R Stouffs and R Krishnamurti. Flexibility and dynamism in digital design representations. *EuropIA'8*, Delft, Netherlands, April 2001

Flexibility and dynamism in digital design representations

Rudi STOUFFS[*], **Ramesh KRISHNAMURTI** [**]

Abstract: There is a strong need for computational representations for design that are both flexible and dynamic. This requires representational structures that can be reconfigured in the selection of components and their compositional relationships. It further implies the adoption of data entities and collections thereof that are empowered to dynamically alter their values, attributes, and composition based on higher-level or conceptual changes by the user. At the same time, the user must be offered new ways of expressing her intent in selecting objects for manipulation. We present a framework for representational flexibility named sorts in order to provide such flexibility and dynamism to the user.

Keywords: representations, object-oriented, grammars, sorts

1 Representational flexibility and dynamism

Effective digital design representations have been a topic of research ever since Sutherland's Sketchpad (SUT 63) 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 a 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 94), 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

[*] Delft University of Technology, Delft, The Netherlands. E-mail: r.stouffs@bk.tudelft.nl
[**] Carnegie Mellon University, Pittsburgh, Pa., USA

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 (BAZ 98), and more recently in XML (AEC 99; TB 00). 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.

Even if successful in the future, it is debatable whether a common product model will support the variety and flexibility it intends to enable. Conceivably, it may further restrict creativity and individuality by imposing a common 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.

In order to support such flexibility and dynamism, a framework must be conceived and developed that provides support for exploring alternative design representations, for comparing design representations with respect to scope and coverage, and for mapping design information between representations, even if their scopes are not identical. Typically, a representation is a complex structure of primitive data entities and compositional relationships (STO et al 96). Comparing different representations, therefore, requires a comparison of the primitive components, their mutual relationships, and the overall compositional structure. On the other hand, the expressive power of a representational framework is defined by its vocabularies of primitive data types and compositional relationships. By carefully selecting the vocabulary of compositional relationships, designers can be given freedom and flexibility to develop or adopt representations that serve their intentions and needs, while 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.

Flexible representations necessitate the ability to reconfigure both the selection of components and the compositional structure of these representations. Dynamic representations require data components that are empowered to dynamically alter their values, attributes, and composition based on higher-level or conceptual changes by the user (see section 2). Hereto, a behavioral specification of the data is required that enables a uniform approach for dealing with and manipulating different

data components. At the same time, representational flexibility necessitates new ways of expressing one's intent in selecting objects and their structure for manipulation (see section 3). We present a framework for representational flexibility named *sorts* in order to provide such flexibility and dynamism to the user (see section 4). It is based on a maximal element representation for shapes and enables shape and data recognition within a rule-based system.

2 The object-oriented myth

Besides a specification of the compositional relationships, a careful selection must be made of the primitive data types. Inducing dynamism into representations not only requires the ability to reconfigure representational structures by modifying the compositional relationships, but also implies the adoption of data entities and collections thereof that are empowered to dynamically alter their values, attributes, and composition based on higher-level or conceptual changes by the user. Most CAD applications have historically, and still at present, adopted an object-oriented approach at the conceptual level, providing users with line segments, surfaces, or solids as objects with attributes that maintain their properties at all times, unless explicitly altered by the user. It is a common misconception that the adoption of an object-oriented paradigm is generally beneficial at every level of an application development. While it is a conceptually attractive approach, because it is completely predictable and as such very understandable to the user, it is inimical to creative design. Creative design activities rely on a restructuring of information uncaptured in the current information structure, as when looking at a design provides new insights that lead to a new interpretation of the design elements (XXX yy). It can be proven that continuity of computational change requires an anticipation of the structures that are to be changed (KS 97). Creativity, on the other hand, is devoid of anticipation.

