On the road to standardization

Rudi Stouffs and Ramesh Krishnamurti[1]
Delft University of Technology
[1]Carnegie Mellon University

Key words: Information exchange, Standardization, Representations

Abstract: This paper offers an analysis of current standardization efforts, including a classification of their approaches and an evaluation of their advantages and disadvantages with respect to different contexts. In focusing on the design context, a *syntactic* approach to standardization is recommended, and exemplified with a concept for representational flexibility termed *sorts*.

1. INTRODUCTION

Effective digital representations for design have been a topic of research since Sutherland's Sketchpad (Sutherland, 1963) marked the beginning of CAD research. Early efforts into purely geometric representations led to the establishment of geometric modeling as a research field, presenting us, amongst others, with polygon-based and NURBS-based three-dimensional representations that currently form the basis of most modeling applications. More recently, product modeling research has taken a much wider view of design representations, considering geometric design as only one aspect in the product design process and focusing on design as a collaborative process between a variety of actors and experts from many different design disciplines. These different disciplines are concerned with different aspects of the final product and require different representations to work with. Furthermore, different actors adopt different design techniques and methodologies, demanding alternative design representations for the same product aspect. Integrating these different design views into a single product model, or supporting information exchange between alternative representations, possibly in coordination with a central product model, is far

from straightforward, as current research into product models, such as ISO STEP (ISO, 1994), illustrates.

In architectural and building design, this problem is even more prominent, as design methodologies are varied and diverse, the actors in a collaborative building project are numerous and from a large body of disciplines, and not least, both the project and team are potentially unique from project to project. Furthermore, the building industry is fragmented and characterized by a large number of small- and medium-sized companies, making it even harder to impose common models or processes for information exchange. As a result, information exchange in the building industry has long been, and still is, dominated by a data exchange format, DXF, that is mostly concerned with geometric information and which was designed for use with a single commercial application, AutoCAD™. At the same time, many efforts exist and have existed to conceive a common product model for building design, for example, within the STEP developments, by the International Alliance for Interoperability (Bazjanac, 1998), and more recently in XML (Tolman and Böhms, 2000, aecXML, 1999). Despite the many efforts, little real progress has been made, both in agreeing on common models for various building design aspects, and in convincing the building industry to adopt such models on a general level.

This paper offers an analysis of current standardization efforts, including a classification of their approaches and an evaluation of their advantages and disadvantages with respect to different contexts. In focusing on the design context, a *syntactic* approach (e.g., O'Brien, 2000, Stouffs and Krishnamurti, 1997) to standardization is recommended, and exemplified with a concept for representational flexibility termed *sorts*.

1.1 Data exchange

Between and within disciplines, building partners use a variety of different applications and tools based on many distinct data formats. This diversity of data formats makes supporting information and data sharing within a building project a complex and difficult task. Various approaches to facilitate data exchange among partners exist, based on a number of different techniques and technologies. The most obvious approach is to develop a specific utility for translating data between two given formats. Despite attempts at developing alternative approaches, this is still the most widely used. The advantages are clear: the single purpose supports a focused development towards a highly effective and efficient tool that emphasizes the nature of either or both formats or the specifics of the context in which the utility will perform its task. Such a utility may be used stand-alone or integrated into an application that uses either data format, e.g., in the form of

an import or export functionality, or into a system that offers multiple translation facilities.

Most often, such specific utilities serve data sharing in conjunction with a standard or pseudo-standard. Consider DXF, a data exchange format developed for the purpose of AutoCAD™'s own translation needs between subsequent versions of the software. This format was adopted by the CAD software industry as a pseudo-standard for CAD data exchange. By integrating import and export functionalities from and to DXF into every CAD application, the format serves as a standard for data sharing among CAD applications. The most obvious advantage of such a standard is the fact that each application needs to support translation only between a single pair of data formats, that is, from the proprietary format to the standard and back. However, pseudo-standards are ill suited to this task. As these were never developed for this task, they neither reflect the nature of the proprietary format nor the context of the exchange. For example, DXF supports neither NURBS, a popular geometric representation for curved surfaces, nor textures, making it ill suited for sharing advanced 3D modeling data.

