Grammar Based Design: Issues for User Interaction Models

Scott C. Chase
University of Sydney, Australia

Grammar based production systems are considered potentially powerful design tools by their ability to generate sets of designs adhering to user specified constraints. However, development of such tools has been slow, partly because of the lack of good interaction between user and system. This paper describes modes of user interaction and control possible with grammar based design systems and presents issues to be examined in the development of models that represent the locus of interactions possible with such systems. The examination of existing grammar based systems provides empirical evidence to support the validity of such models.

Background

In architecture, media can be defined as a tool for selecting, gathering, organizing, storing, and conveying knowledge in representational forms. Media enable ideas to be externalized and evaluated and hence become a highly influential factor in the design process.
Excerpt from ACADIA '99 theme

This definition of media can be seen to expand the more traditional definition of media as representation to include that of mechanism and process. Design grammars certainly fit into this definition, as they have been shown to be useful mechanisms for storing design knowledge, generating designs, and analysing existing sets of designs.

The use of grammars in design has been twofold: 1) to analyse existing sets of designs and develop grammars that could be used to generate the set of existing designs, as well as new designs in the same style; 2) to generate new, stylistically consistent languages of designs. Grammars have also been used to transform existing styles in a new language of designs (Knight, 1994).

Grammar based design systems have the potential to both automate the design process and allow greater exploration of design alternatives. Grammars are production systems that generate designs according to a specific set of user defined rules (the grammar). A well-defined grammar will generate a set of designs that adhere to a specific set of user defined constraints. A grammar based design system has the potential to generate designs with little or no input on the part of the user. As a grammar can generate a large set of designs that fulfil the designer's needs, this can give the designer the potential to evaluate a large number of alternative designs without laborious work. This set of designs generally includes many that might have been overlooked by the designer working without the aid of a grammar, thus paving the way for possible innovative designs.

Grammar based systems may be considered active tools, as opposed to the more traditional CAD tools on the market today, which, in contrast, may be considered passive. Through the use of a grammar, a design state is progressively refined, usually by the construction of objects. A CAD system typically documents modifications to the design imposed by external means (the user's design decisions). In addition, grammars can support the emergence of unexpected objects during the design process (Stiny, 1994), which enable the designer to explore an unpredicted design path.

One area in which traditional CAD tools have the edge is in their interfaces. CAD systems in general require different types of interfaces to grammar based tools. CAD system interfaces have become quite sophisticated, evolving over many years of development and use. Grammar based design tools, however, are in their infancy. The user interactions currently available in such tools tend to be rather limited in scope. This may be due in part to the difficulty of handling the unexpected nature of emergent features, and a lack of understanding how grammars relate to the design process. Further research on interactions in grammar systems could bridge this gap between CAD system and grammar based interfaces.

In spite of the fact that there exists a large body of research in design grammars, there has been little success in developing useful computerised design grammar systems. This can be seen as due to several factors in the complexity of the object representations used in the systems (Piazzalunga & Fitzhorn, 1998). Because of this complexity, current grammar systems tend to be restrictive in their operations and provide the user with little choice in how to interact with and control the workings of the system. Many different methods of interaction have been used. Only recently have researchers seriously considered the user interaction issues with shape grammar based systems, now that the computer tools can support grammar based systems to some extent. However, no formal categorisation of these interactions has ever been developed.

A model of the potential user interaction and control of grammar based systems would be of

great benefit. Such a model, as a theoretical representation, may be manipulated independently of the actual system it represents. A model also serves as a predictor of a system's behaviour. Thus, the model can serve as a template for prototyping interfaces for grammar based systems. This can lead to the development of better interfaces to grammar based design tools and enhance their utility.

The remainder of the paper is structured as follows: previous developments in grammar based design systems are described. The components of a grammar are introduced and their utilisation in design generation is described. A user interaction model is formulated, followed by a discussion of areas for further investigation. An appendix describes interaction features of the existing grammar implementations studied for this project.

Previous work

There has been a significant amount of research in grammar based design, most significantly with the work in shape grammars (Stiny, 1980). Due to the complexity of computations with the object representations used in such grammars, much of this research has focused with the representation of shape and the application of rules. The user interaction with the system has generally been a secondary consideration.

The characterisation by Gips and Stiny (1980) of the underlying mechanisms of production provides a useful analogy and starting point for an analysis of the parts of such a system that could involve user interaction. While their analysis dealt with the components of such systems (e.g., type of vocabularies, productions, interpretive mechanism), the analysis here focuses on areas of user interaction and control.

