Thoughts on a Designer-friendly Shape Grammar Interpreter

Andrew I-kang Li
The Chinese University of Hong Kong, China
http://www.arch.cuhk.edu.hk/servera/staff1/andrew/

Abstract. *Discussions of shape grammar interpreters overlook a fundamental issue: the model of the designer's work. Such a model would provide guidance for developing an interpreter with an appropriate interface.*
In this paper, I first propose a model in which the designer's work is to create and test generative specifications of languages of designs. I call this model designer-centered generative design. Then, I examine the characteristics of shape grammar and how they support or impede this model of work. Finally, I discuss the implications for the design of an appropriate shape grammar interpreter. These provide guidelines for implementing such an interpreter for testing.

Keywords. *Shape grammar; interpreter.*

Introduction

A number of interpreters have been developed to make shape grammars easier and more appealing for designers to use[1]. However, these have not been as successful as one might have expected, and it is recognized that more work needs to be done.

It is often suggested that what needs the work is the user interface. For example, Knight (1999) writes that:

More efforts have gone to computational problems than to interface ones. Implementations of simple, restricted grammars that require only graphic, nonsymbolic, nonnumerical input are needed.

Similarly, Chase (2002, 162) writes:

Further research on interactions in grammar systems could bridge this gap between CAD system and grammar-based interfaces.

This view is correct as far as it goes. However, it overlooks a more fundamental issue: the designer's model of work. We must understand this before we can understand the interface of a system. That is, usefulness precedes usability. As Mirel (2004, 32) puts it:

[T]he idea for usable systems is to be "useful" by supporting the right model of people's work and "easy to use" by disclosing an application's logic and operations.

In this paper, I first propose a model in which the designer's work is to create and test generative specifications of languages of designs. I call this

[1] *For a recent list see Chau et al. (2004).*

model designer-centered generative design.

Then, I examine the characteristics of shape grammar and how they support or impede this model of work.

Finally, I discuss the implications for the design of an appropriate shape grammar interpreter. These provide guidelines for implementing such an interpreter for testing.

Designer-centered generative design

Description

In designer-centered generative design, there are a designer and two entities, both of which are acted upon by the designer:

- A generative specification that defines a language of designs (the specified language). This is manipulated by the designer.
- A language of designs that satisfy some criteria (the target language). The criteria are determined by the designer.

The designer's goal is to converge the specified language and the target language. The formalism used to characterize the specification is immaterial; it could be a shape grammar, an L-system, or a programming language.

The designer's work consists of two tasks: creating the specification, and evaluating whether the specified designs are in the target language. These tasks are present in both analysis and synthesis.

In analysis, the designer begins with a finite sample of designs that are considered to belong to the target language. Her goal is to create a specification that defines all and only the designs in the target language. To do this, she carries out an iterative process of revising the specification and evaluating whether the specified designs are in the target language.

In synthesis, the designer may begin with criteria for the target language[2]. As in analysis, her goal is to create a specification that defines all and only the designs in the target language. And, again, she revises the specification and evaluates the specified designs iteratively.

One key feature here is that the designer revises the specification repeatedly. That is, although her focus is a specification and not an object, she is engaged in design. And, as is well known, design is unpredictable. This leads inevitably to the conclusion that the designer must not be restricted in her manipulation of the specification: emergence is essential. We will have more to say on this below.

It also follows that restrictions on the specification are restrictions on the specified language and may prevent it from converging with the target language. Creating specifications under such restrictions falls outside the proposed model.

A typology of generative design

As mentioned above, in this model, both tasks – creating the specification, and evaluating the specified designs – are performed by the designer. Hence the name designer-centered generative design.

However, either of these tasks could be performed other than by the designer alone, by a human or a nonhuman, with or without the designer. These suggest a context within which we can place designer-centered generative design (see table 1).

Consider the first task, creating the specification. It is easy to imagine that a first version is created by someone else. It could be a teacher, for an exercise[3]; an analyst, studying a style; or even the designer herself, at some earlier time. The design-

[2] In practice, the designer often begins with few or no criteria for the target language; determining these is then part of the design process (Lawson 2004). In addition, she may not need all and only the designs in the target language; she may be satisficing, in which case only a few designs, or even one design, may suffice. Both these cases are subsumed within the model.

[3] This was the premise for Li's (2002) Yingzao fashi interpreter.

Table 1. Typology of generative design, suggested by different ways of creating the specification and of evaluating the designs.

	designs evaluated by designer	**designs evaluated by an external authority**
specification created by designer	designer-centered generative design	scientific method
specification created by designer and another	design from the known	scientific method
specification created by another	genetic algorithms	simulated annealing

er takes this first version and reworks it, creating a new specification. This is design, not from scratch, but from the known. It is equally easy to imagine an entire specification being created or revised automatically, as in genetic algorithms.

