OBJECT-ORIENTED MODELLING USING XML IN COMPUTER-AIDED ARCHITECTURAL AND EDUCATIONAL CAD

The Problem of Interoperability exemplified in two Case Studies

T. FISCHER, M. BURRY*, ROBERT WOODBURY**
Fachbereich Erziehungs- und Humanwissenschaften
University of Kassel (GhK), Germany
**School of Architecture and Building*
Deakin University, Australia
tfischer@hrz.uni-kassel.de, mburry@deakin.edu.au
***School of Architecture, Landscape Architecture and Urban Design*
University of Adelaide, Australia
rw@arch.adelaide.edu.au

Abstract. This paper highlights our application of XML as a messaging and storage format for parametric 3D modelling and pattern-oriented online teaching. As a recent format for data description and transport technology XML is designed to allow communication between arbitrary data platforms - and to communicate purpose-insensitively. We have used it to communicate design patterns as well as design parameters and as a consequence experienced a remarkable technical similarity between both approaches with their common manifestation in object orientation. There is a necessity to perform dynamic synchronizations of semantics between 'knowledge domains' involved in design processes in order to provide the necessary conceptual openness. At this time, this requirement appears to be alien to available XML schema specifications and tools.

1. Introduction

Data modelling in computer-aided design contexts requires re-modelling of real world sections to the same extent that its product relates to problems of the natural world (this is typically the case). In architecture an example is part of a building or a construction process; in teaching a class of students and the design of the learning progress. As neither the design processes themselves nor the applications of their outcomes happen in isolation, we experience a significant demand for interoperable data formats in order to exchange digitally modelled reality and to run trivial subtasks automatically. In the field of architecture as well as in the field of education, XML specification initiatives have developed data formats which are intended to cater for this purpose (aecXML and IMS XML metadata schemas). Working on two research projects, one related to architectural 3D-modelling and one to architectural online education, we have had to face the reality of real-time cross-domain data modelling with the conclusion that the XML specifications mentioned above suffer from a conceptual deficit. In this paper we discuss the progress made with both projects from a technical standpoint and report on the findings we gained regarding data modelling strategies.

2. The *Paramorph*

Intended as a rejection and an alternative to accident-oriented computer-based design methodologies, the paramorph is a numerical/geometrical engine connected to a 3D-Renderer which allows the controlled generation of highly variant 3D geometry through so-called ‘parametric design’, or ‘associated geometry’. At the beginning of a design process, the paramorph in its initial state is a description of a potential geometry without real manifestation, or a ‘pattern’. To produce such a manifestation the designer has to provide a finite number of parameters that are applied to the pattern. The mathematical description of the associative geometry is sufficient to assign defined attributes to any element of the model. Basic geometrical properties are initially defined constants; morphological variations are the results of controlled manipulations to the parameter values.. In contrast to the freedom the designer experiences sculpting a ‘blob’ in a click-and-drag 3D modeller, the paramorph allows a structured design process that is easily repeated, that is, the results of experimentation are reproducible. Parametric design, as with other design methods, aims to limit the problems that come along with totally free design situations – but potentially at the cost of their freedom. Although this cost does not remain unchallenged (Miller: 1990) parametric design nevertheless offers considerable benefits to the architectural user: the initial pattern itself can be regarded as a variable, as a replaceable pattern or object(Burry: 1996). Pattern descriptions can be communicated between designers and platforms. They might be developed by expert designers and used by non-experts with the expertise captured within their description becoming reusable expert-knowledge. The distinction between geometry designer and educational-tool designer becomes blurred. Moreover, parametric design provides a means to design buildable virtual architecture, as it were: a central problem of building hypersurfaced architecture is the transfer of Cartesian information from drawing or model-space into natural space. Some geometries, however, are more likely to be transferable than others such as ruled surfaces. As the principle topology of a paramorph pattern is its persistent property and not affected by given parameters, any of its possible manifestations are equally likely to be transferable into natural space within the conventions of gravity recognition and successful structural design. Setting up a ruled surface paramorph leads to a ruled-surface manifestation with any given set of parameters and the transferability of any of those manifestations will benefit from the characteristics of its underlying principles. The paramorph is driven by a set-up of two engines: a logic/mathematical processor and a 3D modelling/rendering package. To allow interactive, visually graspable and yet flexible parameter manipulation, the logic/mathematical engine is implemented through mathematical manipulation via a typical proprietary spreadsheet. The parametric design modelling package used is Parametric Technology's CADDS5 with the possibility of two-way communication between the spreadsheet and the 3D modeller using instruction sets mapped as comma separated parameter tables. Controlling the interfaces of both communicating applications and specifying problem-oriented semantics was a straightforward approach. Following the relational model, parameters in transported table fields make sense according to their position relatively to other fields and table edges.