User interaction paradigms in previously developed grammar based systems have been varied. These include

• minimal user interaction, using instead optimisation techniques to select optimal or satisficing designs (Schmidt & Cagan, 1997);

• tight control over rule application (Chase, 1989);

• computer generation and user selection (Heisserman, 1994; Tapia, 1999).

Other related work in implementations of generative design systems includes *DiscoverForm* (Carlson & Woodbury, 1994), *Tartan Worlds* (Woodbury, Radford, Taplin, & Coppins, 1992) and *TopDown* (Mitchell, Liggett, & Tan, 1990). Each has a different mode of user interaction. Recent research by Tapia (1999) begins to address the issue of user interaction in a systematic way with a detailed discussion of the design rationale behind a particular user interface to a shape grammar system, treating these as issues of presentation and selection.

Development of the interaction model should also account for the possibility that a grammar may be used as only part of the design process. This sense of contingency, in which the designer makes a conscious choice when to use a grammar, has been researched by Bruton (1997). In addition, a hermeneutical position, involving rearticulation, reinterpretation and re-engagement, can be useful in understanding how grammars fit into the design process (Woodbury, 1993).

General issues concerning the computer implementation of shape grammars are discussed by Gips (1999), who provides a comprehensive listing of shape grammar implementations. Systems examined for this project are described in the Appendix.

Grammars and derivations

In this section the various components and the operation of grammars are introduced. It should be noted that most references to grammars in this paper refer to design grammars in general. However, the examples will reference shape grammars, which are well documented in the design literature (e.g., Knight, 1994) and lend themselves to a variety of interaction techniques. Other kinds of grammars are similar in their types of basic components, but may be more limited in the types of interaction feasible.

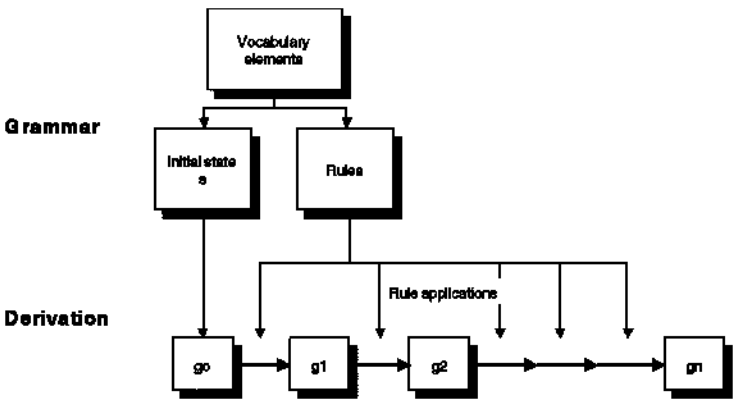

Figure 1. Elements of a grammar and derivation

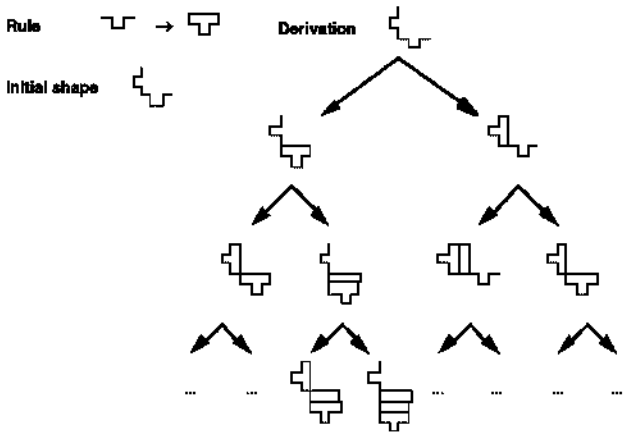

Figure 2. Sample grammar and derivation tree (after Chase, 1989)

A grammar typically consists of (Figure 1):

• a vocabulary or vocabularies of objects

• rules of the form A --> B, where A and B are vocabulary elements

• an initial state s, generally consisting of an element or elements from the vocabulary

A rule A --> B applies to a state g whenever there is a matching condition that maps A to g in some predefined manner. The result of a rule application is g - t(A) + t(B), meaning: replace the matched (transformed) portion of A in g with a similarly transformed portion of B.

A derivation consists of a sequence of states g_0, g_1, ..., g_n, where g_0 is the initial state s and g_i is produced from g_{i-1} by applying a rule of the grammar. Figure 2 shows an example of a shape grammar (consisting of lines, with the matching condition being Euclidean transformation to a part of the shape g) and a derivation tree. At any stage of a derivation there may be a choice of rules and matching conditions that may be invoked, thus providing the potential for user interaction with the system. As is evident from the derivation tree, the result of rule applications generates a design space that, depending upon the problem, may require some method of finding a desired or optimal solution. In these cases, the interaction issues become ones of controlled search through the space.