Consider for example the combination of two squares of line segments presented in figure 1. In a classic object-oriented approach, each square may define an object allowing it almost effortlessly to be resized and moved. As a result, however, a distinction of the individual line segments and a manipulation of these requires a conscious and explicit operation to redefine the square as a collection of four line segments. Visually, on the other hand, the composition in figure 1 contains not two but three squares. Irrespective of whether the composition has been defined as a collection of two square objects or as a collection of twice four line segments, neither representation allows the third square easily to be distinguished and manipulated, unless it is additionally defined in the composition, possibly by drawing the shape over. Instead, if line segments can be considered and represented as dynamic data entities that may be split into any number of smaller line segments depending on the purpose and task, the resulting composition of line segments would not only represent each of the three squares, but also an infinite number of other collections of line segments. Furthermore, representationally, none of these configurations has any higher importance, unless by choice of the designer. This provides the designer with the freedom to re-interpret a design in any way and have this interpretation supported by the system.

Such a representation has been at the heart of shape grammars research since more than two decades (STI 80). First for line segments in two dimensions, later extended to three dimensions and to plane segments and volumes (STO 94), as well as to various attributes or weights (STI 92), such as labels, line thickness, and colors (KNI 89), this maximal element representation defines any element as a possibly infinite set of (sub)elements that are each part of the original element. That is, any part of an element is an element, and users can deal with elements and shapes in indeterminate ways.

Figure 1. A composition of two or three squares.

2.1 A behavioral approach

In order to facilitate the comparison and recognition of elements under the part relationship, the maximal element representation is defined as a canonical representation, that is, identical elements must have an identical representation. Such a representation can easily be imposed on all types of data, including geometric and attribute entities, with an appropriate specification of their behavior as governed by a part relationship. Such a behavioral specification is also a prerequisite to allow an effective exchange of data between different representations and a uniform handling of different and a priori unknown data structures. As an example, consider the association of building performance data to design geometries. The behavior of these data as a result of alterations to the geometries can be expressed through a number of operations chosen to match the expected behavior. When the data is offered to an application, the behavioral specification is provided as well, allowing the application to properly interpret, manipulate, and represent this information without unexpected data loss.

Depending on the specification of this behavior, a primitive data type can be as simple as a single rational or real number, or as complex as a polygon- or NURBS-based surface model of a solid object. The simplest behavior that meets the requirements is a discrete behavior, corresponding to a mathematical set, where the part relation reduces to the subset relation. However, even a purely object-oriented behavior can be simulated, simply by combining the data type with a type of (unique) identifiers under an attribute relationship. The resulting data form is akin to a database of entities, where each entity has a unique key assigned. Under the discrete behavior, a conscientious decision is still required from the user on any change to the data entity. In order to take full advantage of the maximal element representation for line segments, an interval behavior may be specified: a line

segment defines an interval on an infinite line carrier; a collection of line segments is termed maximal if no two intervals on the same carrier are adjacent or intersect, i.e., these must be disjoint. Then, a line segment is part of another line segment if it is embedded in the other segment on the same carrier. Similar behaviors can be specified for plane segments and volumes.

Stiny (STI 92) explores the application of the maximal element representation to geometries with weights as attributes. Weights may be considered to denote thickness for points and lines, or tones for planes and volumes. A behavior for weights becomes apparent from drawings: a single line drawn multiple times, every time with different thickness, appears as it was drawn once with the largest thickness, even though it assumes the same line with other thickness. Thus, unlike behaviors described above, a collection of weights always combines into a single weight. This (maximal) weight has as value the least upper bound of all the individual weight values, i.e., their maximum value. This behavior can be termed *ordinal*; using numbers to represent weights, the part relation on weights corresponds to the less-than-or-equal relation on numbers.

The specification of a data type's behavior is commonly achieved through the description of a number of operations and their result on instances of the data type, including the creation and deletion of a data entity, and the merging of entities under some formal operations. If the selection of operations is chosen to be the same for every type, a uniform handling of all data types is enabled, with the specific differences hidden in the description of the type and its behavior under these common operations. If the selection of compositional relationships enables composite data representations to receive their behavior under the same operations from their primitive components and the relationships governing this composition, then, the uniform handling of data entities can be maintained even for complex data structures. For example, consider a co-ordinate composition of different data types into a single structure. Then, the behavior of the resulting representation is that of the component type for each component. On the other hand, consider a subordinate composition of different data types where each component, except the first, defines an attribute to the previous component. In this case, the behavior of the representation is defined by the behavior of its first component type. For example, a collection of line segments, each with attributes, is maximal if no two segments on the same carrier overlap and any two segments that touch have distinct attributes.