The visualizations in the 3D GUI, its ray-traced renderings and the original visual spreadsheet table equal each other insofar as they are all interpretations of the same set of data. The direction of the inter-domain communication (shown in Figure 1) implies a one-way numeric-to-geometric design intent but it can also be inverted or reciprocal – depending on the port between the pattern (the constraint and logic model) and the CADDS5 domain. To allow a flexible number of model attributes to be handled parametrically, the input domain would need the ability to dynamically extend the data exchange format. As the resulting 'semantics' would be incompatible with the target domain's interface, associated identifiers are required to express the sense a parameter should make. An extendable set of identifiers for values can be specified in XML but nonetheless can only transport data and not the associated semantics: in other words a human has to synchronise the way data has to be interpreted unless possible geometry has sufficiently been captured in a common XML schema at the outset. The nature of design tasks is likely to require more objects and attributes than contained by this presupposed sufficiency. For this reason, our interest in the aecXML schema is accompanied by an essential curiosity and some scepticism

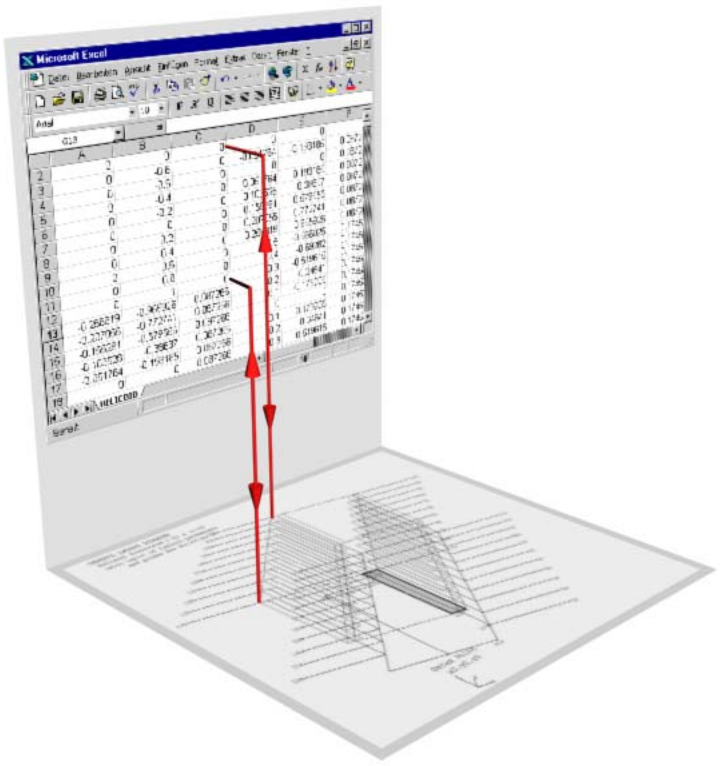
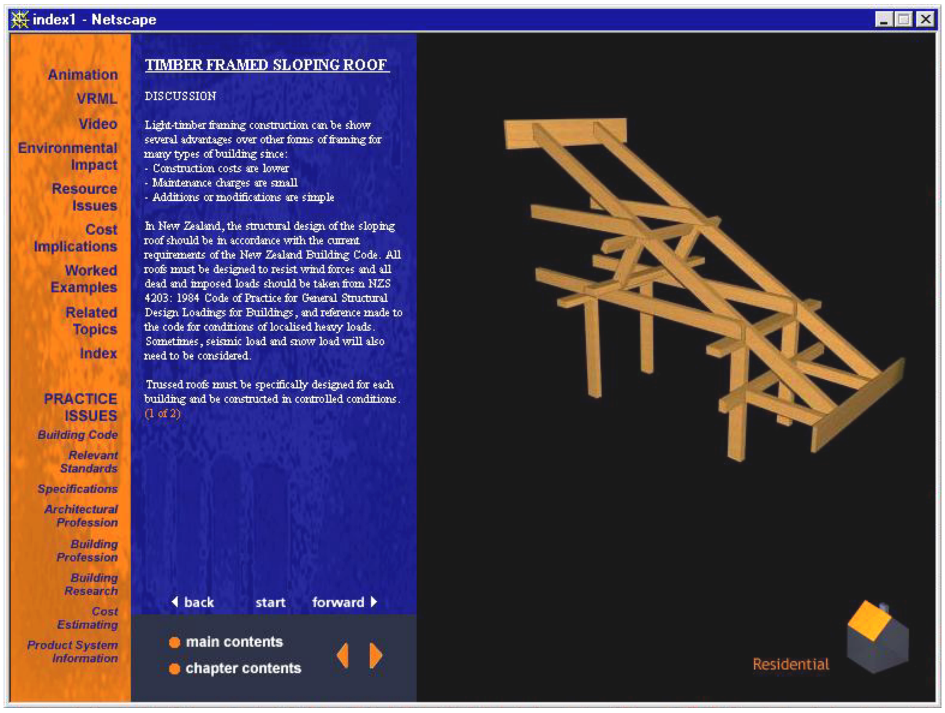


