Formal Engineering Design Synthesis

Edited by

ERIK K. ANTONSSON
California Institute of Technology

JONATHAN CAGAN
Carnegie Mellon University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
10 Stamford Road, Oakleigh, VIC 3166, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2001

Printed in the United States of America

Typefaces Times Ten 10/13 pt. and Helvetica Neue *System* LaTeX 2ε [TB]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Formal engineering design synthesis / edited by Erik K. Antonsson,
Jonathan Cagan.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-79247-9
1. Engineering design. I. Antonsson, Erik K. II. Cagan, Jonathan, 1961–
TA174 .F67 2001
620′.0042 – dc21 2001025164

ISBN 0 521 79247 9 hardback

Contents

Contributing Authors

Erik K. Antonsson is Professor and Executive Officer (Dept. Chair) of Mechanical Engineering at the California Institute of Technology, Pasadena, CA 91125-4400, where he founded and supervises the Engineering Design Research Laboratory; he has been a member of the faculty since 1984.

Jonathan Cagan is Professor of Mechanical Engineering at Carnegie Mellon University, Pittsburgh, PA 15213, and Director of the Mechanical Engineering Computational Design Laboratory; he holds appointments in the Schools of Design, Computer Science and Biomedical and Health Engineering.

Scott D. Barnicki is a Research Associate with the Eastman Chemical Company, P. O. Box 1972, Kingsport, TN 37662-5150.

Georges G. E. Gielen is a Professor in the Department of Electrical Engineering at the Katholieke Universiteit Leuven, Kasteelpark Arenbert 10, 3001 Leuven, Belgium, working in analog and mixed-signal integrated circuit design and design automation.

James L. Greer is an Assistant Professor in the Department of Engineering Mechanics at The United States Air Force Academy, 2354 Fairchild Dr., Suite 6H2, Colorado Springs, CO 80840-2944.

Leo Joskowicz is an Associate Professor at the School of Computer Science and Engineering, Ross Building, Room 223, The Hebrew University of Jerusalem Givat Ram, Jerusalem 91904, Israel.

Kenneth Kotovsky is a Professor in the Psychology Department at Carnegie Mellon University, Pittsburgh, PA 15213.

Cin-Young Lee is a Mechanical Engineering Ph.D. candidate in the Engineering Design Research Laboratory at the California Institute of Technology, Pasadena, CA 91125.

Lin Ma is a Mechanical Engineering Ph.D. candidate in the Engineering Design Research Laboratory at the California Institute of Technology, Pasadena, CA 91125.

Lionel March is a Professor in Design and Computation at the University of California, 1200 Dickson, Box 951456, Los Angeles, CA 90095-1456.

J. Michael McCarthy is a Professor in the Department of Mechanical Engineering at the University of California, 4200 Engineering Gateway, Irvine, CA 92697-3975.

William J. Mitchell is a Professor of Architecture and Media Arts and Sciences and the Dean of the School of Architecture and Planning at the Massachusetts Institute of Technology, 77 Massachusetts Ave., Room 7-231, Cambridge, MA 02139.

Panos Y. Papalambros is the Donald C. Graham Professor of Engineering and Professor of Mechanical Engineering at The University of Michigan, 2266 G. G. Brown Lab, Ann Arbor, MI 48109-2125.

Rob A. Rutenbar is Professor of Electrical and Computer Engineering and (by courtesy) Professor of Computer Science at Carnegie Mellon University, Pittsburgh, PA 15213.

Kristina Shea is University Lecturer in the Department of Engineering at the University of Cambridge, Trumpington St., Cambridge CB2 1PZ, U.K.

Jeffrey J. Siirola is a Technology Fellow at the Eastman Chemical Company, P. O. Box 1972, Kingsport, TN 37662-5150.

Herbert A. Simon* is a Professor in the Departments of Computer Science and Psychology at Carnegie Mellon University, Pittsburgh, PA 15213.

Thomas F. Stahovich is an Associate Professor in the Department of Mechanical Engineering at Carnegie Mellon University, Pittsburgh, PA 15213.

George Stiny is a Professor of Design and Computation in the Department of Architecture at the Massachusetts Institute of Technology, 77 Massachusetts Ave., Room 10-431, Cambridge, MA 02139.

Allen C. Ward is an independent consultant on lean development processes at Ward Synthesis, Inc., 15 1/2 East Liberty St., Ann Arbor, MI 48104.

Kristin L. Wood is a Professor of Mechanical Engineering in the Mechanical Systems and Design Division of the Department of Mechanical Engineering at The University of Texas at Austin, ETC 5.140, Austin, TX 78712-1063.

* We note with sorrow that Prof. Simon passed away while this books was in press.

CHAPTER ONE

Vitruvius Redux

Formalized Design Synthesis in Architecture

William J. Mitchell

PARTICULATE COMPOSITION

A quarter of a century ago I published a paper titled "Vitruvius Computatus" (Mitchell, 1975). In it I argued that, if computer systems achieved the capacity to execute nontrivial architectural design tasks, we could "expect the designs which emerge from them to be characterized by particular stylistic traits." I went on to suggest, more specifically, "If computer systems become architects they may be academic classicists, heirs to Durand and Guadet." Sadly, I turned out to be right.

