# Querying Design Information through Visual Manipulation of Representational Structures

Rudi STOUFFS, Michael CUMMING
*Faculty of Architecture, Delft University of Technology, NL*

**Keywords:** visualisation, design information, design queries, representation

**Abstract:** Querying design information is an intricate aspect of the design process. Supporting arbitrary design queries requires both flexible design information models that can be modified and geared to the kind of question, and access to information in a uniform and consistent manner. 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. It is based on a framework for representational flexibility that supports the development of query languages. Specifically, we consider this as a framework for data management and include data functions as a means to support data analysis. We emphasise the interaction mechanisms that support both data management and analysis. In particular, we distinguish three types of interaction: the selection of an entity of primary interest, the selection of an extent of interest, and the insertion of data functions as a means for data analysis.

## 1 INTRODUCTION

Computational design relies on effective design information models, for the creation of design artefacts and for the querying of the characteristics of such artefacts. Mäntylä stated in 1988 that these (geometric) representations must adequately answer "arbitrary geometric questions algorithmically." Even without emphasis on the geometric aspects, this remains as important today. However, current computational design applications tend to focus on the representation of design artefacts, and on the tools and operations for their creation and manipulation. Techniques for querying receive less attention and are often constrained by the data representation system and methods. Nevertheless, querying a design is as much an intricate aspect of the design process as is creation and manipulation. Supporting arbitrary design questions requires both flexible design information models that can be modified and geared to the kind of question, and access to information in a uniform and consistent manner, so that new queries can be easily constructed and posed based on intent, instead of on availability.

Querying information from an information structure, whether representing design

information or other types of information, is not limited to retrieving existing information from the structure. Generally, querying information requires the analysis of existing information in order to derive new information that is not explicitly available in the information structure. Query languages commonly offer facilities for extracting and analysing selected information from within large information structures or databases. However, powerful query languages do not as such serve the end user (or designer) who is only interested in having easy access to the information, not in learning a new language. A common approach to this problem in computational design applications is to define a selection of queries that may be of general interest and to offer these to the user. "This strategy, however, means a severe access restriction for end users. Another approach is to define visual query languages that allow the user to express arbitrary queries without having to master the syntax of a rigid textual query language" (Erwig 2002). For example, Erwig and Schneider (2000) define a visual language and query interface for spatio-temporal information based on sketching object traces.

Visual query interfaces may rely on domain knowledge in order to provide specific functionality particularly appropriate for the type of information under investigation. For example, scientific visualisations can make use of the inherent dimensions of scientific data, connecting to three spatial and one temporal dimension, requiring only elementary linear algebra to lay out scientific data on a two-dimensional display (Groth and Robertson 2000). Similarly, data visualisations in Geographic Information Systems (GIS) generally make use of map projections to visualise a variety of geographically-related data. On the other hand, not all kinds of data structures can rely on specific domain knowledge in their visualisation. For example, when exploring general information structures or databases, data may be collected from a large variety of domains and may not fit a single domain-specific visualisation. In such cases, the challenge is to achieve an effective mapping from data to display (Groth and Robertson 2000).

Design is commonly a multi-disciplinary process, involving participants, knowledge and information from various domains. Therefore, 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 interested in 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 - requires a different representation of the same (abstract) 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.

In order to allow for various representations in support of different disciplines or methodologies, while enabling information exchange between representations and collaboration across disciplines, integrated data models are being developed that span multiple disciplines and support different views. This has led to, among others,

the ISO STEP standard for the definition of product models (ISO 1994) and the Industry Foundation Classes (IFCs) of the International Alliance for Interoperability (IAI), an object-oriented data model for product information sharing (Bazjanac 1998). These efforts can be said to characterise an a priori and top-down approach. In this approach, an attempt is made at establishing an agreement on the concepts and relationships that offer a complete and uniform description of the project data, independent of any project specifics (Stouffs and Krishnamurti 2001).

In order to provide a more extensive degree of flexibility that allows for the development of information models that are context, and thus project specific, various alternative modelling techniques are under investigation that consider a bottom-up, constructive approach. Feature-based modelling (van Leeuwen and Jessurun 2001) allows for the extensibility of conceptual schemas and for flexibility in modelling information structures that differ from the conceptual schemas these derive from. *Sorts* (Stouffs and Krishnamurti 2002) offers a constructive approach to defining representational structures that enables these to be compared with respect to scope and coverage and that presents a uniform approach to dealing with and manipulating data constructs. Woodbury et al. (1999) adopt typed feature structures in order to represent partial information models and use unification-based algorithms to support an incremental modelling approach.