A model of user interaction

The aspects of user interaction to be represented in a model of user interaction include:

• the different stages in the design process in which the grammar is involved;

• the various entities that interact with the grammar;

• the possibilities for user control.

Each is described below.

Stages and entities

The complete process of generating a design with a grammar involves two main stages (Figure 3):

• development of the grammar, i.e. constructing the vocabulary, rules and initial state;

• application of the grammar, i.e. a series of rule

invocations (derivation).

Each of these stages involves the manipulation of the grammar by some entity. Three possible entities are considered:

• the grammar developer, typically someone familiar with the technical aspects of grammars and the computational environment in which the grammar will be implemented;

• the end user, i.e. the designer who will generate a language of designs by invoking grammar rules;

• the computer system, which could potentially handle any or all aspects of grammar development and application.

Each stage can be associated with an interaction paradigm involving one of these entities. Their combination provides the foundation of an interaction model.

Grammar development

Given their precise nature, grammars are generally constructed by hand, i.e. by humans. This is in part a result of the problems in predicting the language of designs that a grammar will generate (Knight, 1998). Therefore, the construction of

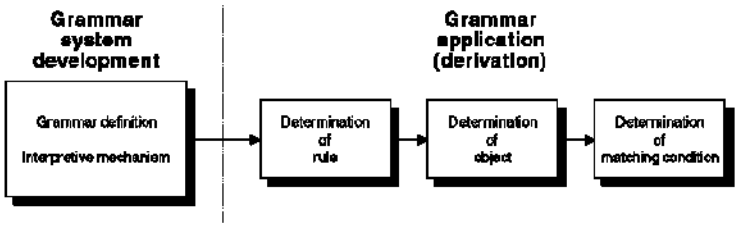

Figure 3. Steps in the development and application of a grammar

a grammar often tends to be a 'generate and test' cycle, in which the grammar is tested and then modified. In this manner, the grammar is refined so as to produce a specific language of designs, i.e. one that incorporates all designs of interest but eliminates those falling outside the specified criteria. Research in automatic grammar generation has yet to make any impact in design domains due to the difficult nature of the problem. Grammar adaptation and transformation to produce new languages of designs i.e., new styles (Knight, 1994) has been more successful and appears to be amenable to computer implementation.

In the past, the construction of design grammars in a computer implementation was rather difficult, generally involving the coding of grammar rules in a textual form that the computer could interpret (for example, see Chase, 1989). More recent systems allow the user to specify rule shapes by using computerised drawing tools, thus allowing the end user to have a greater hand in the development of a grammar (Tapia, 1999).

Grammar application

The application of a grammar, i.e. the derivation of a design in the language of the grammar, can be accomplished by the end user (designer) or by the computer. For each rule invocation, there are several actions that occur:

• A rule to apply is selected from the rule set;

• A part of the design state (e.g., a subshape) is selected to be acted upon;

• A matching condition (transformation of rule shape to design shape) is determined.

There are often interdependencies among each of 1.-3., i.e. the determination of one may be dependent upon the other selections (Table 1). For example, the selection of a rule and a subshape will, in most cases, limit the possible matching conditions (but not necessarily to one, thus allowing additional choice of rule application). On the other hand, by selecting a rule and a specific matching condition, one automatically fixes the part of the design to which the rule can be applied (if at all), thus limiting the result of the rule application to one possibility.

As one can imagine, the possibilities for different interaction techniques can be quite varied. Given the three scenarios shown in Table 1, and

Choose rule & **Choose** subshape	→	Set of matching conditions determined	→	**Choose** one matching condition	→	Apply rule
Choose subshape & **Choose** matching condition	→	Set of rules determined	→	**Choose** one rule	→	Apply rule
Choose rule & **Choose** matching condition	→	Single subshape determined	→			Apply rule

Table 1. Examples of actions involved in rule invocation. Bold text indicates where the potential for user interaction exists, as a choice must be made.

possibility of either user or computer choice of rules, subshapes, or matching conditions, there are 20 different possibilities for user interaction in the grammar application stage, ranging from total user control (Figure 4, Scenarios 1 and 2) to total computer control (Scenarios 5 and 6). If the grammar development stage is also considered, and a distinction is made between developer and end user, the number of possible scenarios increases markedly. This is discussed further in the following section.