Jean N. L. Durand and Julien Guadet were great French architectural pedagogues, and authors of highly influential texts that were used by generations of students.[1] They provided models to copy, and they promulgated a view of architectural composition that was essentially particulate and combinatorial. Within the tradition that Durand and Guadet established and formalized, designers relied heavily upon abstract ordering devices such as grids and axes, and they would typically begin a project by exploring various combinations of these devices to produce an organizing skeleton of construction lines (Figure 1.1). Guided by this skeleton, they would then consider alternative ways to arrange the major rooms and circulation spaces. Finally, they would develop the design by deploying elements from an established vocabulary of construction elements – columns, entablatures, doors, windows, and so on. As Durand's famous plates illustrated (Figure 1.2), it was a recursive process of top-down substitution.

Thus, learning to design was much like learning Latin prose composition. There was a well-defined vocabulary of discrete construction elements, there were rules of syntax expressed by allowable combinations of grids and axes (the graphic equivalent of sentence schemata found in grammar textbooks), and you could produce acceptable designs by arranging the construction elements according to these rules. Many of Durand's plates were devoted to illustrating alternative "horizontal combinations" (plans) and "vertical combinations" (elevations) that could be produced in

[1] See, in particular, Durand's *Précis des Leçons d'Architecture* (Durand, 1819), his *Partie Graphique des Cours d'Architecture* (Durand, 1821), and Guadet's *Éléments et Théories de l'Architecture* (Guadet, 1902).

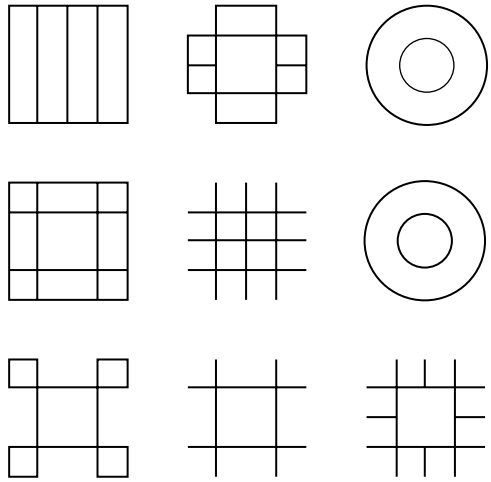


Figure 1.1. Skeletons of construction lines as starting points for plan compositions, from Durand's *Précis* (1819).

this way. In addition to generating a design according to Durand's rules, you could run the process in reverse to parse a completed design into a hierarchy of definite elements and subsystems, just as you could parse a Latin sentence into phrases and parts of speech.

The great figures of early architectural modernism mostly represented themselves publicly as anticlassicists, but they did not forget the intellectual legacy of Durand and Guadet. As Reyner Banham noted, Guadet's *Éléments et Théories de l'Architecture* (1902) "formed the mental climate in which perhaps half the architects of the twentieth century grew up." The compositional strategy that they followed

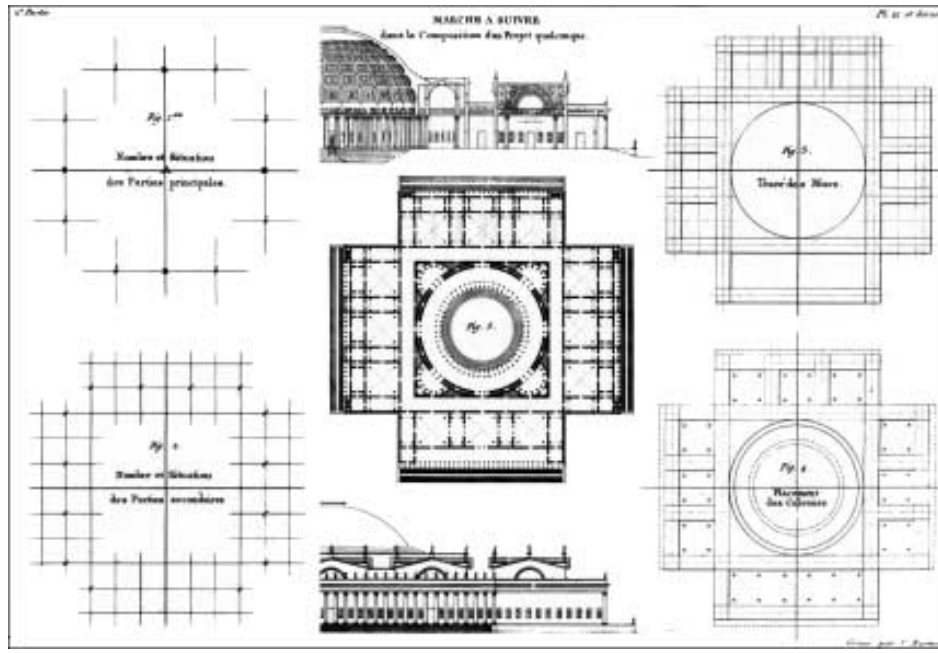


Figure 1.2. Design as a process of top-down substitution, from Durand's *Précis* (1819).