Figure 1. The Paramorph: Data transfer from logical/mathematical engine to 3D modeller.

3. The Construction Primer

The *Construction Primer* is a learning application on generic building construction as it is applied in Australia and New Zealand. Its content is the outcome of students' work and it was compiled into a hardcopy and subsequently an electronic version. The latter was a static HTML application produced using Microsoft Frontpage now migrated to an XML viewer interface. It is available online and as a stand-alone CD-ROM. In order to enhance the degree of its interactivity and individual content-delivery we are porting its content to a database. Our first step was to port the project from HTML to XML; according to the important role distant education traditionally plays in Australia, the comprehensive scope of the Construction Primer and its attendant need to be up-to-date with all the relevant technological standards makes it an ideal vehicle for an examination of large-scale online teaching strategies.



TIMBER FRAMED SLOPING ROOF

DISCUSSION

Light-timber framing construction can be show several advantages over other forms of framing for many types of building since:
- Construction costs are lower
- Maintenance charges are small
- Additions or modifications are simple

In New Zealand, the structural design of the sloping roof should be in accordance with the current requirements of the New Zealand Building Code. All roofs must be designed to resist wind forces and all dead and imposed loads should be taken from NZS 4203: 1984 Code of Practice for General Structural Design Loadings for Buildings, and reference made to the code for conditions of localised heavy loads. Sometimes, seismic load and snow load will also need to be considered.

Trussed roofs must be specifically designed for each building and be constructed in controlled conditions.
(1 of 2)

Figure 2. The Construction Primer: Screen shot of the HTML compilation.

Half a century after the computer's advent started a euphoric optimism towards its didactic potential, the majority of today's learning applications remain static, non-interoperable, require significant amounts of hands-on maintenance and are predicated to behaviouristic interaction levels. Already the first vision that the historians (Niemiec and Walberg: 1989) held in this field emphasized the importance of extensive integration of all domains involved in didactic interaction such as material repositories and -suppliers, methodology libraries, accounting systems and access control, teacher profiles, student administration and marking databases (see Ramo: 1957). We believe that the crucial reason for the contemporary deficits in this regard as the 'openness' and unpredictability which qualify teaching tasks as design problems (see also: Rittel and Webber: 1973). Teachers, for example, as with architectural designers, do not tolerate limitations caused by badly designed tools. Teaching process variables, such as the number of course milestones, sub marks and applied didactic methods are hardly predictable and are not easily handled by predefined database schemas. The availability of object-oriented databases potentially promises changes but on the integration and machine communication side we need data exchange protocols that are equally extendable.

While EDUCAUSE's IMS XML schema documentation refers only to learning material units as didactic objects we perceive the necessity to regard any entity involved in didactic interaction as a didactic object which should be re-modelled in the software in charge of handling (un)expected interactions and transactions. Trivial administrative tasks may then be automated and executed by the computer. Abstract descriptions of (learning) contexts are the key to inter-domain data exchange and resource sharing. It is essential to integrate different knowledge domains in design processes like didactic planning as for example, it effects different levels of institution-administrative trees: the selection of a learning material presentation mode usually occurs on the individual teacher's level while typically the institutional organization level is in charge of access control and accounting. Figure 3 shows a model of material delivery within an integrated environment. In a context like this, teaching software 'knows' which teacher in which course delivers which material in which presentation mode to which learner at a given time.