Control of the grammar

An appropriate control scenario for a given grammar application is constructed by mapping entities to stages. In order to do so, a number of issues must first be resolved. Listed below are some general questions worth considering.

Grammar development

Who (developer, end user, computer) should have control over the following, and how much control should they have?

• creation of the underlying object representation;

• creation of the control mechanism; and

• creation of rules.

As mentioned above, construction of a grammar's components can be extremely difficult to automate. However, given a library of possible object representations and control mechanisms, a sophisticated computer system may be able to employ some heuristic to choose the appropriate paradigms for a given grammar application. Rule creation could possibly occur through a learning mechanism such as a genetic algorithm (Gero & Ding, 1997).

Grammar application

Among the issues to be considered when invoking the grammar for design generation are:

• How does the user operate the system?

• What kinds of choices are offered the user in terms of parameter settings, selection of rules and rule application?

• How does the system present choices to the user?

• Can this range of control (from none to full) be modified by the user?

• Can the use of the grammar be combined or integrated with other techniques, including cognitive approaches?

An in depth discussion of these issues is beyond the scope of this paper. Some of these have been explored by others. Among these are ways

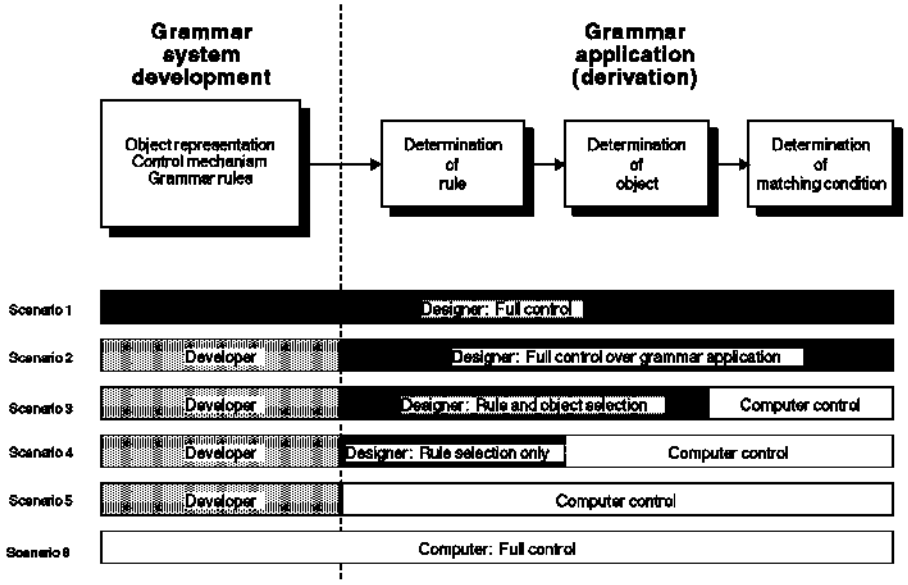

Figure 4. Some of the possible user control scenarios for the development and application of a grammar

to increase the computational tractability or ease of use of the grammar by placing restrictions upon its operation, e.g.

• restricting the design area or the matching conditions possible for rule application (Tapia, 1999);

• classifying applicable rules according to user's role, design task, and design stage (Heisserman & Callahan, 1996);

• employing direct manipulation of grammar shapes to explore designs in a continuous space (Datta & Woodbury, 1998).

Once the answers to the questions above are known, the entity-stage mapping can be determined and an interaction model constructed. Possible interaction scenarios might include (Figure 4):

full control The user is responsible for each rule invocation as well as grammar development. This is analogous to a non-computerised system (Scenario 1).

partial control There is user selection during some aspect of rule invocation, e.g., a rule or object to which the rule is applied (Scenarios 2-4).

no control The grammar is predefined, possibly computer-generated. The system automatically generates designs without user intervention (Scenarios 5 and 6).

Figure 5 illustrates in more detail an example of two possible modes of interaction with a shape grammar system (Chase, 1989): (a) a 'manual' mode (analogous to Scenario 2); and (b) a 'semiautomatic' mode (analogous to Scenario 4). The points of user interaction through a derivation of a design are shown in bold boxes. In this example, the sequence tends to be a linear one, with no choice in terms of the types of actions the user may perform at any time. As designers tend to work in a nonlinear manner, i.e., switching between tasks at any one time, non-modal interaction techniques would be

preferable. Tapia's shape grammar system (1999) provides such an interface, allowing the user to shift focus between rule development and rule application.