could be characterized as follows:

> The approach is particulate; small structural and functional members (elements of architecture) are assembled to make functional volumes, and these (elements of composition) are assembled to make whole buildings. To do this is to compose in the literal and derivational sense of the word, to put together.
>
> R. Banham, *Theory and Design in the First Machine Age* (1967)

The new conditions of industrial mass production clearly strengthened this idea of composing a building from discrete standard parts. The difference, now, was that the drive to achieve economies of scale (rather than the pedagogical strategy of publishing models to copy) motivated standardization and repetition. The extreme response to these conditions was the creation of industrialized component building systems that provided complete, closed "kits of parts" for assembling entire buildings. At a smaller scale, construction toys such as Froebel and Lego blocks reproduced the idea of particulate composition at a smaller scale. Proponents of these systems frequently illustrated their capabilities by producing exhaustive enumerations of possible combinations of elements.

THE BEGINNINGS OF ARCHITECTURAL CAD

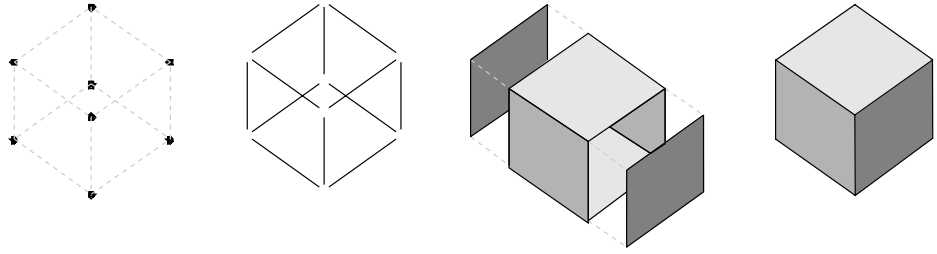
Not surprisingly, when the first computer-aided design (CAD) systems appeared, they reified these well-established design traditions.[2] They allowed drawings to be organized in "layers," which reflected both the earlier practice of drafting on overlaid sheets of translucent paper and Durand's notion of successive layers of abstraction. They provided grids, construction lines, and "snap" operations for positioning graphic elements in relation to the established skeleton of a composition. They also relied heavily upon the operations of selecting standard graphic elements, then copying, translating, and rotating them to assemble compositions from repeated instances.

At a deeper level – that of their underlying data structures – these systems also assumed definite, discrete elements. The basic geometric units were points, specified by their Cartesian coordinates. Lines were then described by their bounding points, polygons by their bounding lines, and closed solids by their bounding polygons.[3] More complex graphic constructions were treated as sets of these basic geometric entities, and nongeometric properties (color, mass, material properties, cost information, etc.) might be associated with such groupings. Subshapes were simply subsets, and complete drawings could unambiguously be parsed into hierarchies of subshapes and geometric entities – all the way down to the elementary points. There were many variants on this basic scheme – some of them very clever – but they were all grounded upon the idea of sets of geometric entities encoded as relations on sets of points (Figure 1.3).

Because the usual goal of early CAD systems was efficiency in the production of construction documents, many went a step further and provided higher-level vocabularies of architectural elements such as standard walls, doors, windows,

[2] For details of early architectural CAD systems, see Mitchell (1977).

[3] This strategy is grounded in the pioneering work of Steven Coons and Ivan Sutherland. See Stiny, Chapter 2, for a critical discussion of the theoretical foundations.



8 vertices 12 lines 6 surfaces 1 solid

Figure 1.3. Selectable geometric entities in a typical CAD system.

columns, desks and chairs, mechanical and electrical equipment items, and so on (Figure 1.4). These became the elements that an architect could quickly select, transform, and assemble in a process that directly (though frequently unknowingly and uncritically) recapitulated Durand. In other words, these systems assumed and enforced a conventionalized architectural ontology. This was reasonable if you were prepared to accept the ontology, but not if you wanted to stretch the boundaries of architectural discourse. You were out of luck if you wanted to explore an architecture of free-form blobs, or of nonrepeating, nonstandard parts, for example.

At an implementation level, however, there were many practical advantages to this approach. It was conceptually simple, and it supported the creation of elegant and efficient graphics pipelines that took advantage of increasing computer resources. It lent itself to some useful elaborations, such as parametric CAD. It also benefited from new approaches to software engineering, such as object-oriented programming.