General standards for exchanging building data may overcome these limitations. However, standards are difficult to develop, as these require a broad consensus among industry members. Particularly in the building industry, such consensus is hard to achieve. Many reasons can be thought of. Most commonly, the fragmented nature of the building industry and the uniqueness of each building project (Buckley, Zarli, et al., 1998) are mentioned as primary reasons for this failure to achieve a standard for data sharing among project partners. Equally, neither has hope yet faded. New approaches based on advances in software technology have resulted in renewed and increased efforts and in better chances of achieving such a standard. Object technologies (e.g., Bazjanac, 1998, van Nederveen, 2000) and XML (e.g., Tolman and Böhms, 2000, aecXML, 1999) have served as the catalysts for these activities.

2. STANDARDIZATION APPROACHES

2.1 A-priori versus a-posteriori

Different approaches can be distinguished in standardization efforts. Generally, these adopt an *a-priori* approach: an attempt is made, before or at the onset of the project, at establishing an agreement on the concepts and their relationships which offer a complete and uniform description of the project data. If this collection of concepts and relationships is conceived of independent of the project specifics, i.e., its context, then the approach can

be additionally denoted as *top-down*. The STEP effort is a prime example of a *top-down*, *a-priori* approach, offering a methodology for developing product models and for the exchange of these product models, including its application to various industry domains.

The alternative is a *bottom-up* approach, where the project participants attempt to establish such a conceptualization from practice, based on the project specifics, at the onset of the project. *Object trees* (van Nederveen, 2000) are an example of an *a-priori*, *bottom-up* approach. Primarily aimed at the construction planning phase, object trees serve to improve electronic communication between participants of different disciplines in large-scale construction projects by offering them a methodology for developing representational object trees corresponding to concept hierarchies of construction aspects and elements, and their attributes. The methodology requires all participants to concur on the concepts and attributes involved; in return, it presents them with a unified framework for relating activities and for data exchange among participants. It is specifically suited for the construction and construction planning phases of large-scale projects in which the advantages of the conceptual and representational framework far outweigh the disadvantages of the need for an *a-priori* consensus.

Focusing on the design phase, it is debatable whether an *a-priori* approach, even if successful in the future, will support the variety and flexibility it intends to enable. Conceivably, it may further restrict creativity and individuality by imposing a common product model that caters only to an *a-priori* defined collection of views. For one, new design and analysis techniques or methodologies may be conceived and developed requiring new and different design representations that lie outside of the scope of the product model. Secondly, diversity in design approaches within the same discipline may not be, as a whole, supported by the same model. Lastly, even within the same design process, a single actor may choose to adopt different design representations for different purposes at various stages of this process.

Thus, there is a need to offer both flexibility in representations that allows a designer to adapt a representation to her intentions and needs, and representational dynamism that enables representations to be reconfigured throughout the design process in order to reflect the task at hand. This calls for an *a-posteriori* approach where users are empowered to define their own representations within their design activities and are provided with the tools to, subsequently, communicate the corresponding data into alternative representations as adopted by the other project participants.

2.2 Semantic versus syntactic

In the process of establishing a common product model, two steps – *semantic* and *syntactic* – can be considered. The former refers to the conceptual development, the latter to the translation of this conceptualization into a representational structure for practical use. Standardization efforts tend to focus on one of these steps. For example, the LexiCon effort was primarily concerned with a formal vocabulary for the storage and exchange of information in the construction industry, although representational development is now under way (Woestenenk, 1998, 2000).

S*emantic* development in any standardization effort may serve as the starting point for different *syntactic* developments. For instance, the STEP effort includes the specification of representational structures; other standardization efforts consider adopting the STEP semantics, or even part of the technology, in order to offer alternative syntactical expressions. An international consortium has been founded within the building industry that aims to define an object-oriented data model as a basis for project information sharing in the industry. These efforts of the named International Alliance for Interoperability (IAI) have resulted in a specification of Industry Foundation Classes (IFCs) defining a building object model shared by all IFC-compliant applications (Bazjanac, 1998). The IFC product model was developed using the EXPRESS modeling language developed in the STEP effort, and shares many concepts with the STEP product model developed for the building industry. Recently, the IAI also supports the aecXML Working Group, which started working on an extension of XML, a universal format for structured documents and data for the Web, in order to facilitate data exchange in the building industry (aecXML, 1999). The E-Construct project is also concerned with the development of an XML extension, named bcXML, to support e-commerce in the building industry (Tolman and Böhms, 2000). It builds upon the LexiCon semantics defined for projects and adopts the LexiCon tool for the purpose of a user interface to the data structures.

