Incommensurability of Criteria and Focus in Design Generation

Pegor Papazian

School of Architecture and Planning
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

An approach to developing design systems is presented, informed by the recognition that design criteria are incommensurable. The degree to which an artifact satisfies one criterion cannot be compared to the degree to which it satisfies another. Given this principle, it is not valid to combine different "scores" given to independent features in an evolving design into a global evaluation function. The design framework proposed here represents an alternative to the traditional approaches for combining independent criteria and organizational principles. It is based on the opportunistic nature of designing, the multiplicity of semantics active in a design session, and the dynamics of focus and distraction. By way of illustrating both this characterization of designing and the abstract computational framework on which it is based, a simple system for arranging blocks according to a set of formal massing principles is presented. The massing generator has some important properties that other systems lack, such as dynamism, robustness and the ability to deal with partial designs. Through a comparison with some artificial intelligence methods such as production systems and search, the proposed framework is used as a guideline for developing design systems. This paper focuses on designing as an activity, rather than engaging in an analysis of finished designs with the hope of capturing their syntactic properties. Thus the stress is placed on the generator's behavior, by giving examples of how it converges on a series of design alternatives in a dynamic fashion, avoiding oscillations and blocks.

Keywords: design, criteria, opportunism, focus, CAD.

1 Introduction

Evaluation is easy. It is not necessary to be a famous architect to evaluate the work of one. It is particularly easy to evaluate a design in terms of one independent criterion, such as monumentality, cost of execution, some notion of formal coherence, safety, and so on. What can be considerably more difficult is to produce a design which simultaneously satisfies a number of criteria, or even just one. The temptation of many researchers in design automation (and other areas within artificial intelligence) has been to simplify the task of generation, of designing, by proposing relatively simple constructs, such as a "generate-

and-test” mechanism, or an optimizing constraint satisfaction algorithm, in order to reduce the task of generation to that of evaluation. I will not go into the advantages and disadvantages of these systems here. Rather, I will limit myself to pointing out two issues which, to me, seem to constitute fundamental refutations of such approaches.

The first of these, has to do with the fact that typical design activity involves a large number of criteria, many of them unstated, and many of them mutually exclusive or contradictory. Consider the case of an architect commissioned to design a residence with three bedrooms and one guest bedroom, and who presents to the client a design which omits the guest bedroom. Not only is this a possible scenario, but it is also possible that the client will happily accept the design, for the same reason that the architect proposed it. Not because it was impossible to meet all the client’s requirements and the guest bedroom was the “optimal” sacrifice, but because the designer constructed a system of criteria and an artifact which satisfied them in a coherent and interesting manner. The guest bedroom was sacrificed for the sake of some discovered system or set of values with which it was incompatible.

The second related, and perhaps more basic principle, is that different design criteria cannot be directly compared. Although it may not be obvious at first, it does not make sense to compare, for example, the degree to which a building is safe, to the degree to which it is expensive. We can say that it is “very safe” and “very expensive,” but it is meaningless to say that is more safe than it is expensive. What makes this notion confusing to some is that we often use expressions such as “safety is of utmost importance and price is secondary in this building.” Such statements are perfectly coherent, and the prioritization they supply can guide us in being selective in focusing on some design criteria (such as safety) during the course of a design, or when constructing a design generation system, as long as the system does not compare safety and expense as if they were commensurable numbers. The semantics of individually stated design criteria precludes their comparison. The burden of coherence and rationality (the thing that makes designers of nuclear power plants come up time after time with relatively safe expensive buildings, rather than relatively unsafe inexpensive ones) is delegated to the faculty of focus. By creating this abstraction which uncouples the relative values given to criteria from the criteria themselves, we can avoid comparing apples and oranges but still chose, after any given lunch, one fruit or the other, or going for a walk instead. I hope to illustrate below, an approach to designing design systems which uses the principle of focus as a means of bringing criteria into play during a design session in a way which does not violate the incommensurability principle. I will introduce elements of a characterization of designing, and present a computational framework for design generation based on that characterization[1]. The aim is to develop a dynamic substratum of design activity which, even though simple, will be conducive to the complexity associated on the one hand with expertise in designing and, on the other, with design objects themselves.

By way of preview, let us look at an application of the generator meant to illustrate some characteristics of its behavior. The “design task” for this demonstration is to produce arrangements of five blocks of variable height and rectangular footprints of arbitrary dimensions, according to some formal principles of massing. These principles are quite simple and can be expressed in the following four points:

1. Maximum alignment and abutment of the blocks;
2. Compactness (smallest possible maximum length and width);

[1] A more detailed treatment of this characterization of designing, and the computational framework based on it can be found in (Papazian, 1993).

