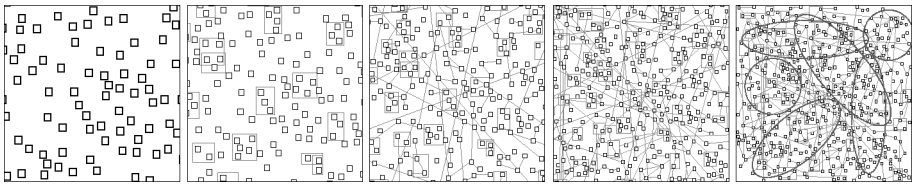

Figure 5. The integrated structure of a collection of abstractions. a) components, b) components grouped into meta-components, c) relationships between components, d) relationships between components and meta-components, e) abstractions distinguished within the network of components and relationships.

A sort is defined as a composition of other sorts under formal compositional operations, elementary data types define primitive sorts (Stouffs and Krishnamurti, 1997). Examples of such compositional operations are an operation of sum, allowing for disjunctively co-ordinate compositions of sorts, where each sort may – but does not have to – be represented in the data form, and an attribute relationship, providing for (recursively) subordinate compositions of sorts in both one-to-many and one-to-one instantiations. Other compositional operations are also considered, such as an array- or grid-like composition of sorts. The definition of a sort also includes a specification of the operational behavior of its members and collections thereof for common arithmetic operations. This behavioral specification enables a uniform handling of forms of different sorts. A primitive sort has its behavior assigned in order to achieve a desired effect; a composite sort receives its behavior from its component sorts, based on its compositional relationships (Stouffs and Krishnamurti, 1997).

Corresponding the representational format of an abstraction, a sort is defined from an appropriate selection of primitive sorts and a definition of its composition using formal operations. The formal character of these operations enables the recognition of formal relationships between different compositional structures that provide for the comparison and mapping of the respective sorts. Comparing sorts also informs on potential data loss when converting information structures between these sorts. The behavioral specification of sorts supports the mapping of data onto and between different sorts such that the resulting information structures are conform to the definition of the respective representations or sorts. Thus, different abstractions can be mapped onto a single sort and their information structures integrated without information loss, provided a careful selection of the sorts involved. Deriving a new abstractions corresponding a specific vocabulary follows the same process of development of a sort and of mapping the global information structure onto this sort.

5. CONCLUSION

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. Furthermore, as the World Wide Web gains more importance in all fields, including collaboration 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. 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.

We believe that XML as a structuring language is specifically suited to define and develop this information structure when dealing with otherwise unstructured information, such as texts and images. However, XML does not provide any information on how to manipulate its data and, as such, is ill-suited to represent complex geometrical data. Analyses commonly include other data formats than texts and images, such as drawings, models, and numerical analyses. Integrating these into a rich representation suggests

a different representational language and requires a different approach for decomposing the abstractions into conceptual entities and for recognizing the relationships between these entities, both dependent on the format. XML and *sorts* both offer a hierarchical description of a data structure in terms of substructures, and enable the mapping of data structures by comparing their hierarchical descriptions. The conceptual framework behind *sorts* offers better support for complex entities, as it allows a behavioural specification of entities. This prompts us to develop the comparison and mapping of data structures using sorts, in order to take advantage of the additional capabilities of this framework. By comparing and evaluating the two presented approaches, resulting in two different applications, we also intend to extract key requirements concerning the representation and presentation of architectural analyses.

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.

6. REFERENCES

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, p. 265-274.

Egli, H.G., 1997, *Sinan: An interpretation*, Ege Yayinlari, Istanbul.

Erzen, J.N., 1996, *Mimar Sinan: Estetik bir analiz*, Şevki Vanli Mimarlik Vakfi Yayinlari, Ankara.

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

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

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. and R. Krishnamurti, 1997, "Sorts: A concept for representational flexibility", in: Junge (ed.) *CAAD Futures 1997*, Kluwer Academic, Dordrecht, p. 553-564.

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

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