XML can be considered as an alternative to object technology in order to develop representational structures corresponding to conceptual product models. However, XML is more than a technology; it 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. When project partners can agree on the tags, they can exchange data described in any markup language based on these tags, even if their own markup language differs in scope or composition. As such, XML may be considered as a *syntactic* standard (O'Brien, 2000). XML can also be considered as an alternative

modeling language to the EXPRESS technology of STEP. XML has the advantage that it is readable both by humans and by the computer. Markup languages based on XML can easily be adapted or extended to one's own specific purposes or needs. Thus, XML structures can easily be defined corresponding to a conceptual product model and such structures can be compared between different models. In fact, it may be quite ironic to consider product model standards as structures fixed in XML (O'Brien, 2000).

3. REPRESENTATIONAL FLEXIBILITY

While these standardization efforts have more or less the same aim, their strategies are quite different each with distinct advantages and disadvantages in supporting flexibility in data formats for varying purposes and needs. The necessity for the support of data exchange reflects a desire to use alternative design representations that enable a particular expression, analysis, or organization. Translation utilities can support data exchange between the standard and proprietary data formats. However, there is a limit to what can effectively be catered for in this fashion. Advances in methodologies, techniques, and technologies repeatedly require new representations of the same building component or building aspect. Standards, however, are necessarily based on current knowledge, uses, and needs. The difficulty in establishing a standard and having it adopted as a basis for data sharing among all or most software applications on the market almost inhibits any subsequent changes in order to update it to new requirements – unless, such flexibility is built into the technology.

The IFC effort attempts to overcome this difficulty by adopting an object-oriented approach and envisioning an evolving object model. Objects encompass both data and access to this data, possibly including operations on the data. Applications can use this model, or parts of it, to define the underlying representation, or incorporate a translation from and to this model into their functionality. When the model is altered through a modification or extension of the object functionality or the development of new object classes, a corresponding adaptation of the applications may not be necessary, unless one wants to make use of the additional functionality provided in the model. In this manner, a single model can respond to advances in knowledge and technology. At the same time, however, this model still depends on a consensus and, as such, will not be able to support the entire spectrum of alternative design representations that can suit particular users or specific situations in the building process. Furthermore, access to this model is only available to software developers and, as a result, a designer will, in most

cases, be restricted to those representations that are provided by the software applications on the market rather than be able to exploit the potential of a truly flexible standard.

The Lexicon model suggests an alternative approach. Though as part of a semantic model, it considers a semi-syntactic approach in which concepts are unambiguously defined by their constituent attributes (Woestenenk, 1998). These attributes then comprise the primitive concepts that define the semantic vocabulary of this model. Taking this descriptive approach one step further, the attributes themselves can be described syntactically, leading to a purely syntactic description of the concepts as compositions of primitive data types. Within a formal structure, these syntactical descriptions may be compared independently of their conceptual meanings, thus allowing for synonym concepts. XML offers such a formal framework. As such, XML allows for an *a-posteriori* and *syntactic* standardization approach, providing all participants with the ability to define or adopt their own data model in XML, and considering ways of translating these different models between one another at a later stage, using tools developed for this purpose.

3.1 A framework for representational flexibility

XML is particularly suited to structure otherwise unstructured information, such as textual data, and to organize information available over the Web. However, it does not provide any information on how to manipulate the data and, as such, is ill suited to represent detailed graphical or geometrical data. Instead, a framework for supporting representational flexibility may be conceived of by borrowing from the different approaches in order to combine their respective advantages. From XML, it may inherit a foundation consisting of an extensible vocabulary of data components that can be composed hierarchically into a representational language. From the IFC effort, it may borrow the object-oriented approach, defining the data components as objects that encapsulate both the data structure and the operations defined on these structures. The symbiosis of these two approaches requires that the compositional operators be defined so that any compositional structure offers the same functionality as each component object separately. Hereto, a behavior can be defined for every component and structure as a collection of common operations on these structures for creation or deletion, or the merging of structures under some formal operations. Through a careful definition of the compositional operators, structures may derive their behavior from their components in accordance to the compositional relationship.