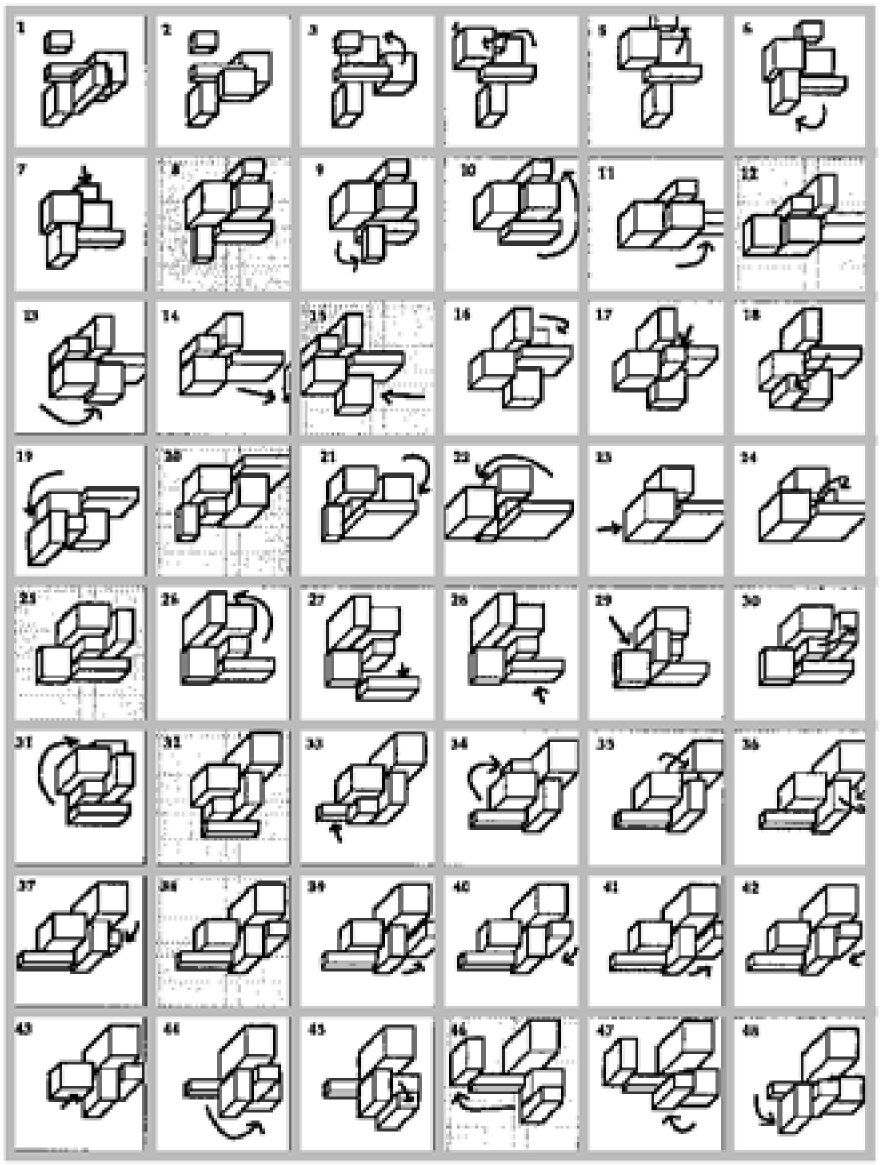

Figure 1. Set of moves of seven blocks, given initial arrangement.

3. Some constant footprint-to-volume ratio; and
4. Visibility of all the blocks from a given view point.

shows a set of moves produced by this application, given an initial arrangement of seven blocks. Since the problem is over-constrained, let us re-interpret the first of the four massing principles, to simply mean that the blocks should form one cluster where any block can be reached from any other through a series of adjacencies.

The highlighted frames in Figure 1 correspond to configurations which meet this criterion and also satisfy the principle of visibility and area-to-volume ratio. By Frame 8 the system has a configuration which conforms to our re-interpreted criteria, but interestingly, after a series of moves, it reaches a configuration in Frame 25 which can be considered optimal for the full set of criteria. The larger blocks are arranged in a pinwheel pattern around the smallest one, thus achieving compactness (due to the pinwheel pattern) and maximum abutments and alignments (due to the fact that the smallest block is in the middle). Had a larger block been in the middle, the configuration could have been more compact, but the number of abutments would have been reduced because the smallest block would not have reached the next block in the pinwheel. Below, I will give some details of the system, and an analysis of certain aspects of its behavior. But first, let us look at some observations about design activity which form the basis of the generator.

2 Designing

In a previous experiment (Fargas and Papazian, 1992), designers were show, randomly generated pictures consisting of a frame, two lines and two rectangles. The design task was to make the picture “more stable” by displacing the rectangles. No clarification was given about the meaning of the term “stable.”[2] The computer would record the designer’s moves, and later produce a real-time replay or a dynamic record of the process as shown in Figure 2.

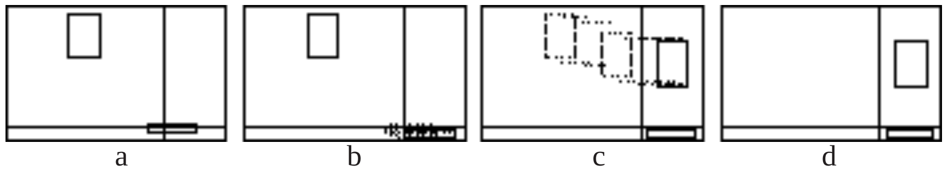

Figure 2. Designer’s moves, as recorded by the computer.

