

ADDED VALUE OF XML FOR CAAD

J.P. VAN LEEUWEN AND A.J. JESSURUN
Eindhoven University of Technology, The Netherlands
Faculty of Building and Architecture, Design Systems Group
<http://www.ds.arch.tue.nl>

Abstract. In early stages of the design process, designers' activities concentrate on trying to understand the design problem and to identify the concepts that make up design solutions. Information is dealt with in these early stages in a very flexible manner that asks for flexible computer support. XML offers much of this flexibility. This paper discusses how XML technology can be used to develop computer support for creative architectural design.

1. Information Systems for Design Support

The way information is dealt with during design, especially in early stages of design, imposes specific requirements on computer support for design. The nature of these requirements and possible solutions for the development of computer support for early design has been the topic of research by [Eastman et al. 1995], [Ekholm and Fridqvist 1998], [Fridqvist 2000], and [Hendricx 2000]. In the VR-DIS¹ research programme at Eindhoven University of Technology, one of the major targets is to support early design in a VR environment [Achten, de Vries, and van Leeuwen 2000]. The underlying Design Information System for this environment supports the dynamic way of dealing with information that is characteristic for early conceptual design.

The rationale of design in these early stages is not static; the identification of design concepts is an important aspect of the early design process. Design rules are being established, sometimes followed, sometimes violated in the search for creative solutions to the design problem (which itself is not static either). Computer support for this dynamic process cannot build on static data-definitions and predetermined classifications of design

¹ VR-DIS: Virtual Reality – Design Information Systems

elements. Defining the concepts and typologies is an essential part of the design process; computer aided design therefore must involve in supporting this process of concept definition.

A number of research projects in the VR-DIS programme concentrates on this form of computer aided design. The aspect of human-machine interfaces to complex, abstract design data is addressed by [Coomans and van Leeuwen 2001]. The development of design methodologies and knowledge representation based on these innovative modelling techniques are discussed in [Achten and van Leeuwen 2001]. This paper concentrates on the technical aspects of the information modelling approach.

2. XML-Schema as a Conceptual Modelling Language

eXtensible Markup Language, abbreviated XML, is a restricted form of SGML that can be used to define the logical structure in documents and constraints on the contents of documents [W3C-XML 1998]. XML provides a syntax and generic mechanisms to structure data in documents. Documents that are structured in XML basically have a tree-like structure but may also contain references; XML is similar to, but syntactically richer than HTML. Any kind of data may be contained in an XML document, as long as it is structured in elements (tags). Attributes can be added to the elements to further provide information about their content, for example:

```
<space type="office" area="23.5">Room 4.07</space>
```

In this example, space is an element with attributes type and area. The contents of the space element is Room 4.07. An XML document is well-formed if its contents complies with the syntax defined for XML. This means that a well-formed XML document has *some* kind of logical, tree-like structure, that is formatted using the XML syntax of elements, attributes, etc. An XML document may also be valid if its contents complies with the constraints defined for that particular type of document. Document types are defined in so-called DTD's (Document Type Declarations). Validation of an XML document with its DTD involves checking whether the right elements and attributes are used in the document. It provides a much more strict level of syntax checking than the XML well-formedness check does. A disadvantage of the DTD approach, however, is that DTD's are written in a different language than XML, which is not very practical. Moreover, the DTD language provides only a few data types.

As an alternative to DTD's, XML-Schema [W3C-XML-Schema 2000] has been developed to define the logical structure of XML documents. Just like a DTD, an XML-Schema defines logical structures and constraints on data contents that can be used by XML documents. An XML-Schema is

itself written in XML. The underlying vehicle for the functioning of XML-
Schema is XML namespaces [W3C-Namespaces 1999]. A namespace
defines a scope for elements and attributes in an XML document. In addition
to what can be declared using a DTD, XML-*Schema* provides a larger set of
data types and allows inheritance in types. Inheritance makes it possible to
declare restrictions and extensions of types.