In contrast, the limitations were obvious enough from the beginning – though, as it turned out, little reflected upon or heeded in what became the headlong post-PC rush to commercialize and market standard computer graphics technology. In "Vitruvius Computatus" I warned:

> Is this continuity of the academic classical tradition of elementary composition in a rather unexpected way merely a historical curiosity of little practical consequence? I suggest not. It indicates that computer systems cannot be regarded as formally neutral tools in the design process. On the contrary, their use can be expected to demand acquiescence to particular formal disciplines. This implies several things. Firstly it becomes clear that the design and development of satisfactory computer-aided design

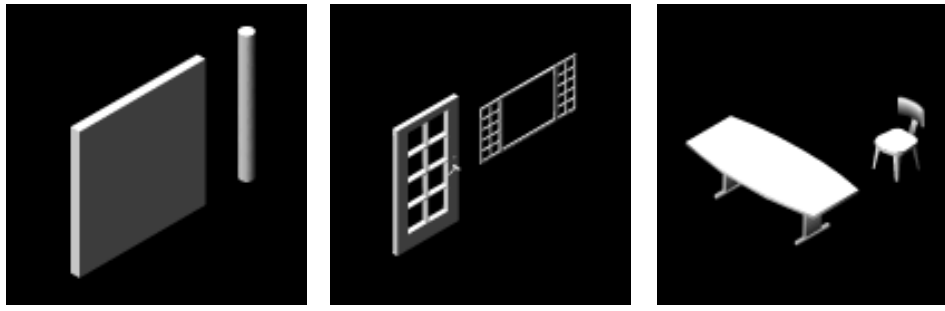


Figure 1.4. Typical vocabulary elements of an architectural CAD system.

> systems is not simply a technical task to be left to systems analysts, engineers, and programmers. It demands a high level of *architectural* skill and capacity for rigorous analysis of built form. Secondly, as architects begin increasingly to work with such systems they will need to devote as much attention to understanding the vocabulary and syntax of form implied by the data structures of those systems as the classical architects of old devoted to study of the orders.
>
> W. Mitchell (1975)

The consequences of implicitly encoding stylistic presuppositions in software eventually showed up in built work. As one prominent and sophisticated designer recently remarked to me, "If you just look at the work of many young designers, you can immediately see what sort of CAD system they are using."

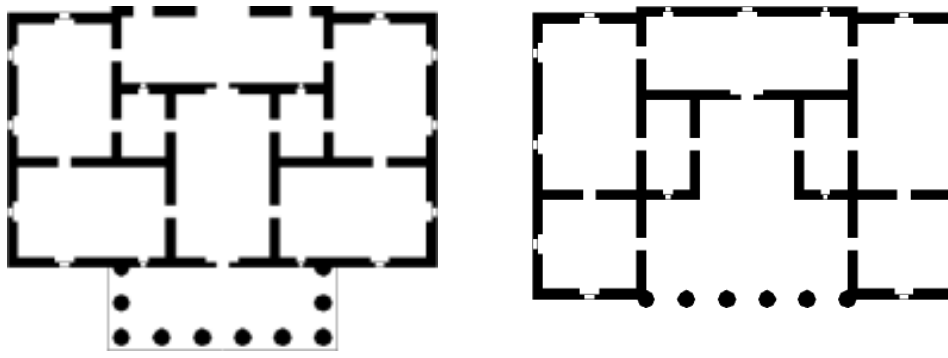
LANGUAGES OF ARCHITECTURAL FORM

Just as Durand conceived of design as a process of recursive substitution of shapes terminated by the insertion of elements from an established architectural lexicon, formal linguistics taught us to understand sentence construction as the recursive substitution of symbols terminated by the insertion of words from a written language's vocabulary. The frequently invoked analogy between classical architectural design and Latin prose composition turns out to be more than superficial!

With the development of the shape grammar formalism, Stiny and Gips (1972) made the idea of shape substitution precise, and they cast it in modern computational form. Whereas formal generative grammars (as the concept is normally understood in linguistics and computer science) generate one-dimensional strings of symbols, shape grammars generate two-dimensional or three-dimensional (or, indeed, *n*-dimensional) arrangements of shapes. They operate by recursively applying shape rewriting rules to a starting shape. A language of designs thus consists of all the shapes that can be derived in this way.

Stiny and Mitchell (1978a, 1978b) first applied shape grammars to nontrivial architectural design by writing a grammar that produced schematic villa plans in the style of the great Italian renaissance architect Andrea Palladio. In some sense, this grammar captured the essence of Palladio's style, and it allowed others to design in that style – thus functioning pedagogically much as Palladio's own *Four Books of Architecture* had been intended to do. Stiny and Mitchell demonstrated the grammar by exhaustively enumerating (at a certain level of abstraction) the plans in the language that it specified. The resulting collection of possibilities included the villa plans that were actually produced by Palladio, together with others that might have been if there had been the time and the opportunity (Figure 1.5).

Subsequently, there have been many other attempts to write grammars that produce particular classes of architectural designs and to enumerate the languages they specify. Numerous of these have been published, over the years, in the journal *Environment and Planning B*. In 1981, for example, Koning and Eizenberg (1981) demonstrated a grammar that produced convincing "prairie houses" in the style of Frank Lloyd Wright (Figure 1.6). A particularly compelling recent example is Duarte's grammar (Duarte, 2000a, 2000b) of house designs in the style of Alvaro



Villa Hollywood Villa Vine

Figure 1.5. Plans of the Villa Hollywood and the Villa Vine, two of the pseudo-Palladian villas generated by Stiny and Mitchell's grammar.