Analysis of the experiment results revealed that each designer would make use of a set of metaphors in order to attach a meaning both to the elements of the picture and to the term “stable” in the problem statement. Seen through what we called a *gravity metaphor* for instance, the rectangles would become “blocks,” the bottom of the frame would represent a “tabletop” and *stable* would come to mean “resting in equilibrium.” The same designer, in a given session, would often shift metaphors (adopting, for instance, a *perspective depth* metaphor where a smaller rectangle would be considered as being “further back”) and would occasionally return to metaphors which were previously abandoned. These different ways of seeing the design document seemed to be the primary driving force behind the designers' generative moves. The mechanism for adopting and abandoning an active metaphor proved to be a fundamentally opportunistic one. In Frame (a) of Figure 2, for example, the fact that the smaller rectangle has proportions very similar to those of the lower right quadrant, which is very close to it, causes the adoption of a center-

[2] Later, a computer program called EstheR (the Esthetics Replicant) was developed to replicate one particular design session of the experiment.

ing metaphor to remain active throughout the session. This characterization of design activity is based on the following four elements which will be presented here without further illustration.[3]

Opportunism, focus and distraction: Designing is based on a substratum of opportunistic activity. At any given time, the designer focuses on a limited number of components and evaluative criteria. When an opportunity is seen which can be exploited in terms of one of a large number of implicit and explicit values, the designer is distracted by it, and a shift of focus takes place. Focus can be a function of the history of a given design session, the bias of the designer, higher-level processes such as planning and inference and, of course, external factors.

Multiplicity of semantics and the exclusivity of seeing: The designer can interpret a design document or artifact by attributing to its components and their relations one of many sets of meanings. At any given (arbitrarily brief) period, the designer can be thought of as actively relying on only one of those potential semantics. The equivalent of simultaneous interpretation in terms of two or more sets of meanings can take place when the design moves generated in terms of each of the semantics are equivalent.

The see-move-see cycle: The above two characteristics can be combined in the see-move-see cycle (Schön and Wiggins, 1992). The designer "sees" the evolving design document or artifact in one of many ways of seeing. The design is appraised in terms of *potentially many* criteria associated with the different ways of seeing. A move is then made which is expected to improve the design from the point of view of *one* criterion of appraisal (or one combination of criteria in the special case of "simultaneous" moves).

The following scenario, illustrated in Figure 3, may help clarify the role of opportunism, focus and multiple semantics in the see-move-see cycle. In Frame (a), the designer sees the layout of a building in terms of its programmatic requirements, and decides that one of the rooms is too small. In Frame (b), the designer extends one edge of the small room in order to enlarge it. As an unexpected result of this extension, an opportunity is seen, in Frame (c), to create a south-facing U-shaped courtyard by extending the right wing of the building, as shown in Frame (d). Notice how evaluation and generation interact in this model. The *appraisal* of the document in Frame (c) involves a recognition of the potential for a courtyard. Inherent in this recognition is the designer's *know-how*: the knowledge of a move expected to create such a courtyard.[4]

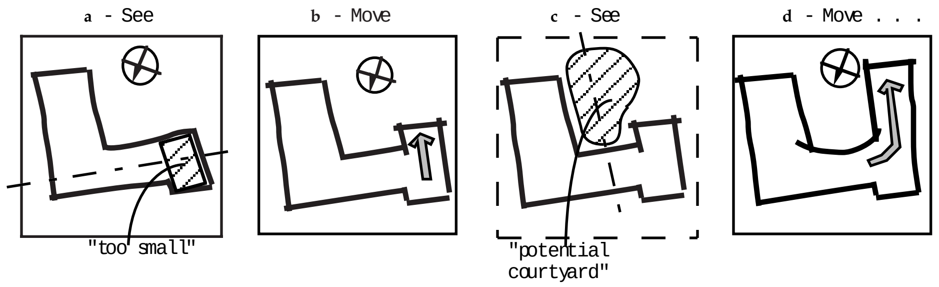

Figure 3. A design scenario that demonstrates the move-and-see cycle.

Note that the see-move-see cycle is not a generate-and-test cycle. Although in the see-move-see cycle there is a succession of generation and evaluation, the basic mecha-

[3] For an elaboration of these elements see (Papazian, 1991).

nism is not one of generating a design, seeing if it meets all the desired criteria (or to what extent it meets each of the criteria) and accepting or rejecting it on that basis. Rather, the idea is to evaluate the evolving design after each act of generation, *in order to generate the next move*. There is no testing. Although designers do backtrack, abandoning a whole path of exploration which turned out to be unsuccessful, they do not do so at each move. Of course there are tentative moves, made and immediately retracted, not typically retracted on the basis of a global test, but as part of an evaluation specific to the criteria which led to that move. Nor is the see-move-see cycle a successive approximations or incremental refinement cycle. The major indication being that a move made with the expectation of improving some quality of the design, may leave it in worse shape in terms of some (or even every) other quality. Of course one would like to make only moves which result in improvements for every relevant quality of an artifact. But in the typically overconstrained and ill-defined domain of design tasks, requiring that every move be universally positive is counterproductive as we will see below.

