Visualizing Representational Structures for Improving Data Conceptualization[1]

Rudi Stouffs, Michael Cumming

Faculty of Architecture, Delft University of Technology, The Netherlands

http://www.bk.tudelft.nl/users/stouffs/internet/

Abstract. Design problems require a multiplicity of viewpoints each distinguished by particular interests and emphases. Alternative viewpoints necessitate different representations of the same entity, albeit a building or building part, a shape or other complex attribute. Even within the same task and by the same person, various representations may serve alternative purposes defined by the problem context and the selected approach.

Understanding a representational structure serves to comprehend its use within the context of one's own intent. This requires the visualization of the representation and data, and the ability to manipulate and alter the data structure in order to explore its composition and scope and compare it to what may be desired or required.

We report on an active research effort to provide a 3D graphical interface for the manipulation of representational and data structures in terms of composition and scope. The interface and system make use of a particular framework for representational flexibility. This additionally supports the integration of data functions into data structures for the purpose of querying derived data.

Keywords. Representation, data visualization, data conceptualization.

Introduction

Design problems require a multiplicity of viewpoints each distinguished by particular interests and emphases. For instance, an architect is concerned with aesthetic and configurational aspects of a design, a structural engineer is engaged by the structural members and their relationships, and a building performance engineer is engaged by the thermal, lighting, or acoustical performance of the eventual design. Each of these views - derived from an understanding of current problem solution techniques in these respective domains - require different representations of the same entity. Even within the same task and by the same person, various representations may serve different purposes defined within the problem context and the selected approach. Especially in architectural design, the exploratory nature of the design process invites a variety of approaches and representations.

When exchanging data and information between various participants, there is a need to convert these into a format or representation that is useful to the recipient. Such conversions are often automated, extracting those parts of the data or information that fit the requirements of the recipient, and presenting these in an understandable way. Information that is not of explicit concern to the recipient in his or her subsequent manipulation or adaptation of the data may not be retained in the conversion, even though it could serve to clarify some aspects of the data to the recipient. Instead, if given the ability to view the information as presented by the author and to interpret it in light of his or her own needs and actions, the recipient may still take advantage of this additional information.

The need to interpret a current representation in light of subsequent activities also applies in the process of a single actor, if different representations serve different

[1] Stouffs, R., & Cumming, M. 2002. Visualizing Representational Structures for Improving Data Conceptualization. Proceedings of Education in Computer Aided Architectural Design in Europe: eCAADe20. 17-21 September. Faculty of Architecture, Warsaw University of Technology, Warsaw, Poland. K. Koszewski, & S. Wrona Eds. eCAADe. pp.328-332.

stages or activities within this process. Descriptions of computational design approaches (e.g. by Greg Lynn, Lars Spuybroek, and others) often articulate the choice of representation to express ideas and methodologies. Also, in the selection of software tools to assist a certain activity, understanding the data representation of the tool may serve to comprehend and conceive the functionalities and possibilities offered by this tool, with respect to the user's intention.

To understand a representational structure requires a visualization of this structure that distinguishes the individual entities and their compositional relationships. These entities and relationships can be reasonably abstracted such as to be able to compare different representations, and data collections thereof (Stouffs and Krishnamurti 2002). Effectively understanding a representation and its use within the context of one's own intent also necessitates the ability to manipulate and alter the structure. This allows one to explore its composition and scope and compare it to what may be desired or required.

In this paper, we report on an active research effort to provide a 3D graphical interface to representational and data structures that allows for the manipulation of these structures in terms of composition and scope. This research builds upon a particular framework for representational flexibility.

Representational flexibility

Stouffs and Krishnamurti (2002) present a representational framework, denoted *sorts*, that provides support for developing alternative representations of a same entity or design, for comparing representations with respect to scope and coverage, and for mapping data between representations, even if their scopes are not identical. A *sort* is defined as a complex structure of elementary data types and compositional operators, and is typically a composition of other *sorts*. Comparing different *sorts*, therefore, requires a comparison of the respective data types, their mutual relationships, and the overall construction.

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 a composition with a *sort* of labels, under an object-attribute relationship, a named collection of drawings is enabled similar to a set of layers in a CAD application. Alternatively, by specifying a composition with a *sort* of points or rectangles, a layout can be represented for these drawings. 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.

Such flexibility in exploring design representations can give designers the freedom to develop or adopt representations that serve their intentions and needs, while at the same time these representations can be formally compared with respect to scope and coverage in order to support information exchange. Consider, as another example, design information in the form of design constraints and related information, e.g., for a steel-framed building project (Lottaz, Stouffs, and Smith, 2000; Stouffs and Krishnamurti, 2002). The information consists, minimally, of a set of authors, a set of constraints for each author, and a common set of variables with each variable linked to the constraints defined over this variable. An organization of the design information by kind, i.e., constraints, variables, authors, and other data, with entities linked as appropriate, presents a straightforward and efficient way of storing this information into a relational database.

In order to effectively support an actual design session, the author's design itself, i.e., his or her design constraints, should form the focus of the information

organization. Other information entities can be made accessible from these, thereby clarifying each constraint's context and role in the design. Specifically, each constraint specifies the variables affected; each variable, in turn, specifies the constraints from other authors that are defined over this variable; and each of these constraints specifies its author. Links to other data can be additionally provided. This representational schema supports the user in evaluating the effect of altering a constraint on the design and whether such a change may interfere with other constraints specified by the collaboration partners.