Siza's Malagueira project (Figure 1.7). Duarte's grammar is presented in the form of a Web site that allows house clients to derive designs that respond to their own particular needs and budgets.

COMBINATORIAL SEARCH

The most obvious way to implement a shape grammar (or some functionally equivalent set of generative rules expressed in a different way) is to begin with the sort of data structure common to CAD systems, as described earlier. Subshapes are thus treated as subsets of geometric elements, and rewriting rules replace one subset with another. This is all a fairly straightforward extension of the computer science tradition dealing with formal languages, automata, production systems, and the like. And it also allows architectural design problems to be assimilated into the intellectual

Figure 1.6. A prairie house in the style of Frank Lloyd Wright, generated by Koning and Eizenberg's grammar.

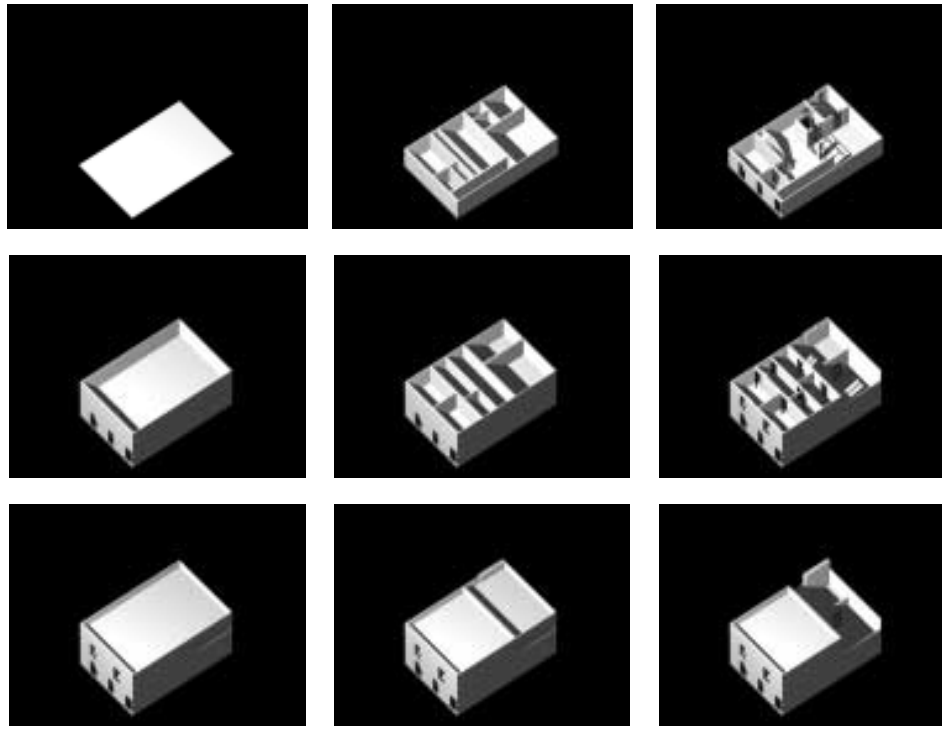


Figure 1.7. Step-by-step derivation, according to Duarte's grammar, of a Malagueira house.

traditions of AI problem-solving (as represented, in particular, by the pioneering work of Newell and Simon) and combinatorial optimization.[4]

Within this framework, design problems can be formulated as problems of searching discrete (or partially discrete) state-spaces to find designs in a language that satisfy specified criteria. (Recall Durand's "horizontal combinations" and "vertical combinations" – variants adapted to different site contexts, functional requirements, and budgets.) These criteria can be expressed, in the usual ways, in the form of constraints and objective functions. The search tasks that result are typically large scale and nastily *np* complete. Heuristic search, simulated annealing, evolutionary strategies, and other such techniques are thus employed to control the search process and produce acceptable results with a reasonable expenditure of computational resources.

This combinatorial approach had its early successes. For example, Mitchell, Steadman and Liggett (1976) produced an efficient and reliable system for designing minimal-cost house plans – much like Siza's Malagueira plans – that satisfied specified area and adjacency requirements for the constituent rooms. More recently, Shea and Cagan (1997, 1999) demonstrated the possibility of deriving unusual but practical and efficient structural designs by means of a shape grammar encoding rules of stability in pin-jointed structures, finite-element analysis of alternatives that are generated for consideration, and the use of simulated annealing to control the search

[4] My early architectural CAD text *Computer-Aided Architectural Design* (Mitchell, 1977) explicitly makes this connection and provides many examples.

Figure 1.8. Unusual dome designs generated by Shea's shape annealing procedure.