Similar to the IFC approach, a language specification can be derived on two levels. A first syntactic level specifies the vocabulary of primitive object

classes and their respective behaviors. This behavior, in itself, does not provide any meaning to the object class. In fact, a same data structure may define two or more object classes if as many different behaviors can be said to apply, for different purposes. On a second level, a selection of object classes is defined and, individually, named in order to express a semantic concept. These named classes can, subsequently, be composed into a hierarchical structure in order to define an appropriate representational schema. In contrast to the IFC approach, this semantic concept can be specified by the user and the representational structure composed accordingly. Alternative representations can be defined by altering the compositional structure or the selection of component classes. As each representation defines the same common operations, these can be reasonably plugged into an applicative interface for manipulation.

Comparing different representations requires a comparison of the component classes and of the overall compositional structures. At the same time, the expressive power of a representational framework is defined by its vocabularies of primitive object classes and compositional relationships. By carefully selecting the vocabulary of compositional relationships, users can be given the necessary freedom and flexibility to develop or adopt representations that serve their intentions and needs. At the same time, these can be formally compared with respect to scope and coverage in order to support information exchange. Such a comparison will not only yield a possible mapping, but also uncover potential data loss when moving data from less restrictive to more restrictive representations. Translation services can be provided based on both semantic identity and syntactic similarity.

4. SORTS

We are developing such a framework for representational flexibility, named *sorts*. Conceptually, a sort may define a set of similar data entities, e.g., a class of objects or the set of tuples solving a system of equations (Stouffs and Krishnamurti, 1998). Representationally, elementary data types define primitive sorts. These combine to composite sorts under formal compositional operations (Stouffs and Krishnamurti, 1997). The operation of sum allows for disjunctively co-ordinate compositions of sorts, where each sort may be – though not necessarily – represented in the data form; an attribute relationship provides for (recursively) subordinate compositions of sorts in both one-to-many and one-to-one instantiations. Other compositional operations can also be considered, such as an array- or grid-like composition of sorts. The result is a constructive, hierarchical description of sorts as compositions of other sorts, where each leaf node specifies a primitive data

type and every other node defines a compositional operation on its operand children nodes (figure 1).

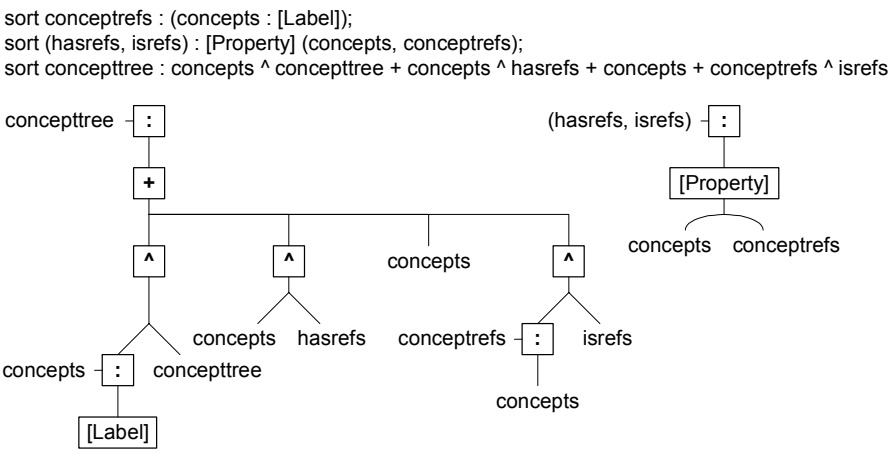

Figure 1. Textual and graphical definition of a recursive concepttree sort. A concepttree may include multiple instances of a single concept, with one instance defined and referenced by all other instances. '+' and '^' denote the operations of sum and attribute, respectively. ':' denotes the naming of a sort. 'Label' and 'Property' are primitive sorts; the latter defines a property relationship sort between two given sorts.

The definition of a sort includes a specification of the operational behavior of its members and collections, denoted as *forms*. The behavioral specification enables a uniform handling of forms of different sorts, on the proviso that the universe of all forms of a sort is closed under the respective operations. Primitive sorts have their behaviors assigned in order to achieve a desired effect, e.g., discrete behaviors for points and labels, an interval behavior for line segments, and an ordinal behavior for weights such as thickness or tones. On the other hand, a composite sort receives its behavior from its component sorts, based on its compositional relationships (Stouffs and Krishnamurti, 1997). The formal relationships between sorts enable the comparison and mapping of sorts as representational structures; the behavioral specification of sorts supports the mapping of information structures onto different sorts, such that the resulting information structures conform to the definition of the respective sorts or representations.

