Bige Tunçer
b.tuncer@bk.tudelft.nl
Faculty of Architecture
Delft University of Technology

Rudi Stouffs
r.stouffs@bk.tudelft.nl
Faculty of Architecture
Delft University of Technology

A Representational Framework for Architectural Analysis

Abstract

Architectural objects are expressed through a variety of abstractions, each presenting a different aspect. In an architectural analysis, abstractions can be treated as individual entities, categorized, and hyperlinked within an organizational structure. However, such systems lack the possibility to distinguish individual components within the abstractions and to relate these within and between abstractions. Instead, by adopting a uniform language such as XML as a common syntax for representing these abstractions, these can be interpreted and broken up into components, these components related, and the relationships added to the representation. The result is a richer information structure: an integrated structure of components and relationships represented in a uniform way. This information structure can provide new views not inherent to the original structure of abstractions, offering new interpretations that can lead to new abstractions. This paper discusses a prototype application for representing abstractions using XML, and the strengths and limitations of XML for this task.

Resumen

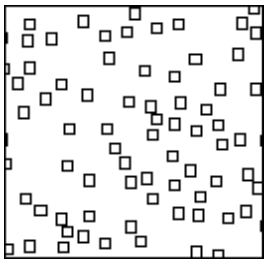
Los objetos arquitectónicos se expresan a través de una variedad de abstracciones, cada una presentando un aspecto diferente. En un análisis arquitectónico, las abstracciones pueden tratarse como entidades individuales, y ser categorizadas e hyperlinked dentro de una estructura organizacional. A tales sistemas les falta sin embargo, la posibilidad para distinguir los componentes individuales dentro de las abstracciones y relacionar éstos dentro de y entre las abstracciones. En cambio, adoptando un lenguaje uniforme, como XML, como una sintaxis común para representar estas abstracciones, éstas pueden interpretarse y partirse en componentes, estos componentes pueden relacionarse, y las relaciones pueden agregarse a la representación. El resultado es una estructura de información más rica: una estructura integrada de componentes y relaciones representadas de manera uniforme. Esta estructura de información puede proporcionar nuevas nociones no inherentes a la estructura original de abstracciones y estas nociones pueden ofrecer interpretaciones nuevas dando lugar a nuevas abstracciones. Este trabajo discute una aplicación prototipo para representar abstracciones que usan XML, así como las ventajas y limitaciones de XML para esta tarea.

Architectural analysis / abstractions

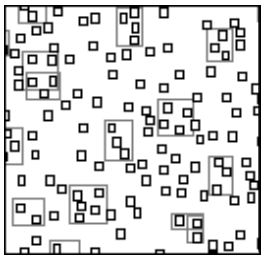
In education, as in architectural history, theory, and design, complete and thorough analyses of architectural bodies or objects are indispensable. An architectural analysis is commonly expressed through a number of abstractions. An abstraction can be a text, drawing, diagram, model, picture, etc. Each abstraction describes a specific aspect of the object, such as function, acoustics, structure, and organizational relationships (Tunçer and Stouffs, 1999). Treated as individual entities or documents, these abstractions can be categorized and hyperlinked within an organizational structure. The Web offers numerous examples of such presentations of architectural analyses (Tunçer and Stouffs, 1999). These studies provide effective ways of accessing and browsing information. However, in such environments, the resulting information structure is generally rather sparse: the organizational structure does not allow different components within the abstractions to be distinguished and related. If enabled, this would offer a richer information structure providing new ways of accessing, viewing, and interpreting the information.

We are developing a prototype application for the presentation of an architectural analysis based on such a rich information structure. Though within a broader context, here we are particularly concerned with documents that lack a strong inherent structure, i.e., text and images. Both can be represented in a similar structure and operated on in a similar way: divided into smaller parts and the parts organized in a hierarchical structure. XML (eXtensible Markup Language) is particularly suited for the purpose of decomposing abstractions, both in the form of text and images, and integrating them into a single structure. XML is a meta-language that serves to define markup languages for specific purposes. By specifying a grammatical structure of markup tags and their composition, a markup language is defined that can be shared with others. Existing documents can easily be converted to XML. Using tools for scanning texts and images and recognizing keywords, concepts or patterns, such a conversion can be automated. 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.

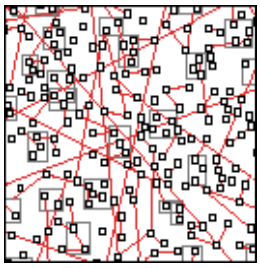
figure 1 - The integrated structure of a collection of abstractions.