In each of these latter approaches, however, the flexibility of specifying design information models at the representational level requires a visualisation and interaction that supports data exploration at this representational level (e.g. Coomans and Timmermans 2001; Datta and Woodbury 2002). This does not exclude the development of effective visualisations that do make use of domain knowledge, thereby relying on a particular representation. However, if the objective is to support effective collaboration between disciplines, then, an understanding of the information model beyond the individual discipline is necessary.

In order to build useful data exploration and analysis systems, Keim et al. (1996) emphasise the need to consider data management, data analysis and data visualisation together and examine the importance of interaction in this context. In this paper, we report on a 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, and that is based on a framework for representational flexibility that supports the development of an extensive query language. We consider the framework for representational flexibility in support of data management and the inclusion of data functions into this framework in support of data analysis. We present our development of a 3D visualisation for data structures within this framework with emphasis on the interaction mechanisms in order to support both data management and analysis. In particular, we distinguish three types of interaction: the selection of an entity of primary interest, the selection of an extent of interest, and the insertion of data functions as a means for data analysis.

# 2 DATA MANAGEMENT AND 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 organisation 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 organisation (Figure 1). 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.

# 3 DATA ANALYSIS AND DATA FUNCTIONS

Stouffs and Krishnamurti (1996) indicate 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).

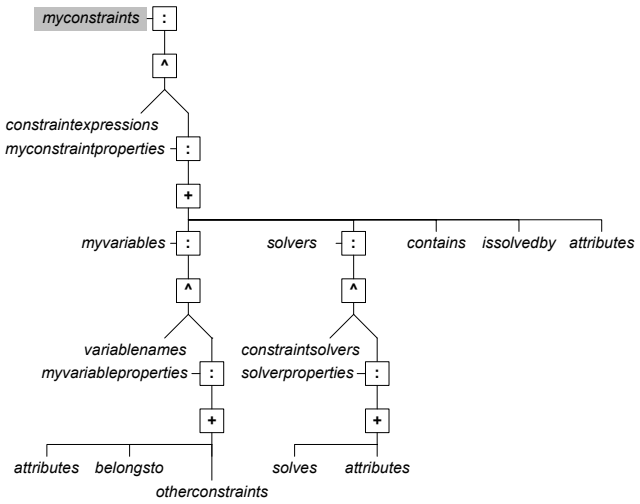


**Figure 1 Graphical depiction of a partial definition of a *sort* to support a design view from the design constraints example. '^' denotes a composition of *sorts* under an object-attribute relationship, '+' denotes a disjunctive co-ordinate composition of *sorts*, and ':' denotes a naming of a *sort*.**

The representational framework of *sorts* extends on this model for weighted geometries by de-emphasising the use of geometries as basic entities with weights as attributes and, instead, considering all kinds of entities equal, allowing even the specification of geometries as attributes to weights. Additionally, it considers data functions as a data kind, offering functional behaviour integrated into data constructs. In particular, 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 visualising 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 behaviours 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.

# 4 VISUALISATION OF GENERAL DATA STRUCTURES

Exploring alternative design representations and integrating data functions into design data models necessitates a degree of understanding of the representational structures that can only be achieved using visual (graphical) means. Figure 2 offers a VRML visualisation of design data from the steel-framed building project mentioned above, using a representational schema that places the constraints of the viewer at the focus of the information organisation. The resulting visual structure is an interpretation only of the representational structure and is, with the exception of the individual data entities, independent of the data kinds.

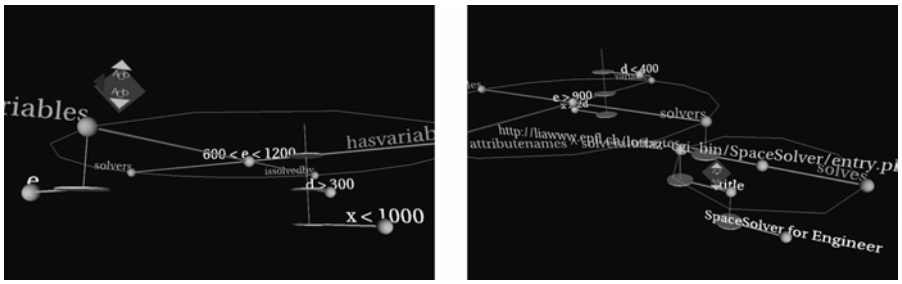


**Figure 2 Snapshots of a VRML visualisation for the steel-framed building project: (left) a view of the architect's design constraints; (right) a view of the engineer's data space.**

In comparison, DDDiver (Coomans and Timmermans 2001) presents a tool for the

interactive visualisation and editing of complex relational data sets, e.g., found in object-oriented databases and product models. Its visualisation is based on the distinction between relations of different kinds and on interactive techniques in order to explore large data sets. It emphasises 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 being based on the feature based modelling approach (van Leeuwen and Jessurun 2001). As such, DDDiver not only concerns the visualisation 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.