(Figure 1.8). However, there were several difficulties. One was the difficulty of capturing the subtleties and complexities of practical design goals in terms of constraints and objective functions. A second was the typical intractability of the search problems that result from such formulations. The third – and most profound – derived from the fundamental assumption of definite, discrete elements and treatment of subshapes as sets of such elements.

LIMITATIONS OF COMBINATORIAL SEARCH

To see the nature of this last difficulty, consider the shape shown in Figure 1.9.[5] It is from master mason Mathias Roriczer's famous pamphlet on the design of cathedral pinnacles (Roriczer, 1486).

Roriczer described its construction, in detail, as follows.

> Would you draw a plan for a pinnacle after the mason's art, by regular geometry? Then heave to and make a square as it is here designated by the letters A, B, C, D. Draw A to B, and B to D, and from D to C, and from C to A, so it may be as in the given figure [Figure 1.10].
>
> Then you make another square: divide A to B in two equal parts, and place E; likewise, divide BD and there make H; and from D to C and there make an F; similarly from C to A, and there make a G. After that draw a line from E to H, and from H to F, F to G, G to E. An example is in the following figure [Figure 1.11].
>
> After that you make like the one made above another square: divide FH into two equal parts and there put a K. Similarly on HF place M; also on FG make L, similarly on GE place I. After that draw a line from K to M, from M to L, from L to I, from I to K, as in the following figure [Figure 1.12].
>
> M. Roriczer, *On the Ordination of Pinnacles* (1486)

It is easy to execute this construction in a standard CAD system, such as AutoCad; the result is a shape represented as 12 lines. Any one of these lines may then be selected and transformed or deleted, to produce design variations of the kind shown in Figure 1.13. (To make this exercise interesting, of course, you have to restrict

[5] See also Stiny, Chapter 2.

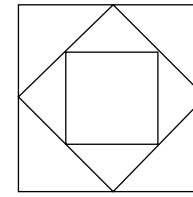


Figure 1.9. Mathias Roriczer's construction of nested squares.

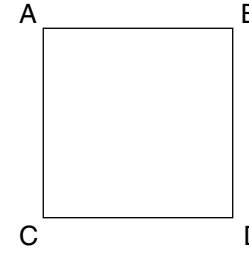


Figure 1.10. The first step in Roriczer's construction procedure.

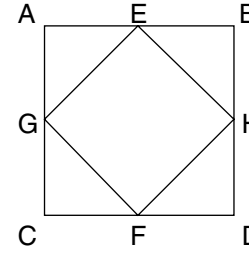


Figure 1.11. The second step in Roriczer's construction procedure.

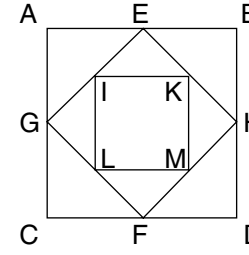


Figure 1.12. The third step in Roriczer's construction procedure.

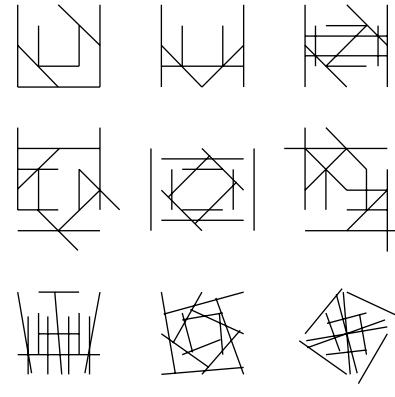


Figure 1.13. Variations on Roriczer's shape produced by selecting and transforming straight lines.

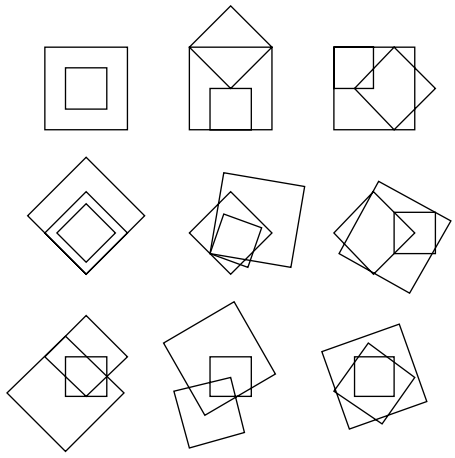


Figure 1.14. Variations on Roriczer's shape produced by selecting and transforming squares.

the allowable transformations such that you preserve some of the properties of the original figure. If you simply allow translation, rotation, reflection, and scaling in general, you end up – unhelpfully – with a universe of all possible figures made from 12 or fewer straight lines.) It is very straightforward to write search procedures that enumerate such designs.

Alternatively, using a standard CAD capability, the 12 lines might be grouped into three squares. Now, selection, transformation, and deletion operations produce variations of the kind shown in Figure 1.14. It is straightforward to produce extended ranges of variants by extending the repertoire of transformations – by allowing shear and stretch, for example.