3. Conceptual Modelling for Design

The abilities offered by XML-*Schema* for extending a conceptual model
with new typologies to be used in XML documents, answer remarkably well
the needs of flexible information modelling for design support. In the VR-
DIS programme, these needs are addressed by a technique called Feature-
Based Modelling (FBM). This technique has its origins in mechanical
engineering, but has been adapted to the context and requirements of
architectural design by [van Leeuwen 1999]. The basis of this technique is
object orientation, but additional functionality provides a high level of
flexibility and extensibility to the modelling environment. Objects in this
approach are called Features, or Feature instances, classes of objects are
called Feature types. The dynamics of the approach are manifested in
extensibility of the set of Feature types and the flexibility of the Feature
instances of these types.

3.1 EXTENSIBILITY

Extensibility involves giving the designer the tools to define new Feature
types that represent design concepts. Compared to pure object-oriented
systems, this aspect allows the user of the system to have influence on the
class definitions, defining new classes from scratch or deriving new classes
from existing ones [van Leeuwen and de Vries 2000].

3.2 FLEXIBILITY

Flexibility is provided by the Feature-based approach in a number of ways.
First of all, relationships in the information model are based on references,
which ensures a maximum of flexibility concerning sharing of information
between elements of the model. Both in the definitions of Feature types and
in the composition of Feature instances, all relationships are defined as
references to types or instances. For example, characteristics such as colour
and durability are not properties owned by building elements, but are
properties referenced by building elements. This makes it possible for
elements to share properties, which is very similar to the way we speak

about these elements: the doors in the building have the colour blue (many doors, one colour).

A second aspect of flexibility is that designers can add relationships to Feature instances that have not been defined in the corresponding Feature types. This provides a means to model ad-hoc design information without the need to first formalise a new typology or modify an existing one. Again, this is much like the way we think about objects around us: this element is a roof, but, in addition to my general concept of a roof, this one has solar panels attached to it. The relationship between the roof and the panels is not defined at the type level, but added at the instance level. Compared to pure object-oriented approaches, this kind of flexibility allows the user to supply properties to an object that are not defined in its class.

4. XML for Design Concept Formalisation

4.1 THE FBM FRAMEWORK

The FBM framework comprises three levels of information (see figure 1). The bottom level contains actual design data for particular design cases. At this level, a model of Feature instances forms a network of design information. The Feature instances are defined by their corresponding Feature types, in the middle level. Here, Feature type libraries represent design knowledge in the form of formalised concepts. These libraries can be either standardised or managed by the individual designer. The flexibility and extensibility of the framework are manifested in the top layer, the so-called meta-layer, which defines the format for both the Feature types layer and the Feature instances layer. The meta-layer contains the structures and rules that determine the way Feature types are defined and the way Feature instances are modelled.

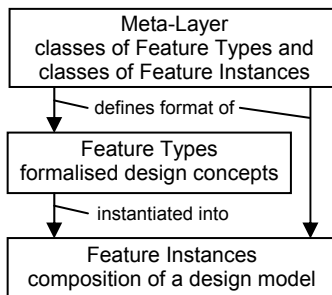


Figure 1. Framework for Design by Features

For the definition of Feature types and the description of Feature instances, a C like language has been developed. The FBM framework is implemented as an API (Application Programming Interface) providing a data management environment for the development of design system prototypes. The previous implementation of the API, which was based on an object oriented database, is now being redeveloped on the basis of XML-Schema technology.

4.1 XML-SCHEMA IN THE FBM FRAMEWORK

The designed functionality of the FBM framework and of XML-Schema have some important aspects in common. Both the FBM framework and XML-Schema provide a syntax to declare logical structures of data. The FBM framework provides a language to define libraries of Feature types; XML-Schema defines a syntax (actually an XML namespace) to declare data types and structures of elements to be used in XML documents. The FBM framework provides a language for the description of design model data in Feature instances that are created from Feature types; in the XML-Schema approach, data is provided in XML, with reference to the appropriate XML-Schema.

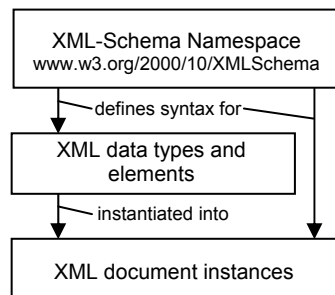


Figure 2. Similarity of the XML-Schema functionality with the FBM framework

Both the FBM framework and XML-Schema offer extensibility of the conceptual model, i.e. the set of typologies to be used in modelling. In the FBM framework this is done by allowing the user to build up Feature type libraries; in XML-Schema, the user can add data-types and element definitions to the schema.