# 5 VISUAL MANIPULATION FOR QUERYING DESIGN INFORMATION

Supporting the exploration of alternative design representations additionally requires the ability to alter representational structures, e.g., by adding and removing components, or modifying the compositional relationships. Integrating data functions into design data constructs similarly necessitates the ability to intervene into the data structure and manipulate its composition of data entities and constructive relationships. In this section, we focus on three kinds of actions directly related to the querying of a design data model. These are the specification of a focus onto the data model expressing a particular interest, the selection of a part of the data model as extent of our interest, and the inclusion of data functions into the data structure as a means for data analysis with respect to our interest. Each of these actions results in a transformation of the representational structure.

First, such actions can be expressed directly at the level of the data structure's individual components and relationships (Coomans and Timmermans 2001). This approach offers complete control at the cost of a detailed knowledge or investigation of the data constructs and often painstakingly specific manipulations. At the same time, it can be questioned if such detailed control is generally required or even desirable.

An opposite approach may offer the user the ability to take intuitive actions at a higher level of abstraction, such as pulling out data entities of particular interest, that result in complex modifications at the constructive level. As an effect, it may seem to the user that there's no determinate relationship between the user's action and the aggregate constructive transformation or, at least, no determinate relationship that he or she can easily understand. In this case, the user's interaction with the design data model may be guided by trial and error, resulting in a series of similar actions leading towards a desired effect. If the user were to have a specific result in mind with respect to the data model, he or she might not be able to achieve this result in any reasonable amount of time, if ever.

However, if a specific representational result were desired, the user would be best

supported with the ability to manipulate the design model directly at the detailed level. Instead, we argue that when querying a design model, the user is mostly interested in retrieving the appropriate result and far less in the exact data construct that may be required for this purpose. Thus, it would suffice to provide the user with the ability to act upon the data model in such a way as to express his or her intention (i.e., focus, extent and means) and to relate this to a partial understanding of the data construct, in particular as it relates directly to the information under consideration.

Specifically, the expression of a focus onto a data model is directly related to the hierarchical composition of the models data entities under compositional relationships. In general, it can be said that more important entities are commonly found at a higher level in the data composition. For example, in an architectural design description, the geometrical information is commonly considered more important such that other information entities are assigned as attributes to the relevant geometric entities. Also, architectural design models are commonly organised by functional area, such as building, floor, or zone. Large object-oriented models often adopt a hierarchical structure of functional objects at various levels of detail, reflecting upon an increasingly narrower architectural focus. In particular, object-attribute relationships specify a focus onto the object, where the attribute expresses a qualifier with respect to the object.

Thus, when the user expresses a focus onto the data model in the context of querying this model, this must result in a transformation of the hierarchical structure that raises the data under analysis towards the top of the structure. Such a transformation can be achieved by reversing compositional relationships, such as object-attribute relationships. Reversing an object-attribute relationship shifts the focus to the original attribute. Such transformations can be computed automatically upon the selection of a particular data entity as focal point within the data structure. This computation may take place under the objective to maximise compatibility with the original representation and minimise data loss.

From a user's point of view, we envision an expression of a focus onto the data model by selecting a representational component and pulling this component out of the compositional structure. Similarly, specifying the extent of this structure can be expressed by selecting a collection of entities, and data functions can be inserted by attaching these to existing entities or relationships. We are extending the VRML visualisation 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.

# 6 CONCLUSION

In order to alter the scope and target of a query with respect to a given data structure, one can either rewrite the query or modify the data structure. The former requires detailed query specifications, the effect of which may not always be easily comprehensible. The latter requires a conceptual understanding of the data structure and of the effect of any manipulation of this structure. Effective visualisations of the

data structure, combined with intuitive ways of manipulating this structure, can support such understanding. We propose an approach for manipulating data structures that considers three types of actions: the specification of a focus onto the data model expressing a particular interest, the selection of a part of the data model as extent of our interest, and the inclusion of data functions into the data structure as a means for data analysis with respect to our interest. Each of these actions results in a transformation of the representational structure.

The adopted representational framework provides support for comparing representations with respect to scope and coverage, based on a comparison of the hierarchical structures. Such a comparison not only yields a possible mapping, but also uncovers the potential for data loss when moving data from less restrictive to more restrictive representations. As such, reorganisations can be guided in order to maximise compatibility with the original representation and minimise data loss.