3 Generation

The design generation framework I will introduce at this point is inspired by the characterization of design activity presented above. It is based on an arbitrary number of metaphor modules. Each of these modules represents a different way of interpreting the current state of the design document (a different “seeing-as function”) as well as a set of heuristics corresponding to a formal organizing principle (such as symmetry, for example). Each module dynamically generates a subjective description of the evolving design document from its particular point of view, and proposes transformations *of this subjective description* which are expected to make it more consistent with the module's organizing principle. The modules thus interact in a see-move-see cycle, where all modules *see* the design document, producing their subjective descriptions, and propose transformations which are mapped into potential design moves effecting the evolving design. The design generation framework then chooses one of these potential moves as the transformation of the document for the current cycle, and solicits the set of move proposals for the next cycle. The choice of the particular move is based on an opportunistic mechanism, where the degree of salience of the opportunity for applying a heuristic is the main factor determining the adoption of the corresponding move.

Once a move is made, the system adjusts its own focus, favoring future transformations which apply to elements of the design document involved in the execution of the current move. Given a particularly salient opportunity for a given module to transform elements of the document which are not currently in focus, a phenomenon corresponding to distraction takes place, resulting in a shift of focus in the next cycle. The choice of a particular move is also a factor of the history of previous transformations. For example, the likelihood of a given module being the author of the next move is higher if it had just been the new author of the move adopted in the previous cycle, but drops off as the module authors a long series of consecutive moves. Finally, the system favors moves which satisfy the proposals of more than one module in a process of consolidation which will be explained in more detail below. Notice that this mechanism respects the principle of the

[4] This idea that know-how is somehow intrinsic in the act of appraisal is in some ways related to Wittgenstein's notion that expectations contain a “picture” of the thing expected: “Expectation is connected with looking for. My looking for something presupposes that I know what I am looking for, without what I am looking for having to exist” (Kripke, 1982).

incommensurability of design criteria, since any appraisal of the document done by a module is independent of all the other modules' appraisals, and the system, when choosing one of the several proposed moves, never refers to the evaluation that the modules may have made of the current state of the document.

Let us now return to the application of arranging blocks according to the massing principles of maximum alignment, compactness, constant footprint-to-area ratio and visibility of all the blocks. Depending on one's reading of these principles, the task of arranging three blocks according to them can be considered underconstrained. Let us say, for instance, that what we mean by “maximum abutment” is that there be a condition of adjacency among every pair of blocks in the configuration. And let us take “maximum alignment” to mean that, in addition, there be as many conditions of alignment as possible among the footprints of the blocks, given the particular blocks in question. In that case, given any three (or fewer) blocks, a satisfactory arrangement can be found. But in the case of five blocks shown in Figure 1, the task is clearly overconstrained.

Figure 4 shows the initial condition of the document (in the upper left window) which contains seven blocks of arbitrary dimensions and location. The other windows show the document as seen by each of five modules. Each module corresponds to one criterion in the design task. The *Overlap* module “sees” the document as the configuration spaces of the blocks, and tries to undo overlaps; the *Number* module sees the document simply as a number (the number of blocks) and tries to add or remove blocks to keep the number at 5 (in this case); the *Align* module sees continuous lines and tries to make them coincide; the *Corners* module sees concave and convex corners and tries to match them; the *Environ* module sees the blocks in perspective from a specific view point and adjusts heights for visibility and a fixed footprint/volume ratio.

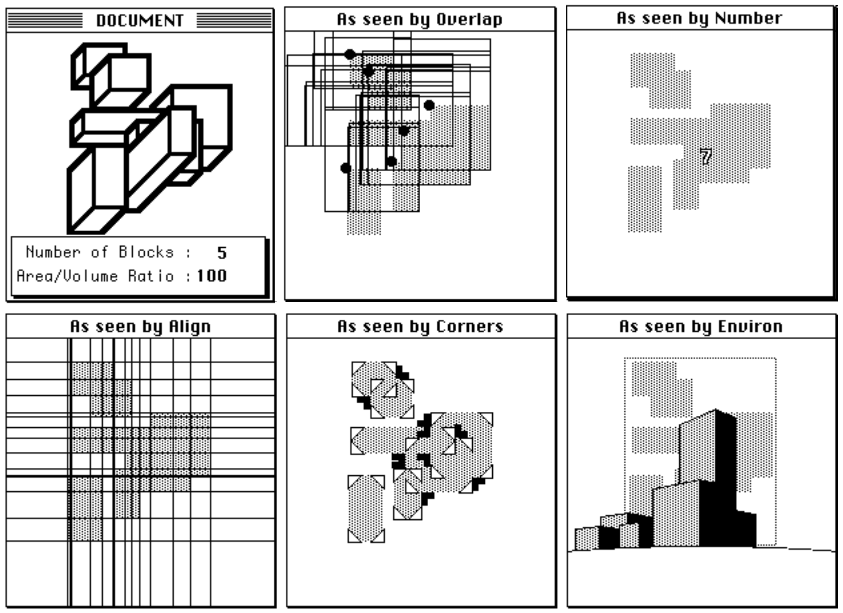

Figure 4. Initial (upper left) and subsequent conditions of a document containing seven blocks of arbitrary dimensions and location.