The illustrated interaction scenarios also raise the question: in terms of grammar development and usage, does there need to be a strict progression of control from human to computer in grammar development and application? The example scenarios seem to indicate that there is a logical progression from human control to computer control, i.e. that there must be human input before a computer can take control. Is this necessarily true? A model of a system in which human/computer control is more interleaved during development through one rule application deserves further investigation.

Discussion

Possibilities for further investigation

The model developed here has been based on examination of a few grammar based systems and some intuitive concepts of interaction paradigms. As can be seen from the few examples described here, the combination of stages and entities leads to a wide variety of interaction paradigms for grammar based design systems. There are a number of areas worth exploring in the further development of an interaction model. These include:

• utilisation of a more rigorous approach to model development by applying theories of user interface design (Thimbleby, 1990);

• a closer examination of design processes and the parties involved. For example, different domain experts participate in design at various stages throughout the process. How can these differences be incorporated into a formal interaction model?

An issue not discussed here is how grammars can actually be utilised in the design process. A reflective approach, in which the designer moves between grammar and non-grammar based paradigms (Bruton, 1997; Woodbury, 1993) should be further explored. This could include protocol studies of designers' use of

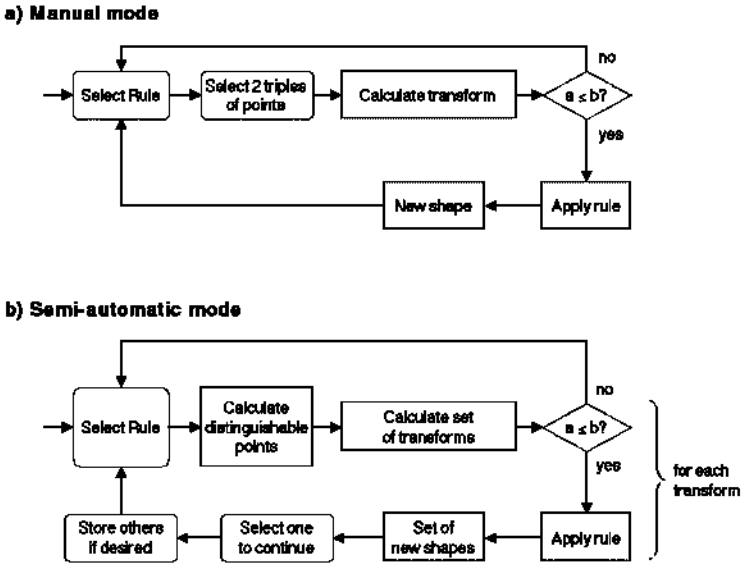

Figure 5. Two modes of user interaction (after Chase, 1989)

grammars. This also raises the issue of purpose built systems: a grammar based design system developed for routine design may have a different interface than one developed for innovative design. The scale of the design problem will also have some impact upon the interaction techniques utilised. For example, compare the interaction techniques of GENESIS (see Appendix) with systems developed for smaller design problems.

Another area mentioned briefly, but not explored here, is that of emergence (Stiny, 1994). A powerful design system will be able to account for the unexpected nature of emergent features. Thus, a robust interaction model will support this and allow the user to control changes of a) design representation and b) design direction. This may represent a change of focus of the user and a change in the mode of interaction, requiring an active interaction environment.

Conclusions

By presenting a method for modelling user interaction in grammar based design systems, a

wider range of interaction possibilities for such systems is elucidated. This can lead to better understanding of the possible utilisation of such tools in the design process. In addition, the development of a formal model can lead to the introduction of guidelines for user interaction, and possible uniformity of interfaces for such systems. The categorisation of the interfaces of existing grammar based systems (described in the Appendix) provides empirical evidence into how interaction techniques might be improved, and also suggests possible refinements to the formal model. The improvement of interaction techniques for grammar based design tools should also strengthen the case for their consideration as preferred alternatives to the more passive CAD tools currently available.

References

Agarwal, M., & Cagan, J. (1998). A Blend of Different Tastes: The Language of Coffee Makers. *Environment and Planning B: Planning and Design*, 25(2), 205-226.

Bruton, D. (1997). *A Contingent Sense of Grammar*. Ph.D dissertation, University of Adelaide.

Carlson, C., & Woodbury, R. (1994). Hands-on exploration of recursive patterns. *Languages of Design*, 2, 121-142.

Chase, S.C. (1989). Shapes and Shape Grammars: From Mathematical Model to Computer Implementation. *Environment and Planning B: Planning and Design*, 16(2), 215-242.