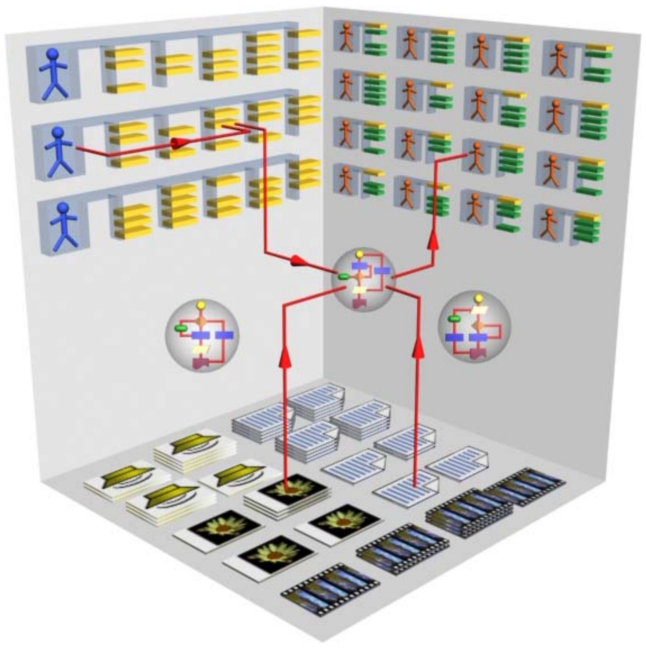


Figure 3. The Construction Primer: Re-modelling of individualized learning material delivery.

The dynamic content processor in the center of Figure 3 is a so-called SeL masterfile: a context-sensitive dynamic behaviour filter written in the *SeSAMe Language* which was developed for the WWW-Authoring environment SeSAMe at the University of Kassel (Fischer: 1999). The SeSAMe system is based on the separation of data and behaviour. Hence, in contrast to XML, SeL is not only able to model data, but also model behaviour. The underlying understanding of content elements as *parameters* for documents allows very flexible data handling: the addition and dynamic selection of master-files add new methods to existing document objects which can then behave in formerly impossible ways. Using the resulting design freedom, the University of Kassel was able to implement a method which optionally generates WWW documents optimised for blind and visually disabled from its existing WWW document tree, ten months after SeSAMe was implemented.

4. Discussion

As a team working on both projects, we initially had very different design paradigms in mind and finally perceived a common nature behind them: The paramorph is a morphological experiment aiming at the controlled authoring of geometric shape variance driven by any given numerical input set. In this project our design work followed our understanding of parametric design. The Construction Primer is a learning environment with which we examine the possibilities of individually compiled real-time delivery of didactic material in contexts ranging from distance learning to essential design decision support in the professional office to on-site application tuition (how to install the window, for example, *in acto*. We regard document elements and interaction participants as didactic objects and approved system behaviours as patterns. Our model for this is Pattern Design which - poor as it might appear from many points of view (see Protzen: 1977) – nevertheless offers practical ways to identify, describe, catalogue, communicate and therefore reuse expert knowledge. After Alexander published 'A Pattern Language' (Alexander: 1977) his approach had a significant impact on object-oriented programming (Gamma et al.: 1995) and finally inspired computer science teachers to develop the *pattern teaching'* approach (Lilly: 1996). In pattern teaching, didactic methods are methods in an object oriented sense while material elements and learner attributes can be understood as parameters. To apply this model where it initially came from – in software – we need inter-domain messaging beyond predefined data sets. The common major objective of XML initiatives like aecXML and IMS XML is to attain interoperability – the ability of software to share and exchange data in useful ways using defined standards in order to model for example business, design or teaching transactions. XML schemas are therefore not primarily intended to provide containers for data storage but to enable data transport between domains. Domains in our sense are models of 'problem spaces' : real-world sections within which human understanding or software is sufficiently used to model that particular part of the world in order to (inter)act meaningful. But objects of the real world and descriptions of them do not only occur domain-internally: they are also communicated between domains. Examples are any kinds of social interaction models such as collaborative design, business negotiation or teaching. For this kind of data exchange we regard XML as a very productive, useful and an easy-to-learn approach but our application of XML has revealed a number of problems. The following are our principal observations:

The straightforwardness and compatibility of XML with its forerunners SGML and HTML that can be obtained through the rhetoric in much of the XML documentation needs to be viewed with a little more realism. Texts from the XML community somewhat rashly suggest that simply the availability of XML will provide immediate interoperability for all future. XML is a different description language and data conversion costs effort, money and time. Converted data still needs to be maintained. Moreover, there is no backward compatibility to older data processing- or browser software. Having a converted set of data does not necessarily mean it is immediately applicable. At this stage we have created a semantic system within our domain that other domains may not understand.