Each module proposes a different design move, and it is up to the system to try to "consolidate" them as much as possible (see Figure 5) and to choose one of them as the next transformation of the document. For example, the *Align* module, as shown in Figure 5, sees the document as pairs of vertical and horizontal lines. For each block it sees two pairs of lines corresponding to each pair of parallel sides of the block. This module favors coincident lines in its subjective description. The moves it proposes displace one line to place it over another. At the level of the document, this results in moving a block to one of a set of possible destinations, all of them having the property that they are expected to create alignments. The gray area in the window "Candidate: ALIGN-MOVE-2" of Figure 5 is one set of destinations proposed by the Align module.

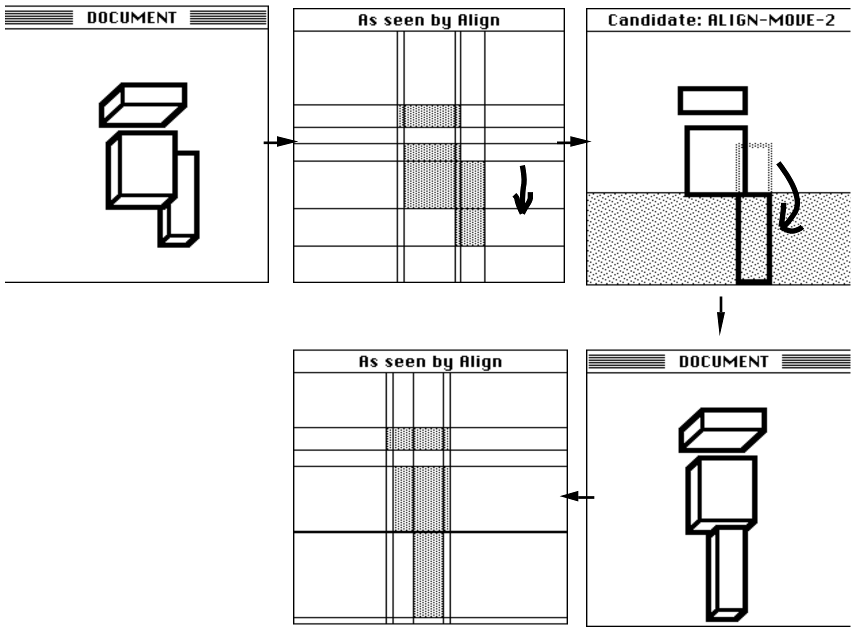

Figure 5. Possible design moves, as proposed by different system modules.

The different modules propose candidates for the next state of the document in a parametric fashion. They specify a set of acceptable positions for different blocks as disjunctions of points. It is part of the system's task to find a more specific next state which is the consolidation of any two proposed parametric next states (see Figure 6). It is important to make the following two points about this process: First, *Consolidation* is not an averaging operation; it involves no compromise among the different proposals. Second, the parametric nature of the final proposal is abandoned once it is chosen as the next state of the document; only its default position is retained.

4 Evaluation

Basing our evaluation of the system on some ratio of "successful solutions" to "failures" would not be very significant because this particular block arrangement problem is

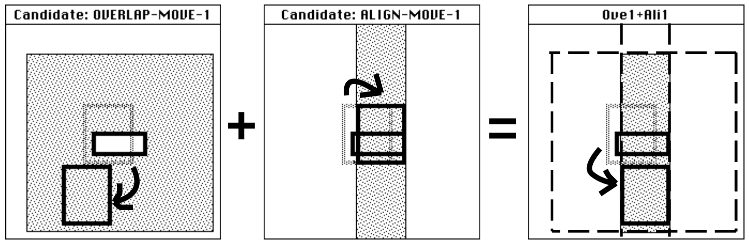

Figure 6. Consolidation of any two proposed parametric next states.

probably better handled by some other, more algorithmic method if one's aim is simply to find solutions. What is interesting about this generator is the way in which it reaches its "solutions," rather than their validity. Its performance is based on the idea that guarantees of dynamism are important at a more fundamental level than guarantees of optimum solutions.[5]

Similarly, to assume that the merit of the computational framework discussed above is in its architecture, its structure, would be a misreading of the work presented here. The truth of this becomes evident once we consider that such a framework can easily be used to construct applications with almost any behavior, and that some possible applications of the framework (for example, to implement a shape grammar) can be quite efficient and appropriate, even though the workings of the resultant system would be very different from the advocated one. Furthermore, it is only a little more difficult to imagine how the behavior which characterizes the massing application shown above can be achieved using a substantially different technology (but not, I will claim, while ignoring the importance of the incommensurability of criteria, the concepts of focus and distraction, the opportunistic nature of designing, and so on).

The merit of the framework, then, partly depends on its usefulness in illustrating the concepts of my characterization of designing and the fact that it acts as a guide for designing applications informed by this characterization. But more importantly, the presentation of the computational framework serves the aim of constructing a vocabulary as well as a graphic expression which reflect the spirit of my approach. Why is the form of expression important? Because the computational framework is meant to represent a type of *tool for designing* and not just a program for solving certain kinds of problems. As such, it needs to make very explicit the concepts and principles which its user must manipulate in order to use it creatively. My main reason for making these points, however, is not to justify the use of an alternative terminology but to clear the way for a review of the lessons learned from this experiment. The way I propose to do this is by comparing the approach suggested here to some traditional techniques, by asking, for example, "If this were a 'production system' or was considered a 'blackboard architecture,' what would be its distinguishing features?" and "If this were seen as a way of searching a 'solution space,' what kind of search would it be?"