The mechanism of namespaces makes it possible to add elements and attributes to a document's structure that are not declared in the XML-Schema. The schema can be declared to allow the addition of these external constructs to the document; this does not violate the rules of XML well-

formedness. Moreover, additional elements and attributes may be declared in other XML-Schemas that are referred by the document as well. This compares to the ability of the FBM framework to model relationships between Feature instance that are not defined by the Feature types. These are called instance-level relationships and are used to model ad-hoc properties of a particular design element that are not an considered to be a part of the underlying design concept.

Sharing of Feature type libraries through Internet requires unique identification of libraries and types. Namespaces in XML, when used with Uniform Resource Identifiers, provide a mechanism to do this in a human interpretable manner.

5. Information Modelling with XML-Schema

In the XML-Schema approach, Feature types are defined as elements in XML-Schema. Instantiation of Feature types into Feature instances is done in XML instance documents, by writing elements as defined in the XML-Schema. An incomplete extract of a small Feature model with instances is the XML instance document shown in table 1.

TABLE 1. XML instance document containing a Feature model.

XML instance document containing a Feature model	
1	<?xml version="1.0" encoding="UTF-8"?>
2	<!-- author: Design Systems group (TU/e) -->
3	<ftrbase:Model
4	xmlns:ftrbase="http://www.ds.arch.tue.nl/ftrbase"
5	xmlns:spatial="http://www.ds.arch.tue.nl/ftrlib/arch/spatial"
6	xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
7	name="Offices">
8	<spatial:Area name="Area1" value="29.7"/>
9	<spatial:Room name="Office1">
10	<ftrbase:Author>Joran</ftrbase:Author>
11	<ftrbase:Description>Room 4.14</ftrbase:Description>
12	<area ref="Area1"/>
13	<enclosedBy ref="WallB" index="1"/>
14	<enclosedBy ref="WallC" index="2"/>
15	<numberOfWorkplaces xmlns="" ftrbase:ref="NoDesksInOffice1"/>
16	</spatial:Room>
17	<spatial:Wall name="WallB">
18	<ftrbase:Author>Joran</ftrbase:Author>
19	<element ref="ElementB1" index="1"/>
20	<element ref="ElementB2" index="2"/>
21	</spatial:Wall>
22	<!-- ✕ -->
23	</ftrbase:Model>

In line 4 of this document, a namespace is declared that defines a number of elements and attributes that form a meta-level for the definition of Feature

types and instances, such as the Model element used on line 3. Line 5 declares a namespace for a library of Feature types representing spatial concepts, such as Area and Room.

The first Feature instance, Area1, is of the Feature type Area and has a value of 29.7. The used Feature types are explained below. The Office1 instance has a property with the name area that relates to the Area1 instance. Two other relationships are part of the property named enclosedBy and refer to WallB and WallC respectively. This property, as can be found below in the definition of the Feature type Room, has a cardinality of 0 or more.

The XML-Schema that provides the declarations of the Feature types used in the above model, is shown in table 2. In line 4 the namespace is declared that will be used by the schema in this document; in line 7 it is also made the default namespace for this document. The Area Feature type is declared starting from line 13. After some descriptive data, the content of this type is declared, on line 22 and further, as a simple type that is inherited from float, but allows only non-negative values. The default value is set to 0.

On line 43 starts the declaration of the Feature type Room, which is a subtype of the type Space, adding the enclosedBy property to its definition. This property represents the association of the room with its enclosing walls. It has a minimum occurrence of 0 walls and an unbounded maximum.

Feature types are declared as types in the XML-Schema. For the purpose of instantiation, elements are declared for each of the Feature types, as in lines 54-56.

TABLE 2. XML-Schema document declaring Feature types.