3 Of rules and grammars

The flexibility that a dynamic representation provides necessitates new ways of expressing one's intent in selecting objects for manipulation. As an infinite variety of possible interpretations of an object may exist and any collection of parts can be conceived as the subject of manipulation, computational methods for specifying a selection of parts are indispensable. For example, if a designer wishes to select any of the three squares in figure 1, she cannot be required to specify exactly which parts of line segments make up the boundary of this square. Instead, the system must be able to recognize all square shapes and offer these to the designer for selection. Computationally recognizing emergent shapes requires determining a transformation

under which a specified similar shape is a part of the original shape (XXX yy). Clearly, this matching problem depends on the representational structure. The maximal element representation is particularly appropriate as each element type specifies its own part or match relationship (KE 92; KS 97). Furthermore, if composite representational structures derive their behavior and part relationship from their component structures, the technical difficulties of implementing the matching problem only apply once for each primitive data type. As the part relationship can be applied to all kinds of data types, recognition algorithms can easily be extended to deal with arbitrary data structures, though a proper definition of what constitutes a transformation is still necessary. For example, search-and-replace functionalities in text editors generally consider case transformations of the constituent letters.

A specification of shape recognition, in combination with a subsequent manipulation of the recognized shape, can also be expressed in a shape rule. In general, a rule has the form $lhs \rightarrow rhs$; *lhs* (left-hand-side) specifies the similar shape or structure to be recognized, *rhs* (right-hand-side) specifies the manipulation leading to the resulting shape or structure. An application of a rule, then, consists of replacing the part matching *lhs*, under some allowable transformation, by *rhs*, under the same transformation. Rules can further be grouped into grammars. A grammar is a formal device for the specification of a language; it defines a language as the set of all structures or objects generated by the grammar, where each generation starts with an initial object and uses rules to achieve an object that contains only elements from a terminal vocabulary. The specification of shape rules and grammars leads naturally to the generation and exploration of possible designs; shapes emerging under a part relation is highly enticing to design search (MIT 93; STI 93). However, the concept of search is more fundamental to design than its generational form alone might imply. In fact, any mutation of an object into another one, or parts thereof, can constitute an action of search. As such, a rule can be considered to specify a particular compound operation or mutation, that is, a composition of operations and/or transformations that is recognized as a new, single, operation and applied as such. Similarly, the creation of a grammar is merely a tool that allows a structuring of a collection of rules or operations that has proven its applicability to the creation of a certain set (or language) of designs. Thus, rules and grammars are a means to contain and facilitate the flexibility and dynamism the maximal element representation provides us with.

4 A theory of sorts

An extension of the maximal element representation to a variety of data types has resulted in a concept for representational flexibility, termed, *sorts*. Conceptually, a sort specifies a set of similar models; sorts combine algebraically to form new sorts (SK 98). Consider a sort as a set of models that are described in terms of a single set of equations. Then, each individual of the sort corresponds to a distinct assignment of values to the parameters in the set of equations. For example, a point is specified by its tuple of coordinates. Furthermore, sorts can be related by comparing their systems of equations, in a mathematical manner. For instance, since each equation

constraints the values its parameters may adopt, a sort *subsumes* another sort if its equations form a subsystem of the other sort's equations.