Different domains – even when they operate at an equal technological level – occasionally also use different semantics. But as XML itself is purely syntactical with no means for semantic definitions, data might easily be misinterpreted. In a educational marking context the value '6' for a test result might have opposite meanings in the German and the Swiss educational systems. The same applies not only to data elements but also to data descriptors. If semantics differ between domains, they lose their ability to process data in intended, unambiguous ways. Syntactical declarations in document type definitions and individual attribute definitions are required to avoid (trivial) confusions like the mix-up of marking semantics mentioned above. They provide common interpretation frames in the source and in the target domain. The visual and syntactical interpretation frames that XML uses to compensate modelling differences between domains are style sheets and document type definitions which are sent along with the actual data. In describing its purpose the aecXML documentation explains: "It includes an XML schema to describe information specific to the information exchanges between participants involved in designing, constructing and operating buildings, plants, infrastructure and facilities" This ambition ignores the fact that information as such on an advanced level is utterly alien to computers and networks: computers store, process and exchange data which might become information when it is interpreted in the context of a knowledge domain (which, when it comes to non-trivial natural world problems in architectural or educational design, our computers can hardly achieve).

Semantic (or pragmatic) differences between domains may not only occur between different domains at one time - they also potentially emerge within one domain along the time axis. What makes sense today might not make sense tomorrow and vice versa. A popular example of a failure in this respect is the specification history of HTML. Though each version of DTD there was (apart from browser-specific differences) a constant, hard-coded in everyone's browser and the understanding of how documents should be described turned out to be utterly different in the views of the W3 consortium which is responsible for its specifications and the millions of document publishers. Consequently, from version 1.0 in 1990 to version 4.0 in 1998 HTML developed step by step from its focus on logic document description to a focus on desktop publishing issues. Metadata – a protocol closely related to XML – shows comparable shortcomings: different domains may name things in different ways and they may understand contents in different ways. When XML metadata declares a document's type as a "letter" why should it not be a "poem" as well? Semantic inter-domain problems might also be caused by intention: document keywords, for example, might be intended as an index for content-related searches which is more efficient that full-text searching. But (as happens) authors use them as a means to attract attention to the wrong information.

Object orientation on the programming language level typically implies the concept of inheritance: the potential of objects to copy data or behaviour of themselves to child objects. XML does not support this strategy and models of hierarchically structured sections of the real world have to include a significant amount of redundancy at the data level. At the behaviour level there are comparable shortcomings: as no dynamic actions can be modelled, XML requires the explicit transportation of basic and redundant derived data. If, as in the example of a student marks record, the handling of a (derived) average mark is desired, it has to be expressed explicitly though it is logically contained in the basic marks, unless there is a way to communicate and execute the respective algorithm.

We see a major problem in the nature of the application areas of aecXML and the IMS XML metadata schema. They are intended to describe data in design contexts probably based on little understanding about the *nature of design*, or how designers design. Design tasks are "wicked" (see Webber and Rittel: 1973) insofar as they always have a unique nature, they consist of an unpredictable number of elements with an unpredictable number of attributes that have to be considered by design process participants. Experiences are not trivially applicable to new problems. The reason for this wickedness is the deeply social nature of design tasks as they occur in architectural and teaching design – fields within which processes are hardly repeatable. Design problem spaces are domains of an unknown and unforeseeable nature. But interoperable standards require the opposite: experiences made in well-known problem spaces upon which a finite number of data elements and attributes can be collected and become specified as a problem-oriented communications protocol. Problems with finite numbers of elements and attributes are tame ones as they occur where rational solution steps and strategies are available as in engineering contexts. To handle a wicked problem it is very likely that a data exchange protocol needs problem-oriented extensions automatically. XML, in principle, does support those extensions. In this regard aecXML and IMS XML metadata appear hardly to be future-proof: interoperable standards require specifications on two levels. Firstly, the actual data format (as they are available in aecXML and IMS XML schemas), and secondly, a protocol to negotiate specification changes and to update participating domains about those changes in real time. We consider this to be essential in wicked problem contexts. To fulfil this requirement, XML environments need to support elaborate querying facilities and appropriate inter-domain co-ordination strategies on a basic level as well as agents which detect the requirement for schema extensions on an advanced level. At this time, we are not aware of any support for these requirements.