As for the second task, evaluating the designs, this is usually done by the designer[4]. But she might also entrust the task to an external authority, such as a connoisseur, a fitness function, or an informant or "native stylist" (such as Alvaro Siza in Duarte's (2001) study of Siza's Malagueira housing). If the authority is considered objective, then we have the scientific method.

The extreme case is that in which both creation and evaluation are both done automatically, such as simulated annealing (Shea and Cagan 1997). In this case, no human is directly involved in the design process.

No doubt other scenarios can be fit into this framework.

Shape grammars and designer-centered generative design

Shape grammars have three important characteristics which support designer-centered generative design.

First, they are themselves generative specifications. They can be freely manipulated by designers (or artificially) and support the scenarios in the typology.

Second, they support emergence. This, as we have seen, is crucial to creating specifications freely, as opposed to merely implementing them.

Third, they are graphic. Designers work graphically, so they are likely to find shape grammars more congenial than, say, L-systems.

However, shape grammars also have a characteristic that tends to impede the proposed model of work: the transformation articulated in a single shape grammar rule is often smaller than designers want to consider. Put another way, a designer often thinks about operations that are too complex to be expressed as a single rule. Instead, they are encoded as a deterministic sequence of rules (Li 2001). These are trees when the designer is thinking forest.

For example, the designer may want to put an opening in a wall. For her, this is a single decision, but a grammar requires many steps to execute that single decision, steps that will not likely interest her. Liew (2004) has investigated this issue, which is simply another aspect of usefulness versus usability.

Implications for a designer-friendly shape grammar interpreter

We have considered a model of work and a formalism. Now we consider how to mediate between the two, which is the task of the interpreter. In this case, a designer-friendly shape grammar in-

[4] *Which is why grammatical analysis is inherently subjective (Li 2004).*

terpreter is simply a tool for both making and testing shape grammars. Thus, the guiding principle is that the interpreter must manifest this model of work to the designer. From this we can derive more concrete desiderata. We proceed by considering the designer's lower-level tasks.

Manipulating rules

The designer's first such task is to manipulate rules. These must be freely variable; it follows that emergence must be supported. Tapia's (1999) GEdit is notable in allowing such freedom in creating rules and in supporting emergence. However, revising rules is impossible; the designer must create new ones. In contrast, McGill's (2002) Shaper2D allows only a narrow range of rules, but within that range allows the designer to revise them freely.

A second desideratum is direct manipulation of rules (and, by extension, shapes and basic elements). This would exploit the graphic immediacy of shape grammars. A good example is McGill's (2002) Shaper2D, which allows the designer to interact directly with the rules; this is reinforced by instantaneous feedback. Another example of direct manipulation of graphic rules is Stagecast Creator™ (previously known as KidSim and Cocoa) (Smith et al. 1996), a children's simulation program that uses visual programming.

A third desideratum is the ability to create subroutines. Liew's (2004) implementations are compelling arguments for this ability. There have been no other implementations in shape grammar, but Creator™ (Smith et al. 1996) is an impressive example of direct manipulation. This is a solvable problem.

One might ask about technical capabilities, such as labels, weights, descriptions, and parameterization. Insofar as omitting them would not compromise the model of work, they are not crucial. Of course, implementing them would make an interpreter more powerful, but the model of work would be the same.

Manipulating grammars

Designer manipulates not only rules, but also grammars. The same logic applies, that designers must be able to manipulate grammars directly and freely. They must be able to organize and annotate the rules as they see fit. They must also be able to store and recall grammars. This last capability will make it possible to implement design from the known, as discussed above. It would also allow grammars to be exported, subjected to artificial evolution, and re-imported. At the very least, it would enable the designer to spread her work over several sessions. Again, Creator™ (Smith et al. 1996) is a good example.

Evaluating designs

At this level, there are no implications for how an interpreter should handle evaluation; it is a matter purely for the designer. At the same time, a particular method of evaluation may require particular treatment. For instance, if the designer wants to evaluate real objects, and not screen images, then the interpreter will have to export the designs in a format appropriate for digital manufacturing, as Wang and Duarte (2002) do.

Producing designs

According to the proposed model, the designer does not produce the designs that she evaluates. Since she is interested in all and only the target designs, then she needs to see all the specified designs. And the best way to ensure completeness, especially when emergence is supported, is to produce designs automatically. Tapia's (1999) GEdit implemented such an automatic production mechanism.

But this approach could also overwhelm the designer, who might well to have some control over the generation of designs. This could take many forms. For instance, she could choose individual derivations and generate single designs. Or she could have the interpreter generate all the designs below a single node on the derivation tree.

Or she could have the interpreter generate designs randomly. In addition, the designer may not need to produce all the specified designs. If she is satisficing, she may be content to produce only a few designs and choose among those. Chase's (2002) discussion is relevant here.