Visualization of representational structures

Exploring alternative design representations necessitates an understanding of the representational structures that can only be achieved using visual means. Figure 1 offers a VRML visualization of design data from the steel-framed building project represented using the latter schema described above. The visual structure is an interpretation only of the representational structure and is, with the exception of the individual nodes, independent of the information kinds.
[FIGURE]

Figure 1. Snapshots of a VRML visualization for the steel-framed building project: left, a view of the architect's design constraints; right, a view of the engineer's data space.

As another example, DDDiver (Coomans and Timmermans, 2001) presents a tool for the interactive visualization and editing of complex relational data sets, e.g., found in object-oriented databases and product models. Its visualization is based on the distinction between relations of different kinds and on interactive techniques in order to explore large data sets. It emphasizes the different kinds of relations in order to improve the understanding of the object model and its instances. DDDiver is developed in part in response to the needs of a CAD system for the building and construction industry that is characterized by an innovative design-information modeling technique (Coomans and Timmermans, 2001; Coomans and van Leeuwen, 2001). As such, DDDiver not only concerns the visualization of a design object or database model to improve understanding of the model, but also provides the necessary interaction to build instance models, define new object types, and add instance relations.

Supporting the exploration of alternative design representations additionally requires the ability to change representational structures, e.g., by adding and removing components, or altering the compositional relationships. The latter may be achieved simply by altering the relationship type, e.g., between disjunctive and conjunctive, or between co-ordinate and subordinate. To a more complex extent, subordinate (object-attribute) relations may be reversed, altering the sequence of data types within the representational structure. The impact of such manipulations may be hard to conceive, and simply allowing one to visually move components within the compositional structure won't help much. Instead, altering and reversing object-attribute relations can also be considered in an alternative way as an expression of a different focus onto the data model. In an object-attribute relationship, the focus is primarily on the object. Reversing the relationship shifts this focus to the original attribute.

Visually, such a change of focus can be expressed by selecting a representational component and pulling this component out of the compositional structure. As a result, all compositional relationships between the selection and the remainder of the structure reduce to object-attribute relationships with the selection as the object. Other changes may also result in an attempt to maximize compatibility with the original representation and minimize data loss. We are extending the VRML visualization in

figure 1 to support such manipulations. This will provide the user with an enticing way for exploring alternative data representations in order to seek particular answers or otherwise query design representations.

Querying design information

Stouffs and Krishnamurti (1996) show how a query language for querying design information can be built from basic arithmetic operations and geometric relations that are defined as part of an extensible representational model for weighted geometries. These are augmented with operations derived from techniques of counting, pattern matching, and rules for composing more complex and versatile geometric and non-geometric queries. For example, by augmenting networks of lines that are represented using plane segments or volumes with labels as weights, and by combining these augmented geometries under the operation of sum, as defined for the representational model, colliding lines specifically result in geometries with more than one label. These collisions can easily be counted, while the labels of each geometry specify the colliding lines and the geometry itself specifies the location of the collision (Stouffs and Krishnamurti, 1996).

The representational framework of *sorts* extends on this model for weighted geometries. Firstly, it does not emphasize the use of geometries as basic entities with weights as attributes, but considers all kinds of entities equal, allowing even the specification of geometries as attributes to weights. Secondly, it considers a much broader variety of data kinds, including data functions. Data functions specify both a functional description and a result value; the result value is automatically recomputed using the functional description each time the data structure is traversed, e.g., when visualizing the structure. Data functions apply to a specified *sort*, which itself may be a data function. This *sort* must be related to the data function's *sort* within the compositional structure under a sequence of one or more attribute relationships, with restrictions.

Data functions can introduce specific behaviors and functionalities into representational structures, e.g., for the purpose of counting. Consider, for example, a composition of two *sorts* under an object-attribute relationship where the attribute *sort* specifies a cost to the object *sort*. Then, by augmenting the composition with a sum function, applied to the cost *sort*, as the top entity in the attribute hierarchy, the value of this function is automatically computed as the sum of all cost values. Binary functions, or compositions thereof, can be used to compute more complex derivations, such as the sum of all cost values multiplied by the respective sizes of the object entities. If in a larger representational structure, these object entities are themselves the attributes to other entities, possibly serving as type or clustering information, moving the data functions within the compositional structure, by altering the representational structure, will automatically alter the scope of the function and thus the result. This presents another incentive for the visual manipulation of *sorts*.

References

Coomans, M. and Timmermans, H.: 2001, DDDiver: 3D interactive visualization of entity relationships, *in* D. Ebert, J.M. Favre and R. Peikert (eds.), *Data Visualization 2001*, Springer, Vienna, pp. 291–299.

Coomans, M. K. D. and van Leeuwen, J. P.: 2001, Abstract but tangible, complex but manageable, *in* K. Nys, T. Provoost, J. Verbeke and J. Verleye (eds.), *AVOCAAD Third International Conference*, Hogeschool voor Wetenschap en Kunst, Brussels, pp. 50–59.

Lottaz, C., Stouffs, R. and Smith, I.: 2000, Increasing understanding during collaboration through advanced representations, *Electronic Journal of Information Technology in Construction*, 5, 1–25. [www.itcon.org/2000/1/]

Stouffs, R. and Krishnamurti, R.: 1996, On a query language for weighted geometries, *in* O. Moselhi, C. Bedard and S. Alkass (eds.), *Third Canadian Conference on Computing in Civil and Building Engineering*, Canadian Society for Civil Engineering, Montreal, pp. 783–793.

Stouffs, R. and Krishnamurti, R.: 2002 (in press), Representational flexibility for design, *in* J.S. Gero (ed.), *Artificial Intelligence in Design '02*, Kluwer Academic, Dordrecht, Netherlands.