[5] The importance of a dynamic characteristic of activity goes beyond the field of design and has been the focus of some efforts in other fields within artificial intelligence (Age, 1988).

5 Production

A production system is based on a set of “if *condition* then *action*” rules. Given the current state of an evolving problem (or, in this case, an evolving artifact) some of these rules may be “triggered,” meaning that the *condition* part of the rule is satisfied. If the *action* part of the rule is then executed, we say that the rule is “fired.” An extremely simple example of such a system is contained in the Numbers module of the massing application above. It consists of two rules, one of which says “if the number of blocks is less than n, then add a block,” and the other says “if the number of blocks is more than n, then remove a block.” Since the conditions of these two rules are mutually exclusive, they will never be triggered simultaneously. Now if we consider the whole massing application as one production system, we will need what is called a conflict resolution strategy for deciding which of the triggered actions (in this case, which of the “candidate moves”) is chosen for firing.

Although rule-based systems may not be the most appropriate technology to use for implementing a given metaphor module, it is easy to imagine how the modules of the massing application can be expressed as sets of “if ... then...” rules. One important organizational consideration, however, will be that both the *condition* and the *action* part of the rule must refer only to information which is available in the subjective description of the corresponding metaphor. The *Numbers* module should not have a rule which refers to whether two blocks are of the same size, for example. It does not “see” two blocks. It just sees the number 2. Another important, and perhaps more fundamental, consideration has to do with the conflict resolution strategy used by the system's arbitration process. Some conflict resolution strategies use a previously established list of priorities in order to choose among triggered rules. Others rely on some formal properties of the rules, such as “specificity” (the rule which has the most stringent conditions is assumed to be the most appropriate one). Other strategies tentatively fire all the rules to determine which one would give the “most favorable” outcome. I will not give a list of objections and counter-example for each of these strategies. Instead, I will try to illustrate the principle of incommensurability of design criteria as the basis for an alternative approach to knowing what to do next.

Consider the task of designing a die for fairly choosing among six numbers. One of the criteria for the design may be the die's shape. Another, its color. We wish to have a die which is cubical (its corners may or may not be rounded, it could be larger or smaller . . . but we want it cubical). And, its color should be dark (black, dark red, navy blue . . . but not sky blue, for example.) Now imagine a design proposal which consists of a spherical black die, and another of a beige cubical die. Which is worse? Why? Assuming you prefer the cubical beige die, here is a more difficult question for you: How much worse is the black spherical one? Let us make the spherical die a little more angular, like an over-inflated cube. Can you decide how angular it needs to be in order to be even in value with the cubical die of the wrong color? I claim that all these questions are meaningless, unless we give enough additional information to make them trivial. What is certain, is that a characterization of designing which systematically prioritizes shape over color in this example, cannot account for the invention of those spherical dies which are perfectly functional (even more than cubical ones) due to a system of internal indentations and a metallic ball. An inverse prioritization will fail to account for a cubical white die being chosen because of the elegance or appropriateness of the markings on its faces, or the sound it makes when rolled.

The alternative approach which will keep us from falling into the trap of comparing incommensurable criteria, is to adopt a principle of exclusivity, focusing on one criterion, component or aspect of a design at any given instant, and relying on a mechanism of opportunism and distraction to favor the inclusion of all elements over the course of a design session. This is why, if our generative system were a production system, its conflict resolution process would rely on factors such as the salience (Nissenbaum, 1985) of features in an evolving design, particularities of the current situation (Suchman,1987), perceived opportunities (Visser, 1991), and so on, instead of an *a priori* ranking, or a global evaluation formula.

Blackboard systems are composed of a number of agents specialized in particular aspects of a task, and a shared construct, the blackboard, as their only medium of communication. This very abstract structure, which has been reincarnated in numerous variations and transformations, can obviously be applied to our generative system. In addition to the characteristics given above for our generator dressed up as a productions system, there is an important principle to consider for dressing it up as a blackboard-type systems. This is the principle of the manifest nature of design documents. Not all the intentions and constraints resulting in the creation of components (and their relationships) in a design document are made explicit in it. A computational system which compensates for this apparent shortcoming by extensive annotation and constraint management, runs the risk of losing the immediacy characteristic of the designer's interaction with the document. Due to the overwhelming ubiquity of constraints, the designer needs not only the ambiguity of a document (an intersection of lines can be a cross or two L shapes) but also the arbitrariness inherent in it (a line which could satisfy the relevant constraints by being anywhere within a range of locations, is actually placed in one particular location and the designer's subsequent interactions with it are, at least in part, *a function of that particular location*). Thus the “blackboard” in such a system needs to have only a limited trace of the intentions associated with the design history, not just to reduce computational complexity, but as a basic strategy for supporting focus and opportunism. Finally, a secondary difference between the approach to designing design systems advocated here and the traditional blackboard architecture approach is the importance given to the abstract architecture of the system. As pointed out above, the claim to merit of this paper is not based on the structure of the generator.