In practice, elementary data types define primitive sorts, which combine to composite sorts under formal compositional operations defined over sorts (SK 97). In shape grammars, a distinction is made between co-ordinate and subordinate compositions of elements from different types. Elements from various graphical types existing alongside one another in a same shape adhere to a co-ordinate relationship; on the other hand, the attribute relationship specifies a subordinate composition of a shape with any label or weight. With respect to sorts, an operation of sum allows for disjunctively co-ordinate compositions of sorts, under many-to-one and many-to-many instantiations, where each sort may – though not necessarily – be represented in the data form. As an example, a rule has both a *lhs* and *rhs* component, either of which can be omitted. An attribute relationship provides for (recursively) subordinate compositions of sorts in both one-to-many and one-to-one instantiations. For example, the sort of labeled points is specified as a sort of points, with one or more labels assigned as attribute to each point in the data form. Other compositional operations can also be considered, such as an array- or grid-like composition of sorts. Assigning names to sorts provides for a semantic-like differentiation of sorts that may otherwise be syntactically identical, e.g., *lhs* and *rhs* denote equivalent – not identical – sorts.

The definition of a sort also includes a specification of the operational behavior of its members and collections, denoted as *forms*, thereof. This 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. These are the common arithmetic operations of sum, difference and product (intersection). 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 (SK 97). 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 are conform to the definition of the respective sorts or representations.

4.1 Representations for CAD

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, compared and related, and translation support provided for. 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 (SK 96). Alternatively, by specifying an attribute 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.
The architecture of *sorts* also serves basic CAD functionality. Consider the most common operations of creation and deletion, and of selection and deselection of objects; consider two spaces, one representing the design, the other the selection. Selecting an object removes it from the design space and inserts it into the selection space, deselecting an object has the opposite effect. In both cases, an object removal is followed by an insertion of the same object. Using two sorts, one for the design and another for the selection, the operations of selection and deselection result in a difference in one sort followed by a sum in the other. The operations of creation and deletion behave similarly, except that one of the sorts is the empty sort.
Under the part relation, every element of a sort specifies an indefinite set of elements that are each part of the original element. Thus, *any* part of an object is an object and users can deal with objects in indeterminate ways. This is quite distinct from the selection process in conventional CAD approaches where the only objects that are selectable correspond to those prescribed minimal entities that have been predefined in the data-structures. As a result, the selection of an object can be commonly achieved by a straightforward search in the database, following a lead by the user, e.g., the position of the cursor at the moment of the activation of the search. Under sorts, since we consider an object as a definite description of indefinitely many parts, the action of selecting a part may include more powerful means of describing what is to be selected. Since any part can be selected, in the extreme this may require the user to create the selection as an object that is a part of the design. Thus, the operations of creation and selection may invoke the same action sequence in terms of describing the resulting object, whether it is a newly created object or a selected existing part. Therefore, these operations may be presented to the user in a similar way as to emphasize the common dialogue in achieving these results, even if the results themselves are conceptually quite different.

4.2 Representations for data exchange

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 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. The ICCS project (LOT et al 00) offers an example of the use of sorts for exchanging data between different representations in the context of a building project (STO et al 98).

5 Conclusion

There is a strong need for computational representations for design that are both flexible and dynamic. Flexible representations can offer multiple design views of a same building or product, in support of a variety of partners and disciplines. Dynamic representations can adapt themselves to suit different tasks or phases in the design process, even for a same designer. Hereto, a framework for representational flexibility is needed that supports an exploration of alternative design representations, a comparison of design representations with respect to scope and coverage, and a mapping of design information between representations. We propose a theory of *sorts*, offering a description of representational structures using formal compositional relationships over primitive data types. This formalism additionally allows for dynamic information entities, enabling creative design by supporting re-interpretations of existing design descriptions through emergent forms, and considers a methodology for dealing with such creative dynamism within a design application. Together, these will offer strong support for creative design by liberating the designer from a representational straitjacket.

6 Acknowledgment

This work is partly funded by the Netherlands Organization for Scientific Research (NWO), grant nr. 016.007.007.

7 References

[AEC 99] aecXML. 1999. *AecXML: A framework for electronic communications for the AEC industries*, IAI aecXML Domain Committee. http://www.aecxml.org/technical/

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

[ISO 94] ISO. 1994. *ISO 10303-1, Overview and fundamental principles*, International Standardization Organization, Geneva, Switzerland.