So far so good; but what if you want to see the shape as a collection of eight triangles, and to produce variations of the kind shown in Figure 1.15? This is a perfectly natural and appropriate thing for a designer to do, but it is impossible to select and transform any of the triangles. As a consequence of the way the shape was constructed, and of the structure that was implicitly assigned in doing so, the triangles simply do not exist in the data structure. A whole universe of potentially interesting design variants has thereby been excluded from consideration.

Of course it is possible to reconstruct Roriczer's shape as a set of triangles, but then the squares are lost. And, as Stiny demonstrates in Chapter 2, the difficulty is

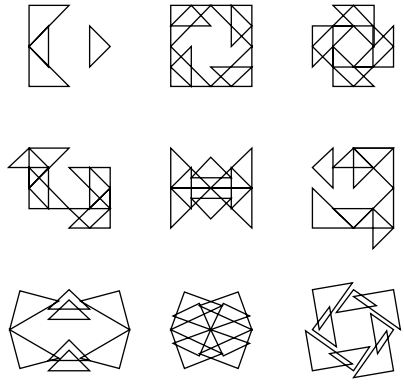


Figure 1.15. Variations on Roriczer's shape produced by selecting and transforming emergent triangles.

far from a trivial annoyance. There are indefinitely many ways to decompose even a very simple shape into subshapes, and any one of these decompositions may be appropriate to a designer's purpose at a particular moment in a design process.

So the classicists and the early modernists got it insidiously wrong. So did the developers of CAD systems, and the authors of design enumeration algorithms, who uncritically accepted the classical idea of particulate composition. Creative "rereading" of shapes, and the subsequent production of variants based upon such rereadings, is an important part of manual design processes. Any computational scheme that prematurely imposes a definite way to parse a composition into parts and subparts will inappropriately constrain a designer's capacity for creative generation of alternatives. It will become a Procrustean bed. The data structures of standard CAD systems, and search strategies that rely upon such data structures, suffer from a very fundamental limitation; they may be useful within their limits, but they do not support a satisfactory general approach to creative design synthesis.

DESIGN DEVELOPMENT AND ASSIGNMENT OF STRUCTURE

Now, however, consider how you might go about *physically* constructing Roriczer's shape – not simply drawing it. Your procedure would obviously depend upon available materials, fabrication techniques, and assembly strategies. If you had some long, rectangular beams, some nails, and a saw, for example, you might create the three-layered structure shown in Figure 1.16. (This is, in fact, a fairly common traditional strategy for framing roofs from heavy beams of limited lengths.) In other words, you would first assign a definite structure – let us call it the "carpenter's structure" – to the shape, and then you would fabricate discrete pieces corresponding to the elements of that structure, and finally you would compose the complete shape by assembling the fabricated elements.

Next, suppose you had a lot of short rods and pin-jointing elements – as in many construction toys. In this case, you might place a jointing element at each node and run rods between, as illustrated in Figure 1.17. Obviously this requires seeing Roriczer's

Figure 1.16. Roriczer's shape developed as a set of heavy roof beams.

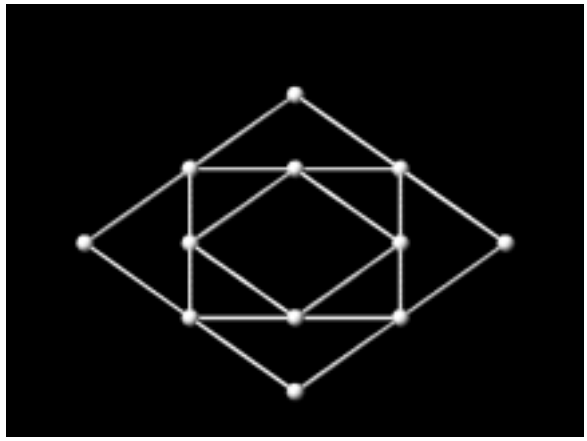


Figure 1.17. Roriczer's shape developed as a pin-jointed space frame.

shape in a different way. And when you see it in this way, you can also quickly see that it might be stretched and folded into the third dimension to produce a beautiful dome (Figure 1.18) – not something that is readily suggested by the carpenter's structure. (However, the carpenter's structure does, of course, suggest *other* interesting things.)

What if you wanted to create the structure from precast concrete components – producing the more complex joints in the factory, and leaving the simpler ones for on-site assembly? You might cast K-shaped and L-shaped elements, as shown in Figure 1.19. Yet another structure – maybe we should call it the "system-builder's structure" – is thereby assigned to Roriczer's shape.

Finally, suppose you had a square piece of plywood and a laser cutter. You might simply cut out four small triangles, to produce the result shown in Figure 1.20. This would particularly have appealed to certain New York painters in the heyday of hard-edge geometric abstraction; let us call it the "minimalist's structure."