5. Conclusion

We regard the growing public interest in XML as an reinforcement of our perception that interoperable standards become broadly recognised as an essential key to inter-domain communication enabling computers to (inter)act meaningfully beyond the boundaries of locally implemented software. But XML schemas in general, and aecXML and IMS metadata in particular, appear to be neither substantially sufficient nor final answers to given design problems. They cannot compete with data- and constraint modelling, exchange strategies and access control as found in database management systems. In comparison XML is easy to learn and applications are easily implemented but when it comes to mission critical large-scale projects, the investment in DBMS-handled storage and transaction appears to be the better choice. XML has nevertheless proven to be a very productive rapid-prototyping language in our research and teaching enabling us to implement experimental set-ups and to handle interim data conversion products in a safe way. It can handle design data at one point in time. XML application in design contexts along the time axis appears to be a risky business. The speed of design processes does not spare the XML development. At this point in time, XML strategies and tools is surely ‘work in progress’ for us all. This – in its worst sense – means that early concepts might turn out as dead end developments. Last year the aecXML documentation stated ”a 90% solution today is better than a 99% solution next year”. As matters stand, current aecXML and the XML IMS metadata schema are solutions for which design problems might still be too wicked.

References

Burry, Mark, Prentice, Reg and Wood, Peter: 1995*, Walking before running: A prelude to multimedia construction information*, eCAADe XIII conference proceedings, University of Palermo, pp. 257-266

Burry, Mark: Summer 1996, *Parametric Design and the Sagrada Família*, Architectural Research Quarterly Number 4: Volume 1, London pp 70-81

Fischer, Thomas: 1999, *Dynamic Style Processing with SeSAMe*, EUNIS'99 conference proceedings, Helsinki University of Technology, pp. 67-72

Gamma, Erich et al.: 1995, *Design Patterns. Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading

IAI aecXML Domain Committee: The aecXML Homepage, http://www.aecxml.org/index2.htm

IMS Global Learning Consortium: The IMS Homepage: http://www.imsproject.org

Lilly, Susan: 1996, Patterns for Pedagogy, *Object Magazine* 1/1996, pp. 95-96

Matthias, Andreas: 2000, *SeL: Benutzerhandbuch, Programmierhandbuch, Referenz*, http://www.uni-kassel.de/design2/Handbook/DE/contents.html

Miller, F.C.: 1990, *Form Processing Workshop: Architectural Design and Solid Modelling at MIT*. In (eds) McCullough, M. Mitchell, W.J. and Purcell, P., *The Electronic Design Studio*, The MIT Press, Cambridge Massachusetts, pp. 441-455

Mitchell, William J.: 1994, Three paradigms for computer-aided design, *Automation in Construction* 3, pp. 239-245

Maruyama, Hiroshi et al.: *XML and Java: Developing Web Applications*, Addison Wesley, Reading, Massachusetts

Niemiec, Richard P. and Walberg, Herbert J.: 1989, From Teaching Machines to Microcomputers: some Milestones in the History of Computer-Based Instruction, *Journal of Research on Computing in Education*, no. 21 spring 1989, pp. 263-276

Protzen, Jean P.: 1977, The poverty of the Pattern Language, *Design Methods and Theories*, vol. 12, no. 3/4, pp. 191-194

Ramo, Simon: 1957, A new Technique of Education, *Engineering and Science*, vol. XXI, October 1957, pp. 17-22

Rittel, W.J. Horst and Webber, Melvin M.: 1973, Dilemmas in a General Theory of Planning, *DMG-DRS Journal*, vol. 8, no. 1, pp. 31-39, reprinted from *Policy Sciences*, vol. 4, no. 2

Schmitt, Gerhard: 1993, Architecture et Machina: Computer aided Architectural Design und Virtuelle Architektur, Verlag Friedrich Vierweg, Braunschweig and Wiesbaden