The concept of sorts only specifies a common syntax, allowing for different vocabularies and languages to be created, and providing the means to develop translation facilities between these. For example, a point may be specified with any number of coordinates depending on its dimensionality, its coordinates may constitute integers, reals or rationals, these may be bounded in space, etc. Sorts can be defined accordingly and, based on their

compositional structures, compared and related. For example, the operation of sum specifies a subsumption relationship on sorts, where one sort may match a part of another sort, under sum (Stouffs and Krishnamurti, 1997). Compositional structures under the attribute relationship, if not equal, may be fully (or partially) convertible: the attribute relationship is associative though not commutative. Based on the result of this comparison, translation support can be provided for and data loss monitored. For example, partial conversions always result in data loss; complete conversions may result in data loss depending on the behavioral categories of the constituent sorts.

Alternative design representations can be defined as variations on a given sort, by altering the components or the composition. As an example, consider a representation for a collection of drawings given a sort that defines a single drawing. By specifying an attribute composition with a sort of labels, a named collection of drawings is enabled similar to a set of layers in a CAD application. Alternatively, by specifying an attribute composition with a sort of points or rectangles, a layout can be represented for these drawings (figure 2). One step further, this sort can be modified to enable drawings to relate to parts within other drawings, allowing for detailing relationships to be specified in this layout.

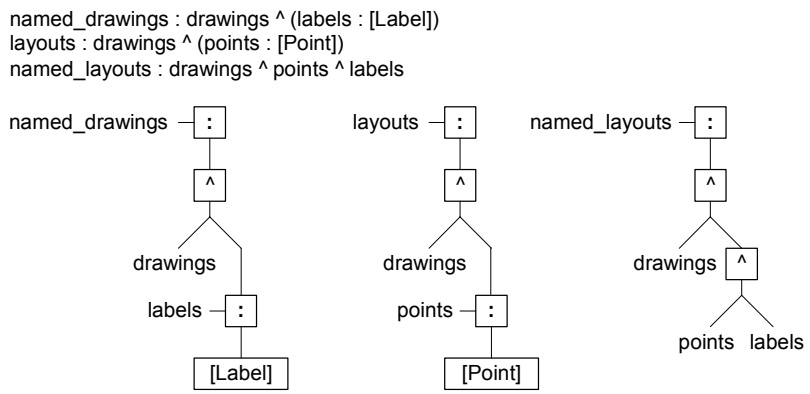

Figure 2. Sort definitions for named drawings, layouts of drawings, and named layouts of drawings, given a sort for a drawing.

As such, there is no imposition of concepts beyond the purely syntactical, and the alphabet of building blocks can be readily extended at all times. No language thus created ever needs to be static. Firstly, a vocabulary may be extended from the existing alphabet or by using newly developed building blocks. Secondly, representations may be updated by reconfiguring the existing composition of sorts or by extending it using additional component sorts. Far from having to redevelop the data structure and the applicative

operations, the concept of sorts aims to provide almost continuous support to evolving representations, providing for an environment that supports exploration and trial, even with respect to the representation. Representational structures can be compared and mapped, data can be readily converted to new and extended (or condensed) representations, and procedural operations remain applicative if such flexibility has been considered.

4.1 Example

Consider design information in the form of design constraints and related information, e.g., for a steel-framed building project (figure 3). This information may be stored in a database organized by type, i.e., constraints, variables, authors, constraint solvers, and other data entities (e.g., images, drawings or explanatory texts), with entities linked as appropriate (figure 4a). This presents an organizationally clean and efficient way of storing design information into a relational database. However, this organization is ill adapted to practical uses. While a representational organization may be dictated by efficiency in data retrieval and management, an effective visualization of the same data depends on user preferences and the task at hand. More important than the distinction between an efficient representation and an effective visualization, is the understanding that different partners in a collaborative environment adopt different views, specify different preferences and use different techniques, while visualizing and manipulating essentially the same information. In the example of the steel-framed building project, the set of design constraints is the result of a collaboration between architect, structural engineer and contractor, to name just a few.