6 Searching

It is interesting to think about the generator's behavior as a process of searching through a solution space. This metaphor may not be a particularly appropriate one for several reasons, including the fact that it is not necessarily clear what the goal of the search is in this case. But looking at the generator as a method of search can be quite revealing, specially in terms of a comparison with traditional search techniques. Some search techniques are based on gradient ascent or “hill climbing,” proceeding from each point in the solution space in the direction of greatest improvement, and consequently stopping at points where progress in any direction leads to a worse solution than the one corresponding to that point. Such points may be local maxima, and in order to reach a better solution the search must have a systematic way of letting things get worse before they get better. Although hill climbing is a particularly naive method, and not representative of search techniques in general, it is a convenient model with which to compare the generator's behavioral characteristics. More importantly, if the generator's movement through a solution space is considered searching, it represents an alternative method in many cases where hill climbing might have seemed an appropriate method. Some techniques, such as “simulated annealing” (Kirkpatrick, 1987) try to alleviate the problems associated with hill climbing by introducing a controlled but arbitrary “jitteriness” in the search path with the hope of dislodging it from local maxima (hilltops) where it may be stuck, and making the search more dynamic. This apparently less naive method is in direct contrast with the generator's use of changes in focus in order to explore a solution space dynamically, yet according to the relevant evaluative criteria or principles.

The generator does not suffer from the problems associated with gradient ascent search methods because its progress is made with the aim of improving one criterion at a time and because it will take steps even when they may seem to lead to “worse solutions.” To be more precise, I should say that it has no notion of the overall value of a current state, and it makes moves simply on the expectation of improving the design from the point of view of one of its modules (or several of them in the case of consolidation). Even this improvement in terms of one (or several) of the modules is not guaranteed. Imagine a solution space where each dimension of the space represents the value of a given configuration of blocks in terms of one criterion. Adjacent points in this space do not represent configurations which are physically similar, but ones which have equal or adjacent values for all the criteria. A gradient ascent algorithm will use some function to assign an overall score to each point. These scores are not necessarily a continuous function in the solution space, and we can assume that the gradient ascent algorithm in question has the necessary repertoire of moves to trace a discontinuous path through it. The critical observation to make at this point is that a reasonable repertoire of moves will make it possible to visit only some of the points starting from any given point. If the gradient ascent algorithm, through a succession of better states, gets to a point from where it only knows how to reach worse states, it will be stuck. Note that this is in spite of the fact that its repertoire of moves may have allowed it to reach a more favorable state if it had taken a different path at certain points in its progress on condition of tolerating worse states. Given the same repertoire of moves, the generator will move to one of the possible solutions without the condition of improvement and, unlike simulated annealing algorithms, it will do so guided by one or more of its metaphor modules.

As an illustration of this phenomenon, consider Frames 22-25 of Figure 1, which are also shown in Figure 7. In Frame 22, a block is moved to a new position where it is adjacent to only one other block. In Frame 23, to maximize the abutments of the configuration, this same block is pushed into a position where it abuts three blocks instead of one. On the other hand, it now completely overlaps with a fourth block. Comparing Frames 22 and 23 from the point of view of all the modules might find Frame 22 to be a better solution, yet the move in Frame 23 would be made.

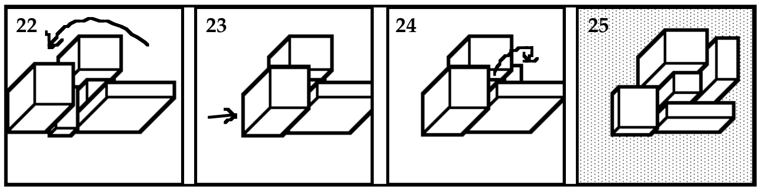

Figure 7. Solutions proposed by the generator as guided by its metaphor modules.

This move was proposed by both the *Align* and *Corners* modules, neither of which takes overlap into consideration. The *Overlap* module, even though it gave a very negative appraisal value to the proposal, has no counter-proposals because it sees no overlap to be undone. Therefore the proposal which diminishes the current value of the design is adopted as the next state of the document. In Frame 24, the *Overlap* module, having missed a turn and having given a very low appraisal value to the current state of the design makes its comeback with high "eagerness." The component-level focus switches from the bigger block to one which is now covered by it, because it has the largest proportion of overlapping footprint area (again, as a result of the *Overlap* module's eagerness). Therefore, in Frame 24, this smaller rectangle is moved by the *Overlap* module to a nearby corner, in collaboration with the *Corners* module. After heights are adjusted, the configuration in Frame 25 proves to be the best one in the entire session.

7 Repetition

A danger associated with optimization approaches, and with generative systems such as the generator presented here (as opposed to more syntactical approaches such as shape grammars) is the possibility for oscillation. Oscillation is a typical problem for algorithms using optimization techniques. In the case of the generator, the danger is in encountering a situation where the arbitration process always leads to the same series of moves being repeated in a cycle. But let us make an intuitive distinction between oscillation and repetition. Let us say that oscillation is pathological in certain kinds of activity such as designing, while repetition may be a natural part of designing, occurring during backtracking, hesitation, learning processes, and other behavior which, although perhaps inefficient, can be considered necessary. Although explicit checks against oscillation may be valid for any systems of this type, the generator has some inherent features which reduce the apparent likelihood of oscillation without any explicit checks. The process recorded in Figure 1 shows instances of repetition which do not degenerate into oscillation, thereby giving some superficial evidence for the system's resistance to such phenomena. There are two main factors which reduce the generator's likelihood of oscillation. The first is that shifts of focus both at the level of components and at the level of the modules themselves reduce the likelihood of the same moves, involving the same components, occurring repeatedly. The second reason is due to consolidation. Moves which are the result of a cooperation between more than one module have inverses which are much less likely to be formed than the inverse of a single move (which is often the move itself applied in an inverse fashion.) Let us take a closer look at two instances where oscillation is avoided in the session depicted in Figure 1.