Datta, S., & Woodbury, R.F. (1998). Reducing Semantic Distance in Generative Systems: A Massing Example. In T. Seebohm & S.V. Wyk (Eds.), *Digital Design Studios: Do Computers Make a Difference?, proceedings of ACADIA '98*, Québec City, Canada, October 22-25, 1998 (pp. 164-171): Association for Computer-Aided Design in Architecture.

Gero, J.S., & Ding, L. (1997). Exploring style emergence in architectural design. In Y.-T. Liu, J.-H. Tsou, & J.-H. Hou (Eds.), *CAADRIA '97* (pp. 287-296). Taipei, Taiwan: HU's Publisher.

Gips, J. (1999). Computer Implementation of Shape Grammars. Cambridge, Mass.: NSF/MIT Workshop on Shape Computation.

Gips, J., & Stiny, G. (1980). Production systems and grammars: a uniform characterization. *Environment and Planning* B, 7, 399-408.

Heisserman, J. (1994). Generative Geometric Design. *IEEE Computer Graphics and Applications*, 14(2), 37-45.

Heisserman, J., & Callahan, S. (1996). Interactive Grammatical Design. In K. Brown & J. Heisserman (Eds.), *Workshop Notes, Grammatical Design, Fourth International Conference on Artificial Intelligence in Design*, Stanford University, June 22, 1996.

Knight, T.W. (1994). *Transformations in Design: A Formal Approach to Stylistic Change and Innovation* in the Visual Arts. Cambridge, England: Cambridge University Press.

Knight, T.W. (1998). Designing a shape grammar: Problems of predictability. In J.S. Gero & F. Sudweeks (Eds.), *Artificial Intelligence in Design '98* (pp. 499-516). Dordrecht: Kluwer Academic Publishers.

Krishnamurti, R. (1982). SGI User Manual. Milton Keynes: The Open University.

Mitchell, W.J., Liggett, R.S., & Tan, M. (1990). Top-Down Knowledge-Based Design. In M. McCullough, W.J. Mitchell, & P. Purcell (Eds.), *The Electronic Design Studio*. Cambridge, MA: MIT Press.

Piazzalunga, U., & Fitzhorn, P. (1998). Note on a three-dimensional shape grammar interpreter. *Environment and Planning B: Planning and Design*, 25, 11-30.

Schmidt, L.C., & Cagan, J. (1997). GGREADA: A Graph Grammar-Based Machine Design Algorithm. *Research in Engineering Design*, 9, 195-213.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B*, 7, 343-351.

Stiny, G. (1994). Shape rules: closure, continuity,

and emergence. *Environment and Planning B: Planning and Design*, 21, s49-s78.

Tapia, M. (1999). A visual implementation of a shape grammar system. *Environment and Planning B: Planning and Design*, 26(1), 59-73.

Thimbleby, H. (1990). *User Interface Design*. New York: ACM Press.

Wang, Y., & Duarte, J.P. (1998). Synthesizing 3D forms: Basic Grammars and Rapid Prototyping. In J. Heisserman & R. Klein (Eds.), *Workshop Notes, Generative Design, Fifth International Conference on Artificial Intelligence in Design*, Instituto Superior Téchnico, Lisboa, Portugal, July 18, 1998.

Woodbury,R.(1993).Grammatical Hermeneutics. *Architectural Science Review*, 36(2), 53-64.

Woodbury, R.F., Radford, A.D., Taplin, P.N., & Coppins, S.A. (1992). Tartan Worlds: A Generative Symbol Grammar System. In K.M. Kensek & D. Noble (Eds.), *Computer Supported Design in Architecture: Mission, Method, Madness, proceedings of ACADIA '92*, (pp. 211-220): Association for Computer-Aided Design in Architecture.

Appendix

Of the 18 shape grammar systems listed by Gips (1999), 7 were examined for this project. The author has either obtained a working copy of each system, or determined its interaction features from a system manual or a published paper. Table 2 shows some of the interaction features for each system, with an explanation following. Figures 6-8 illustrate some of the user interfaces.

SGI (Krishnamurti, 1982) was the first generalised implementation of a shape grammar system, allowing the user to define rules and apply them to generate 2D shapes.

Chase's 2D interpreter (1989) was implemented in Prolog on a Macintosh, and required the user to specify both rule selection and application, with the system executing the rule according to the specified transformation.

GENESIS (Heisserman, 1994), operating on 3D solid shapes, was the first grammar based system to attempt to handle realistically scaled design problems. Development of GENESIS continues in the domain of aircraft design (Heisserman & Callahan, 1996).