XML-Schema document declaring Feature types	
1	<?xml version="1.0" encoding="UTF-8"?>
2	<!-- author: Design Systems group (TU/e) -->
3	<xsd:schema
4	targetNamespace="http://www.ds.arch.tue.nl/ftrlib/arch/spatial"
5	xmlns:ftrbase="http://www.ds.arch.tue.nl/ftrbase"
6	xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
7	xmlns="http://www.ds.arch.tue.nl/ftrlib/arch/spatial"
8	elementFormDefault="unqualified"
9	attributeFormDefault="unqualified">
10	<xsd:import
11	namespace="http://www.ds.arch.tue.nl/ftrbase"
12	schemaLocation="FtrBase.xsd"/>
13	<xsd:complexType name="Area">
14	<xsd:annotation>
15	<xsd:documentation>
16	<ftrbase:Author>Joran Jessurun</ftrbase:Author>
17	This Feature type defines a spatial area.
18	</xsd:documentation>
19	</xsd:annotation>
20	<xsd:complexContent>
21	<xsd:extension base="ftrbase:FtrBase">
22	<xsd:attribute name="value" use="default" value="0">
23	<xsd:simpleType>

XML-Schema document declaring Feature types	
24	<xsd:restriction base="xsd:float">
25	<xsd:minInclusive value="0"/>
26	</xsd:restriction>
27	</xsd:simpleType>
28	</xsd:attribute>
29	</xsd:extension>
30	</xsd:complexContent>
31	</xsd:complexType>
32	<xsd:complexType name="Space">
33	<xsd:complexContent>
34	<xsd:extension base="ftrbase:FtrBase">
35	<xsd:sequence>
36	<xsd:element name="area" type="ftrbase:Role"
37	minOccurs="0" ftrbase:roletype="spec"
38	ftrbase:ftrtype="Area"/>
39	</xsd:sequence>
40	</xsd:extension>
41	</xsd:complexContent>
42	</xsd:complexType>
43	<xsd:complexType name="Room">
44	<xsd:complexContent>
45	<xsd:extension base="Space">
46	<xsd:sequence>
47	<xsd:element name="enclosedBy" type="ftrbase:Role"
48	minOccurs="0" maxOccurs="unbounded"
49	ftrbase:roletype="assoc" ftrbase:ftrtype="Wall"/>
50	</xsd:sequence>
51	</xsd:extension>
52	</xsd:complexContent>
53	</xsd:complexType>
54	<xsd:element name="Area" type="Area"/>
55	<xsd:element name="Space" type="Space"/>
56	<xsd:element name="Room" type="Room"/>
57	<!-- ✕ -->
58	</xsd:schema>

6. Implementation in the VR-DIS programme

The introduction of XML-Schema as the basis for Feature management in the VR-DIS system, has led to an updated architecture for this system. The central module of the system is implemented using Microsoft's Common Object Model (COM). Using XML technology, the COM interface provides access to the Document Object Model (DOM). The Document Object Model [W3C-DOM 2001] is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of documents. The Feature Management Core implements the Feature Management Object Model that can be compared to the DOM and is also defined as platform- and language-neutral interface (see figure 3).

On top of the DOM, a Feature Management Schema is declared that imposes additional constraints on elements and attributes in XML-Schema that are required for proper declaration and instantiation of Feature types.

Application level modules can access the central Feature Management Module in two manners. The first and easiest is by using the Feature Management Object Model provided by the Feature Management Core. The second type of access is directly to the Document Object Model through the Feature Management Schema. This second way of accessing the model is more direct, but requires that the additional constraints in the Feature Management Schema are observed by the application. In the Feature Management Core, this is taken care of for the application.

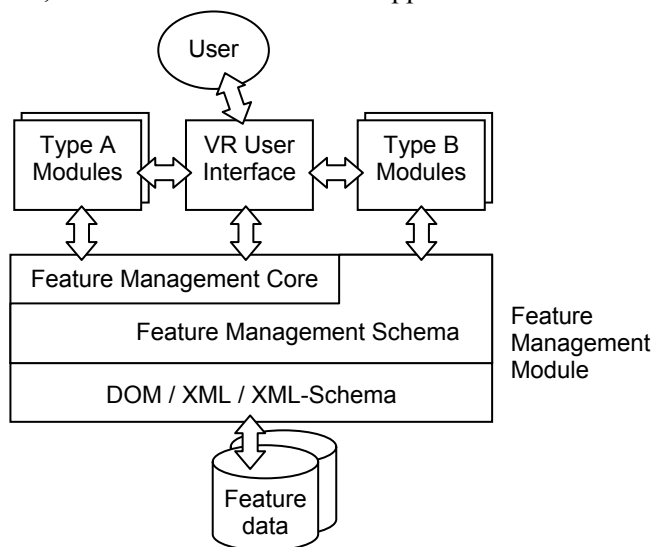


Figure 3. VR-DIS: XML-Schema based system implementation.