For the specification of the query language, we rely on the integration of data functions into data structures for the purpose of querying derived data. Data functions can introduce specific behaviours and functionalities into representational structures that can support basic querying techniques, such as counting. Augmented with basic data operations and relations, and support for pattern matching and the specification of rules for composing complex queries, a query language can be defined for data structures. Data functions can simply be specified to apply to a given data type or aspect thereof. The specific scope and target of the data function will then be dependent on its location in the data structure. Altering the representational structure automatically alters the scope of the function and as such its result. In this way, specific answers may be retrieved by simple visual manipulation of the data structure.

## ACKNOWLEDGMENTS

This work is partly funded by the Netherlands Organisation for Scientific Research (NWO), grant nr. 016.007.007. The research on sorts benefits from contributions by Ramesh Krishnamurti (partially supported by the National Science Foundation, grant nr. CMS-0121549).

## REFERENCES

Bazjanac, V. 1998. Industry Foundation Classes: Bringing Software Interoperability to the Building Industry. *The Construction Specifier,* 6/98: 47-54.

Coomans, M. and H. Timmermans. 2001. DDDiver: 3D Interactive Visualisation of Entity Relationships. *Data Visualisation 2001*, eds. D. Ebert, J.M. Favre and R. Peikert, 291-299. Vienna: Springer.

Coomans, M.K.D. and J.P. van Leeuwen. 2001. Abstract but Tangible, Complex but Manageable. *Proceedings of AVOCAAD Third International Conference*, eds. K. Nys, T. Provoost, J. Verbeke and J. Verleye, 50-59. Brussels: Hogeschool voor Wetenschap en Kunst.

Erwig, M. 2002. Design of Spatio-temporal Query Languages. *Position paper presented at the Workshop on Spatio-temporal Data Models for Biogeophysical Fields, 8-10 April 2002, San Diego Supercomputer Centre, La Jolla, California*. www.calmit.unl.edu/BDEI/papers/erwig_position.pdf. 6 December 2002.

Erwig, M. and M. Schneider. 2000. Query-by-trace: Visual Predicate Specification in Spatio-temporal Databases. *Advances in Visual Information Management – Visual Database Systems*, eds. H. Arisawa and T. Catarci, 199-218. Boston, MA: Kluwer Academic.

Groth, D.P. and E.L. Robertson. 1998. Architectural Support for Database Visualisation. *Proceedings of the 1998 Workshop on New Paradigms in Information Visualisation and Manipulation*, 53-55. New York: ACM Press.

ISO. 1994. *ISO 10303-1, Overview and Fundamental Principles*. Geneva: International Standardization Organisation.

Keim, D.A., J.P. Lee, B.M. Thuraisingham and C. Wittenbrink. 1996. Database Issues for Data Visualisation: Supporting Interactive Database Exploration. *Proceedings of Database Issues for Data Visualisation, IEEE Visualisation '95 Workshop*, eds. A. Wierse, G. G. Grinstein, U. Lang, 12-25. Lecture Notes in Computer Science, 1183. Springer.

Lottaz, C., R. Stouffs and I. Smith. 2000. Increasing Understanding During Collaboration through Advanced Representations. *Electronic Journal of Information Technology in Construction,* 5: 1-25. www.itcon.org/2000/1/. 6 December 2002.

Mäntylä, M. 1988. *An Introduction to Solid Modelling*. Rockville, Md: Computer Science Press.

Stouffs, R. and R. Krishnamurti. 1996. On a Query Language for Weighted Geometries. *Proceedings of Third Canadian Conference on Computing in Civil and Building Engineering*, eds. O. Moselhi, C. Bedard and S. Alkass, 783-793. Montreal: Canadian Society for Civil Engineering.

Stouffs, R. and R. Krishnamurti. 2001. On the Road to Standardization. *Proceedings of* Computer Aided Architectural Design Futures 2001, eds. B. de Vries, J. van Leeuwen and H. Achten, 75-88. Dordrecht: Kluwer Academic.

Stouffs, R. and R. Krishnamurti. 2002. Representational Flexibility for Design. *Artificial* Intelligence in Design '02, ed. J.S. Gero, 105-128. Dordrecht: Kluwer Academic.

Van Leeuwen, J.P. and A.J. Jessurun. 2001. XML for Flexibility and Extensibility of Design Information Models. *Proceedings of CAADRIA 2001*, eds. J.S. Gero,

S. Chase and M. Rosenman, 491-501. Sydney: Key Centre of Design Computing and Cognition, University of Sydney.

Woodbury, R., A. Burrow, S. Datta and T. Chang. 1999. Typed feature structures and design space exploration. *Artificial Intelligence in Design, Engineering and Manufacturing,* 13(4): 287-302.