GEdit (Tapia, 1999), implemented in Macintosh Common Lisp, is a general 2D shape grammar interpreter, with a unique non-modal interface. Tapia was the first to consider the issues of user interface, i.e. presentation and selection of shapes during a derivation.

Piazzalunga and Fitzhorn's system (1998) uses a standard ACIS solid modeller with the Scheme language to produce a 3D shape grammar interpreter in which the user specifies both rule selection and rule application (similar to Chase's).

Shaper (Wang & Duarte, 1998) is a Java based application that allows easy exploration of basic (restricted) 3D grammars, in which the user can modify the parameters of the spatial relations, but not the basic form or application of the rules.

	SGI	Shape grammar system	GENESIS (CMU)	Shape grammar interpreter	Coffee grammar maker	Shaper	GEdit
Reference	(Krishnamurti 1982)	(Chase 1989)	(Heisserman, 1994)	(Piazzalunga & Fitzhorn, 1998)	(Agarwal & Cagan, 1998)	(Wang & Duarte, 1998)	(Tapia, 1999)
2D/3D	2D	2D	3D	3D	2D/3D	3D	2D
Modal	Y	Y	Y	Y	Y	Y	N
Selection of control mode	Y	N	Y	N	N	Y	Y
Rule definition	U	D	D	?	D	U	U
Rule selection	U	U	S*	U	U/S	S (fixed rule application)	U
Object selection	S*	U	S*	U	S	S (only 1 possibility)	S
Results	1	1	1 at a time	1	1	1	All
Backtrack	Y	Y	N	future?	Y	N	Y
Emergence	Y	Y	N	N	?	N	Y
Notes					fixed grammar; generation phase only	basic grammar, limited rule types; definition & generation phases	shape definition external; rule definition internal; generation phase shows all possibilities

*Table 2. Interaction features of some grammar based design systems, adapted from (Gips, 1999). Items with a * are elaborated in the explanation of table entries.*

The coffee maker grammar (Agarwal & Cagan, 1998) is a Java based application running via a web browser, using a fixed, parametric grammar. The grammar works in stages; at any point the user is given a selection of rules to choose. Many rules require the instantiation of parameters through dialogue boxes. Interestingly, the interface makes the grammatical nature of the application less obvious than the other applications examined here. This raises the question of the significance of a grammatical paradigm to the end user, i.e. how important is it that the designer is cognisant of the strict application of a fixed set of rules during the design process?

All of the systems described above do indeed contribute to the development of grammar based design systems. Those that stand out as major contributions, in the author's opinion, are:

- SGI, in being the first of its kind;
- GENESIS, in attempting to handle large scale design problems;
- GEdit, in its consideration of user interface;
- Coffee maker grammar, in its use of a function based grammar for a real design problem.

Explanation of table entries

Modal: is the user restricted to specific types of actions at any given time? Only GEdit provides

a truly non-modal interface.

Selection of control mode: can the end user switch between user control and system control?

Rule definition: end user (U) or system developer (D). D indicates that rule definition may be technically difficult for an end user to attempt.

Rule selection: end user (U), system developer (D) or system (S). GENESIS selects an applicable rule and asks the user whether to apply it.

Object selection: end user (U) or system (S). SGI and GENESIS select an object for rule application, allowing the user to select or reject it.

Results: indicates how many possible results are presented during rule application. Most present just one. GENESIS allows the user to see a result, select or reject it, then presenting an alternative. GEdit displays all possible results.

Backtrack: does the system allow undoing rule applications and returning to a previous state of a derivation?

Interface examples

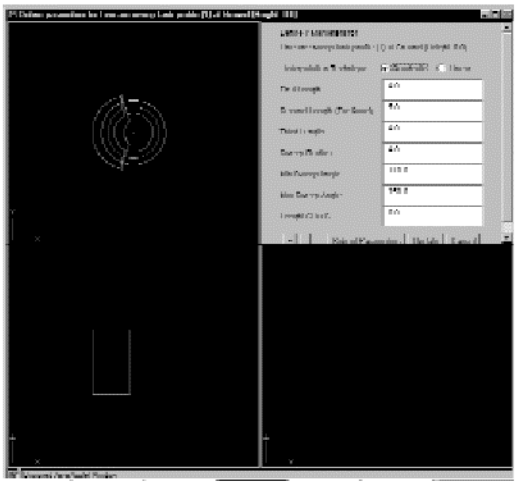

Figure 6. Coffee grammar program, showing parameter input window and three windows depicting orthographic views of the design.