7. Conclusions

The similarities of the approach developed in the FBM framework and the activities in the XML community, especially the XML-Schema activity, can be interpreted as a confirmation of the directions taken in this research project. The wish to provide users with a flexible technology that allows them to express semantically rich typologies, as recognised in the work of the authors, appears to form the basis for the XML developments as well.

Using the XML-Schema technology in the FBM framework solves a number of issues that were of concern in the VR-DIS programme. (1) The open XML standard forms a better and more independent interface to data storage of the information system. (2) Namespaces, when used with URI's with the HTTP protocol in XML-Schema, provide the possibility to retrieve Feature type libraries through Internet. (3) The development of the FBM framework is now based on common technology, which enhances the flexibility and speed of development. (4) Feature models, now stored in the

form of XML instance documents, can be used also by applications that are not part of the VR-DIS programme, as long as they are XML-enabled. For example cost-calculation systems can access the models easily through either DOM or other ways of reading XML instance documents. (5) In terms of dissemination of research, the usage of a common technology to define the meta-level of the FBM framework (i.e. the language used to define types and instances), is much preferred over the previous approach in which yet another syntax was developed for this purpose.

References

- Achten, H.H. and J.P. van Leeuwen: 2001, Scheming and Plotting Your Way into Architectural Complexity, in *These Proceedings of AVOCAAD 3rd International Conference 2001*.
- Achten H.H., B. de Vries, and J.P. van Leeuwen: 2000, Computational Design Research: The VR-DIS Research Programme, in H.H. Achten, B. de Vries, and J. Hennessey (eds.), *Design Research in the Netherlands 2000*, Eindhoven University of Technology.
- Coomans, M.K.D. and J.P. van Leeuwen: 2001, Abstract but Tangible, Complex but Manageable, in *These Proceedings of AVOCAAD 3rd International Conference 2001*.
- Eastman, C.M., T.S. Jeng, H.H. Assal, M.S. Cho, and S.C. Chase: 1995, *EDM-2 Reference Manual*, Los Angeles: University of California in Los Angeles.
- Ekholm, A. and S. Fridqvist: 1998, A Dynamic Information System for Design Applied to the Construction Context, in *The Life-Cycle of Construction IT*, proceedings of the CIB W78 workshop, Stockholm: Royal Institute of Technology.
- Fridqvist, S.: 2000, *Property-Oriented Information Systems for Design*, PhD thesis, Lund Institute of Technology.
- Hendricx, A.: 2000, *A Core Object Model for Architectural Design*, PhD thesis, Katholieke Universiteit Leuven.
- van Leeuwen, J.P.: 1999, *Modelling Architectural Design Information by Features, an approach to dynamic product modelling for application in architectural design*, PhD thesis, Eindhoven University of Technology (<http://www.ds.arch.tue.nl/jos/thesis>).
- van Leeuwen, J.P. and B. de Vries: 2000, Modelling with Features and the formalisation of early design knowledge, in Gonçalves et al. (eds.), *Product and Process Modelling in Building and Construction*, Proceedings ECPPM2000 Lisbon, Rotterdam: Balkema.
- W3C-XML: 1998, *Extensible Markup Language (XML) 1.0*, World Wide Web Consortium Recommendation, <http://www.w3.org/TR/REC-xml/>
- W3C-Namespaces: 1999, *Namespaces in XML*, World Wide Web Consortium Recommendation, <http://www.w3.org/TR/REC-xml-names/>
- W3C-XML-Schema: 2000, *XML Schema*, Part 0: Primer, World Wide Web Consortium Candidate Recommendation, <http://www.w3.org/TR/xmlschema-0/> (see also Part 1: Structures and Part 2: Datatypes).
- W3C-DOM: 2001, *Document Object Model (DOM)*, World Wide Web Consortium Recommendations and working drafts, <http://www.w3.org/DOM/>