[KS 97] Krishnamurti R. and Stouffs R. 1997. Spatial change: continuity, reversibility and emergent shapes, *Environment and Planning B: Planning and Design* 24, p. 359-384.

[KE 92] Krishnamurti R. and Earl C.F. 1992. Shape recognition in three dimensions, *Environment and Planning B: Planning and Design,* 19*,* p. 585–603

[KNI 89] Knight T. W. 1989. Color Grammars: Designing with Lines and Colors, *Environment and Planning B: Planning and Design* 16, p. 417-449.

[LOT et al 00] Lottaz C., Stouffs R., and Smith I. 2000. Increasing understanding during collaboration through advanced representations, *Electronic Journal of Information Technology in Construction* 5(1), p. 1-24.

[MIT 93] Mitchell W.J. 1993. A computational view of design creativity, *Modeling Creativity and Knowledge-Based Creative Design* (eds J.S. Gero and M.L. Maher),

Lawrence Erlbaum Associates, Hillsdale, N.J.
[SK 96] Stouffs R. and Krishnamurti R. 1996. The extensibility and applicability of geometric representations, *3rd Design and Decision Support Systems in Architecture and Urban Planning Conference, Architecture Proceedings*, Eindhoven University of Technology, Eindhoven, The Netherlands, p. 436-452.
[SK 97] Stouffs R. and Krishnamurti R. 1997. Sorts: a concept for representational flexibility, *CAAD Futures 1997* (ed. R. Junge), Kluwer Academic, Dordrecht, The Netherlands, p. 553-564.
[SK 98] Stouffs R. and Krishnamurti R. 1998. An algebraic approach to comparing representations, *Mathematics & Design 98* (ed. J. Barallo), The University of the Basque Country, San Sebastian, Spain, p. 105-114.
[SK 00] Stouffs R. and Krishnamurti R. 2000. Sortal grammars: a framework for exploring grammar formalisms, *Quality, Reliability, and Maintenance* (ed. G.J. McNulty), Professional Engineering Publishing, Bury St. Edmunds, UK, p. 367-370.
[STI 80] Stiny G. 1980. Introduction to shape and shape grammars, *Environment and Planning B: Planning and Design* 7, p. 343-351.
[STI 92] Stiny, G. 1992. Weights, Environment and Planning B: Planning and Design 19, p. 413-430.
[STI 93] Stiny G. 1993. Emergence and continuity in shape grammars, *CAAD Futures 1993* (eds. U. Flemming and S. Van Wyk), North-Holland, Amsterdam, p. 37-54.
[STO 94] Stouffs R. 1994. *The Algebra of Shapes*, Ph.D. dissertation, Department of Architecture, Carnegie Mellon University, Pittsburgh, Pa.
[STO et al 96] Stouffs R., Krishnamurti R., and Eastman C.M. 1996, A formal structure for nonequivalent solid representations, Proceedings of *IFIP WG 5.2 Workshop on Knowledge Intensive CAD II* (eds. S. Finger, M. Mäntylä and T. Tomiyama), International Federation for Information Processing, Working Group 5.2, p. 269-289.
[STO et al 98] Stouffs R., Tunçer B., and Schmitt G. 1998. Supports for information and communication in a collaborative building project, *Artificial Intelligence in Design 98* (eds. J. Gero and F. Sudweeks), Kluwer Academic, Dordrecht, The Netherlands, p. 601-617.
[SUT 63] Sutherland I.E. 1963. Sketchpad, A Man-Machine Graphical Communication System, *Proceedings of the Spring Joint Computer Conference*, Detroit, Michigan, May 1963, and MIT Lincoln Laboratory Technical Report #296, January 1963.
[TB 00] Tolman F.P. and Böhms H.M. 2000. Electronic business in the building-construction industry: preparing for the new Internet, Construction Information Technology 2000 (ed. G. Gudnason), vol. 2, Building Research Institute, Reykjavik, Iceland, p. 928-936.