An additional form of control is navigating through the design space. This could be made possible with a dynamic and interactive derivation tree.

There is another possible goal for the interpreter, a goal that is secondary to but certainly consistent with the model. That is to help designers understand how shape grammars work. This is the primary goal of McGill's (2002) Shaper2D, which, as has already been discussed, is indeed easy to understand. Li's (2002) Yingzao fashi section interpreter also tries to be transparent to the user.

Discussion

We have now identified some features that should be implemented in an interpreter that supports designers in both making and testing shape grammars. Whether these features are useful – and indeed whether the model they are based on is useful – can be determined only by making and testing implementations. This includes expanding the scope of evaluation to include both virtual objects and physical objects produced digitally.

But even without being implemented, our model, which we might call grammatical design, provides some helpful insights into the relation between grammars and design.

One issue is whether grammars are useful in designing from scratch (or perhaps "real" design). Here we may use Smithers's (2002, 7) description of design as a process of "arriving at a kind of solution without starting with a problem."

One view is that grammars are not useful in designing from scratch. And certainly our model is less convincing when the designer lacks criteria for the target language. The other view is that grammars are useful, because they can capture a designer's moves after the fact.

One could say that both views are wrong. The first view ignores the fact that design is almost never totally from scratch. It underrates the necessity of knowledge and experience (Lawson 2004) and their presence in the design process. And the second view conflates retrospection and action. But both views can also be right. Our model is consistent with both the first view (grammatical design from scratch may not be convincing) and the second (grammatical design from the known is credible).

Our model shows how to operationalize design from precedent: some design knowledge can be embedded in analytical grammars, recalled, and reworked to specify new languages of designs[5].

Certainly, analysis has so far proven to be a success of shape grammar. Our model suggests that building on this analytical strength could yet benefit synthesis.

One could imagine education along old lines: students analyze old works and try to produce new examples. This may be a disagreeable idea to some, but automating appropriate parts of the process and making many styles available might make it appealing. At least it would be one more tool that could be used.

References

Chase, S. C. 2002: A model for user interaction in grammar-based design systems, in Automation in construction 11, pp. 161–172.

Chau, H. H., X. Chen, A. McKay, and A. de Pennington: 2004, Evaluation of a 3D shape grammar implementation, in J. S. Gero (ed.), Design computing and cognition '04, Kluwer, Dordrecht, pp. 357–376.

[5] Knight (1994) has discussed the technical aspects of such a scenario.

Duarte, J. P.: 2001, Customizing mass housing: a discursive grammar for Siza's Malagueira houses, PhD dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Mass.

Knight, T. W.: 1994, Transformations in design: a formal approach to stylistic change and innovation in the visual arts, Cambridge University Press, Cambridge, England.

Knight, T. W.: 1999, Shape grammars in education and practice: history and prospects, in International journal of design computing 2 (www.arch.usyd.edu.au/kcdc/journal/vol2/knight/index.html).

Lawson, B.: 2004, What designers know, Architectural Press, Oxford.

Li, A. I.: 2001, A shape grammar for teaching the architectural style of the Yingzao fashi, PhD dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Mass.

Li, A. I.: 2002, A prototype interactive simulated shape grammar, in Krzysztof Koszewski and Stefan Wrona (eds), Proceedings of the 20th conference on education in computer aided architectural design in Europe, Education in Computer Aided Architectural Design in Europe, Warsaw, pp. 314–321.

Li, A. I.: 2004, Styles, grammars, authors, and users, in J. S. Gero (ed.), Design computing and cognition '04, Kluwer, Dordrecht, pp. 197–215.

Liew, H.: 2004, SGML: a meta-language for shape grammars, PhD dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Mass.

McGill, M. C.: 2002, Shaper2D: visual software for learning shape grammars, in Krzysztof Koszewski and Stefan Wrona, Proceedings of the 20th conference on education in computer aided architectural design in Europe, Education in Computer Aided Architectural Design in Europe, Warsaw, pp. 148–151.

Mirel, B.: 2004, Interaction design for complex problem solving: developing useful and usable software, Morgan Kaufmann, San Francisco.

Shea, K., and J. Cagan: 1997, Innovative dome design: applying geodesic patterns with shape annealing, in Artificial intelligence for engineering design, analysis and manufacturing 11, pp. 379–394.

Smith, D. C., A. Cypher, and K. Schmucker: 1996, Making programming easier for children, in Interactions, ACM 3 (5), pp. 58–67.

Smithers, T.: 2002, Synthesis in designing, in J. S. Gero (ed.), Artificial intelligence in design '02, Kluwer, Dordrecht, pp. 3–24.

Tapia, M.: 1999, A visual implementation of a shape grammar system, in Environment and planning B: planning and design 26, pp. 59–73.

Wang, Y., and J. P. Duarte: 2002, Automatic generation and fabrication of designs, in Automation in construction 11, pp. 291–302.