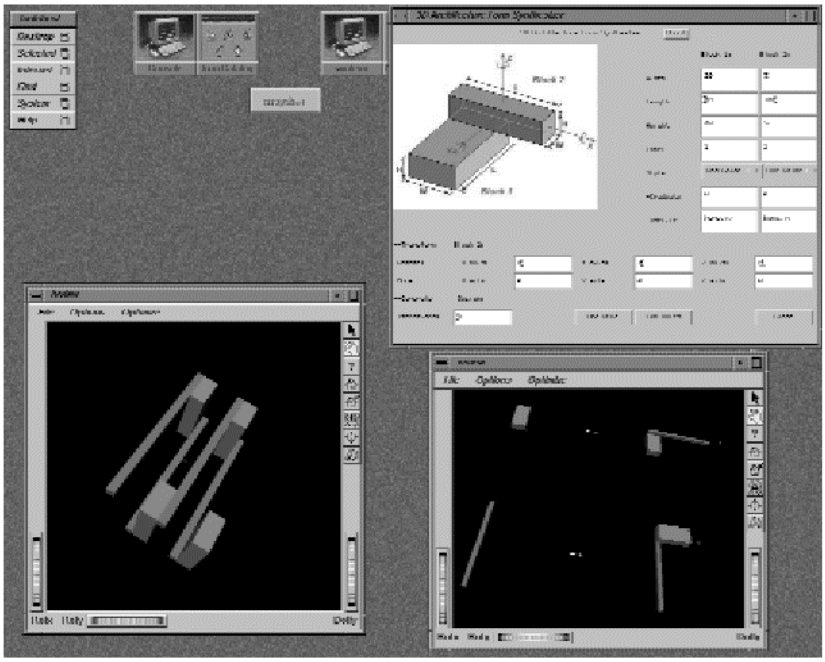

Figure 7. Shaper program. Shown are a) control window for specification of spatial relation and rule application; b) result window; c) rule viewing window (two rules shown).

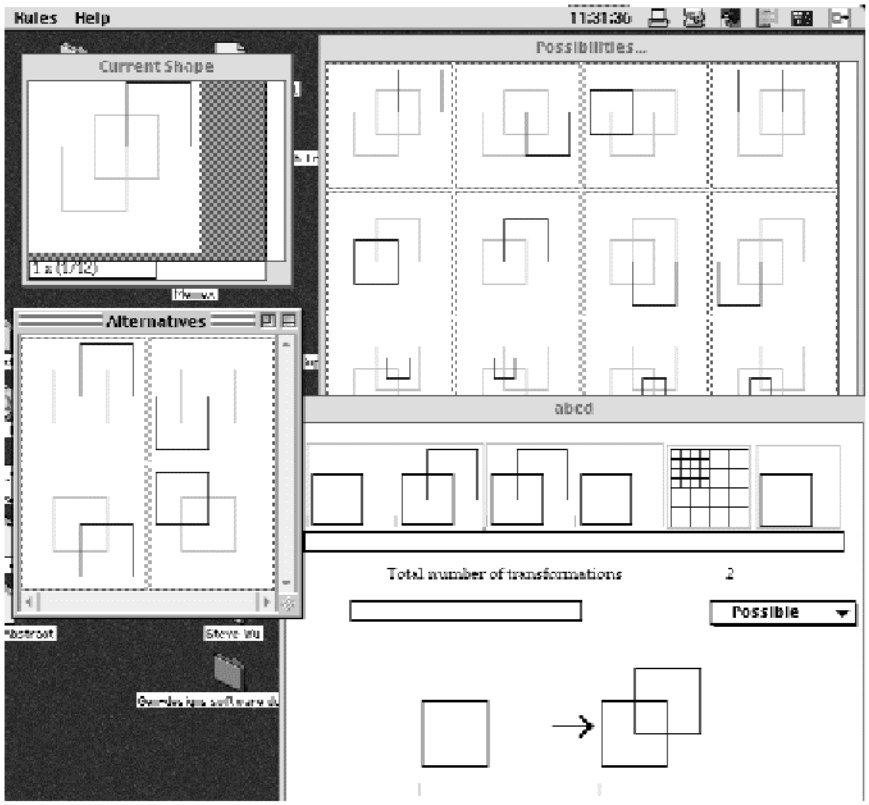

Figure 8. GEdit program. Windows shown include a) rule window (abcd); b) current shape in the derivation; c) alternate ways in which the rule can apply; d) possible results of a rule application.