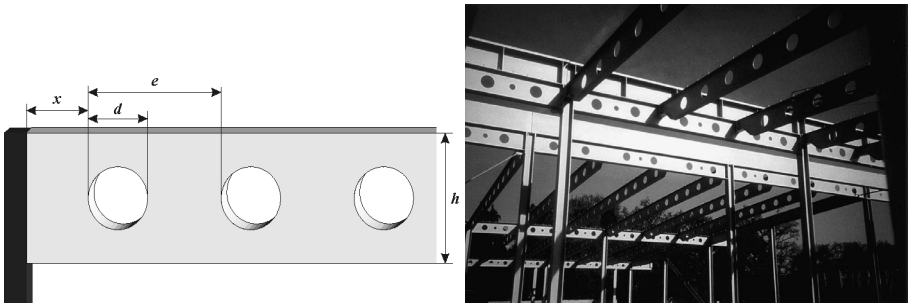

Figure 3. Design problem from a building project: the dimensioning of holes in steel beams.

An alternative visualization of the same project information may take into account the origins of these information entities, that is, for each author

the author's constraints, the constraint solver used, and other data entities provided by this author are specified (figure 4b). Each constraint specifies the variables that are affected by this constraint, and these variables, in turn, link back to the constraints that are defined over these, effectively linking constraints from different authors. Other links, e.g., between constraints and other data entities, can also be maintained and presented.

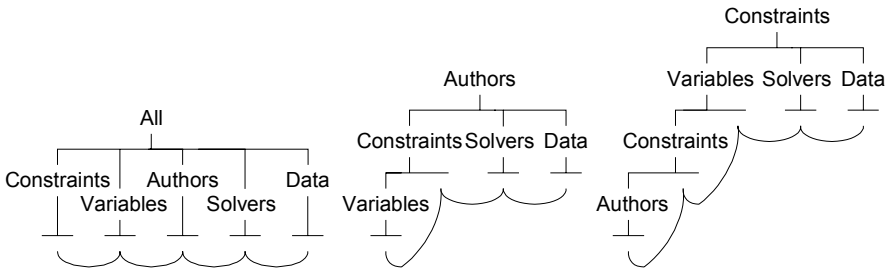

Figure 4. Three different organizational schemes for the same project information: a) by type, b) by author, and c) by design constraint.

Such an organization provides the user with an overview over the different authors' (or domains') contributions. One step further, the effective support of an actual design session may require the design itself, i.e., the design constraints, to form the centerpiece of the visual environment. Other information can be linked from the appropriate constraints in order to clarify each constraint's context and role in the design. A corresponding representation places the author's constraints at the top level (figure 4c). Every constraint specifies the variables affected, the author's constraint solver, and the data related to this constraint. Each variable, in turn, specifies the constraints from other authors that are defined over this variable, and each of these constraints specifies its author. Other links between information entities are additionally provided. This representation allows the author to directly access information related to each constraint. It also enables the user to evaluate the effect of altering a constraint on the design and whether such a change may interfere with other constraints specified by the partners in the collaboration.

As the example attempts to illustrate, *sorts* enable the development of different design views from a same data structure for different users and purposes. In the context of Web presentation, *sorts* can be adopted to prepare the retrieved information appropriately for presentation. Links and connections between information entities are treated as attributes to either or both entities. This approach allows for a uniform and flexible method of presenting information. Figure 5 shows snapshots from a VRML visualization for the steel-framed building project.