Roriczer himself had something else in mind. He thought of the outer square as the plinth of a pinnacle shaft. The second square gave him the size of the shaft, which

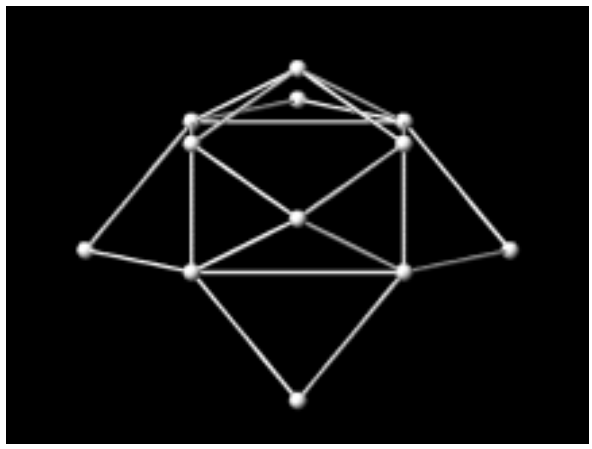


Figure 1.18. The pin-jointed space frame stretched and folded to produce a dome.

Figure 1.19. Roriczer's shape developed as an assembly of K-shaped and L-shaped precast concrete elements.

worked out to half the area of the plinth. The third square gave the size of the faces of the panels of the shaft. He went on to rotate the second square by 45°, add some additional construction, and produce the plan for a carved stone pinnacle as shown in Figure 1.21.

These examples illustrate a general principle of design invention and development. In the early stages of architectural design processes – when architects freely explore possibilities by producing quick, rough sketches – ambiguity of structure and preservation of the possibility of rereading seem crucial. However, sketches are not buildable. As a design is developed, then, and as an architect begins to think in terms of specific materials, components, and fabrication and assembly possibilities, a building design acquires an increasingly definite structure. An individual sensibility about space, materials, construction, and light comes more prominently into play. Finally, the construction documents describe the design as an assembly of definite

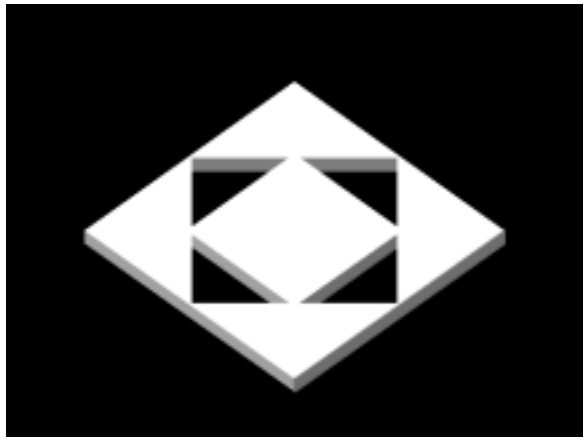


Figure 1.20. Roriczer's shape developed by laser-cutting triangles from a flat sheet.

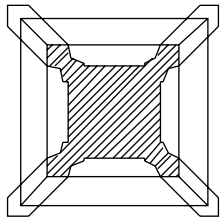


Figure 1.21. Roriczer's own development of the shape as the plan for a cathedral pinnacle.

elements and hierarchically nested subsystems. Ambiguity is a good thing to have at the beginning, but a bad thing to leave there at the end.[6]

FORMAL AND FUNCTIONAL EMERGENCE

Preservation of ambiguity in the early stages of design performs the essential function of allowing a designer to recognize and take advantage of formal and functional *emergence*. For example, if you construct Roriczer's shape by selecting, transforming, and instantiating squares – as an experienced CAD operator might – you never explicitly insert any of the eight visually evident triangles; they simply emerge. Conversely, if you rather perversely construct the shape by selecting, transforming, and instantiating right triangles, then the three squares emerge.

At one level, as was demonstrated in Figure 1.15, recognition and transformation of emergent subshapes supports formal invention. However, there may also be a functional dimension. Recall, for example, that pin-jointed squares are not structurally stable shapes but that pin-jointed triangles are. Thus, recognition of the triangles immediately tells you that you can construct a version of the shape as a pin-jointed frame (as shown, for example, in Figure 1.17 – understood as a truss operating in a vertical plane), that it will then turn out to be structurally stable and statically determinate, and that you will be able to analyze the forces in the members by using elementary techniques of statics.

If you are interested in different functions, you will look for and find different emergent subshapes. Thus, if you read Roriczer's shape as the *parti* for a floor plan, and you know that closed polygons can function as rooms, you can quickly pick out the potential room shapes shown in Figure 1.22. It is a matter of sensibility as well; if you are a postmodernist with a taste for funkiness, then these room shapes may appeal to you, but Durand would have looked straight past them.

Examples could be multiplied indefinitely, but the moral should now be obvious. There are indefinitely many emergent subshapes in even very simple shapes, and the recognition of particular emergent subshapes provides a designer with opportunities to apply knowledge of construction and function. It brings individual experience and sensibility into play, and it provides opportunities to open up new ranges of variants for consideration.

Skilled designers do not simply search for configurations that satisfy predetermined requirements. They watch out for emergent architectural opportunities, they recognize them, and they take advantage of them to achieve unexpected benefits.

[6] To put this another way, a design process should begin within the framework of a shape grammar, but it should end up within the framework of a set grammar.