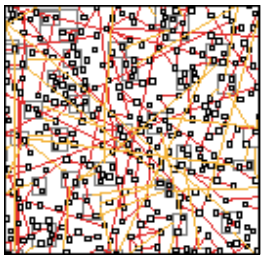
a) components



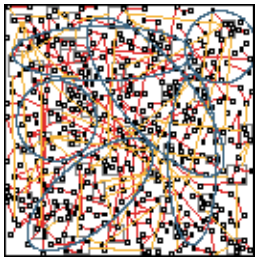
b) components grouped into meta-components



c) relationships between components



d) relationships between components and meta-components



e) abstractions distinguished within the network of components and relationships.

Rich information structures

Information structures are created, at a minimum, by a collection of information entities (including authoring and other attribute information), an organization of these entities, and a specification of the relationships between these entities (Tunçer et al., 2000). In the context of an architectural analysis, the individual abstractions, with their related information, and the relationships between abstractions define the information structure. A richer information structure can be achieved both by expanding the abstraction structure, replacing abstraction entities by detailed component substructures, and by augmenting the structure's relatedness with content information.

In a syntactic manner an abstraction can be understood as a composition of components and relationships between these components. We propose the adoption of XML as a common syntax for describing these component substructures and their integration into a global information structure. By using XML to re-represent abstractions, these can be interpreted and broken up into components (Tunçer and Stouffs, 2000). Furthermore, abstractions can additionally relate through commonalities, similarities, and variations in components. In XML, these components can be related within and between abstractions, 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; 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.

Representing architectural abstractions / prototype application

The input to the prototype application 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: abstractions are to be broken up into their components, these components within and between documents related (Tunçer and Stouffs, 2000) *(figure 1)*.

The system is composed of two main hierarchies: types and components. For easy handling these are initially referenced and linked in the database, and later converted into XML structures. The grammar of XML, i.e., the Document Type Definition (DTD), specifies the structure of both hierarchies in the system: their elements, their nesting and additional properties, and their attributes. Both hierarchies are recursively defined.

The hierarchy of types provides the semantics of how the components are related (Tunçer and Stouffs, 2000). This 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. In the database, the hierarchy is defined by specifying a parent ID for each subtype. This structure is transferred to XML by using the type name as the tag, and by nesting the elements according to the hierarchy. Types are linked to components as included elements.

The different abstractions are broken down into their components defining the component hierarchy. Each component is identified by an ID. The abstractions in the form of images are broken down into sub-images by determining the important components, in correspondence to the types, and by cutting them up using an image processing application. Similar to the type hierarchy, these components are recomposed in the database by specifying a parent ID for each image component. Additionally, the top level abstractions are given a special attribute for easy retrieval. Subsequently, each component is linked to at least one type. The resulting component hierarchy is converted to an XML structure by using the ID as the index, and by nesting the elements. Related types are specified as included elements. Conversely, the abstractions in the form of text are immediately structured in XML. Components within these can be referenced in the database by accessing them as URL's. Similar to the image abstractions, components are related to types, and this information is specified in the XML structure through included elements.

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 XML structures form 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.

Representational flexibility / 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). When the abstractions are numerous and diverse, recognizing their relationships and collating the abstractions into a single structure creates a tight network, where the individual abstractions no longer stand out. Such a network of abstractions can be said to embody a rich representation. The representation of this network specifies a language and vocabulary for the structure. This meta-language is a composition of the languages of the original abstractions.

However, when working with graphical models, the situation is quite different and one would need other, more sophisticated, forms of representation in order to apply this concept. Analyses commonly include other data formats than texts and images, such as drawings, models, and numerical analyses, integrating these into a rich representation. This requires a different representational language as well as a somewhat different approach for decomposing the abstractions and recognizing the relationships, as dependent on the format. In this context, XML is no longer sufficient as the common representational language.

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), can provide support for this. Sorts specifies a common syntax, allowing for different vocabularies and languages to be created, compared, and related. Hereto, it specifies formal operations on sorts and recognizes formal relationships between sorts. Primitive sorts correspond to simple data types, including a behavioral specification. Primitive sorts combine into composite sorts under the formal operations, their behaviors derive from the behaviors of the component sorts. Whereas the formal operations and relationships enable the comparison and mapping of vocabularies and languages, the behavioral specification of sorts supports the mapping of information structures onto different languages.

References:

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

Tunçer B. and R. Stouffs (2000). "Modeling building project information." In *Construction Information Technology 2000* conference proceedings, 28-30 June 2000, Reykjavik, Iceland.

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

Tunçer B., R. Stouffs and I.S. Sariyildiz (2000). "Collaborative information structures: educational and research experiences", In *Analysing and Modelling Collective Design Workshop, Fourth International Conference on the Design of Cooperative Systems*, 20-28. Sophia Antipolis, France.