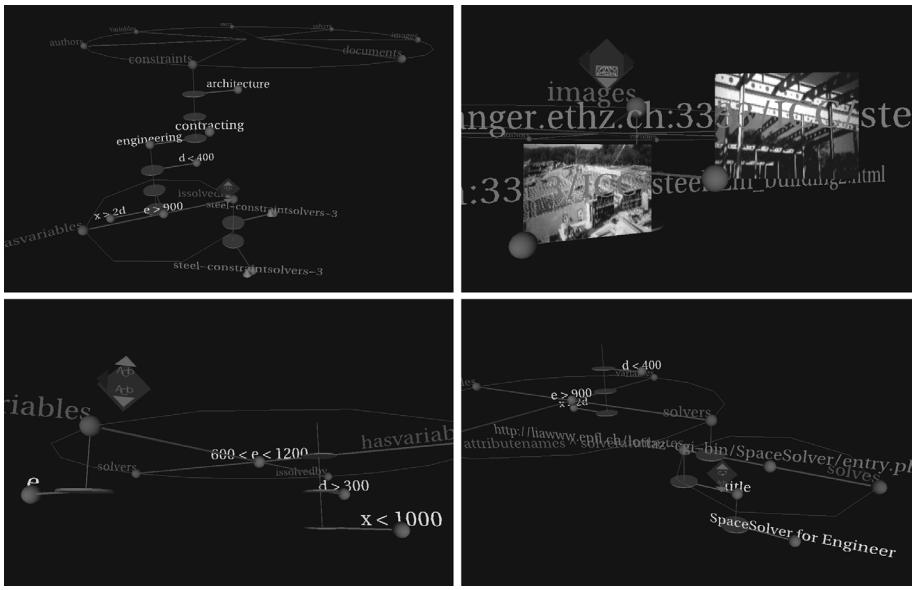

Figure 5. Snapshots of a VRML visualization for the steel-framed building project according to the first and last schemes of figure 4, respectively, a) a set of design constraints and b) related image entities, and c) views of the architect's and d) engineer's constraints. All VRML presentations are generated using an implementation of sorts.

5. CONCLUSION

New technologies, i.e., object technologies and XML, are fueling new interest in standardizing product models for the building industry. However, these same technologies, together with the Internet, reflect a strive for flexibility that stands in contrast to the concept of an all-encompassing standard. XML offers an example of how data exchange can be supported independently of the product models that are applied. *Sorts* attempts to achieve the same flexibility but with increased support for geometrical data and for the comparison of sorts and the translation of data between different sorts.

The concept of *sorts* aims to provide almost continuous support to evolving representations, providing for an environment that supports exploration and trial, even with respect to the representation. By specifying only a common syntax, it allows for different vocabularies and languages to be created, and provides the means to develop translation facilities between these. There is no imposition of concepts beyond the purely syntactical, and the alphabet of building blocks can be readily extended at all times.

6. ACKNOWLEDGMENTS

The research on sorts is partly funded by the Netherlands Organization for Scientific Research (NWO), grant nr. 016.007.007. The example from the steel-framed building project was part of a project funded by the Swiss National Science Foundation, grant nr. 5003-045357, while the first author was working at the Chair for Architecture and CAAD at ETH Zurich. This project constituted a collaboration between ETH Zurich, EPF Lausanne, and various partners from the Swiss building industry.

7. REFERENCES

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

Bazjanac, V., 1998, "Industry Foundation Classes: Bringing software interoperability to the building industry", *The Construction Specifier*, 6/98, p. 47-54.

Buckley, E., A. Zarli, C. Reynolds, and O. Richaud, 1998, "Business objects in construct IT", in: R. Amor (ed.) *Product and Process Modelling in the Building Industry*, Building Research Establishment, Watford, England, p. 117-130.

ISO, 1994, ISO 10303-1, *Overview and fundamental principles*, International Standardization Organization, Geneva, Switzerland.

O'Brien, M.J. and N. Al-Biqami, 2000, "XML, flexibility and systems integration", in: G. Gudnason (ed.) *Construction Information Technology 2000*, Vol. 2, Icelandic Building Research Institute, Reykjavik, Iceland, p. 656-661.

Stouffs, R. and R. Krishnamurti, 1998, "An algebraic approach to comparing representations", in: J. Barallo (ed.) *Mathematics & Design 98*, The University of the Basque Country, San Sebastian, Spain, p. 105-114.

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

Stouffs, R., R. Krishnamurti, and C.M. Eastman, 1996, "A formal structure for nonequivalent solid representations", in: S. Finger, M. Mäntylä and T. Tomiyama (eds.) Proceedings of *IFIP WG 5.2 Workshop on Knowledge Intensive CAD II*, International Federation for Information Processing, Working Group 5.2, p. 269-289.

Tolman, F.P. and H.M. Böhms, 2000, "Electronic business in the building-construction industry: preparing or the new Internet", in: G. Gudnason (ed.) *Construction Information Technology 2000*, Vol. 2, Icelandic Building Research Institute, Reykjavik, Iceland, p. 928-936.

van Nederveen, G.A., 2000, *Object trees: improving electronic communication between participants of different disciplines in large-scale construction projects*, Delft University of Technology, Delft, The Netherlands.

Woestenenk, K., 1998, "A common construction vocabulary", in: R. Amor (ed.) *Product and Process Modelling in the Building Industry*, Building Research Establishment, Watford, England, p. 561-568.

Woestenenk, K., 2000, "Implementing the LexiCon for practical use", in: G. Gudnason (ed.) *Construction Information Technology 2000*, Vol. 2, Icelandic Building Research Institute, Reykjavik, Iceland, p. 1049-1057.