Note that Frame 25 and Frame 30 show identical states of the document (see Figure 8). This similarity is not a negative feature in itself; it could even be considered a positive feature due to the characteristics of exploration which it implies. The danger it raises, however, is that of giving rise to an endlessly repeating cycle which would prevent exploration.

It is clear that in this case, not only are the two similar configurations reached via different transformations, but they also give rise to different paths of exploration, even though they are identical. It is important to point out that this divergence is not due to randomness. It is due to the fact that although there was a repetition in the solution space, that repetition did not coincide with one occurring in other state variables related to which blocks the system is focusing on, and which metaphor modules are currently predominant. In other words, when in Frame 30 the state of the document was the same as in Frame 25, the focus of the system was not the same, and hence the paths diverged. This kind of repetition can be called "long-term" repetition, because it involves a sequence of more than one or two transformations. Nested inside that sequence is an example of "short-term"

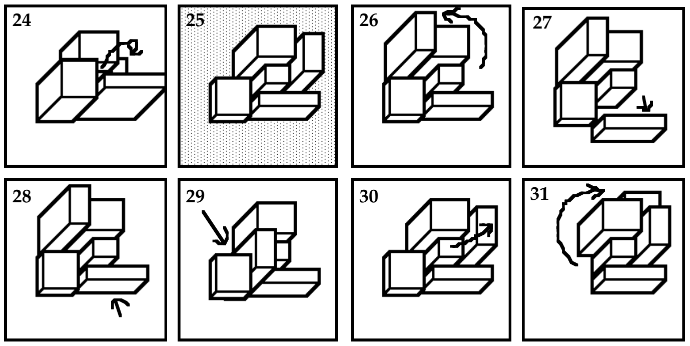

Figure 8. Two similar design configurations reached via different transformations.

repetition which is more common in overconstrained cases such as this. Frames 26-27 and 28 constitute an A-B-A pattern which is immediately followed by a different state of the document. Here, the reason oscillation is avoided is due to a component-level shift of focus. A more critical example of such a repetition occurs in Frames 39-42 (Figure 9), where an A-B-A-B repetition takes place. This pattern is also followed by a new state of the design in Frame 43.

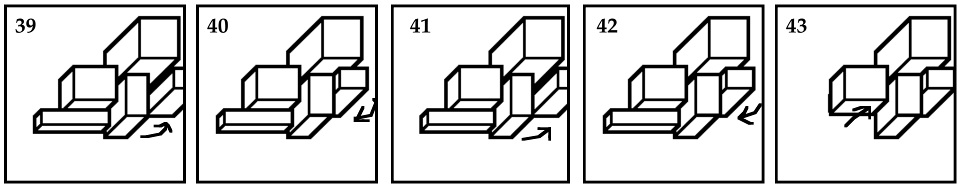

Figure 9.

Thus, the generator makes it more likely that cycles which do occur can be expected to be relatively long-term. This may seem problematic because long-term cycles are more difficult to detect than short-term ones. But because there are only a finite number of configurations, the ideal exploratory behavior would be one which cycles over a large number of favorable ones. The shorter the cycle is, the more limited the exploration will be.

8 Conclusion

I have tried to illustrate an approach to combining a set of basic formal principles in a generative framework, based on the incommensurability of design criteria. On one level, this paper illustrates how an opportunistic approach to transforming a current state of a design, according to one of many criteria at a time, can be combined with the idea of focus (that is, of a selective filtering of objects, intentions, and systems of representation) in order to maintain dynamic and principled behavior in the face of complexity. On another level, the paper suggests a new mode of creativity, where the designer would construct “design personas” as a collection of metaphor modules.

The act of building a machine that generates a family of novel artifacts is a creative one. A computational framework such as the one presented in this paper can provide a level of guidance and organization, but using it in specific instances involves a special kind of designing. It is important to stress that the design task in question would not typically be that of replicating existing designs or mimicking designers, but the creation of new designs. As such, it is a constructive task, involving some skills which are closely related to ones traditionally considered important in the field in question, as well as other special design abilities. This approach involves an unprecedented interaction between conceptual/analytical issues and systems of production. It implies, on the one hand a high degree of explicitness in generative principles and hence of control over the process of production and, on the other, a loss of direct control due to the mediation of the computational framework. In that sense, a radical transformation of the current tradition of design is involved in such a process, just as the current tradition of architectural design, for example, involved a radical (though gradual) break with that of the master mason.

References

Agre, P., 1988. *The Dynamic Structure of Everyday Life*. Doctoral Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Fargas, J., and Papazian, P.,1992. “Metaphors in Design: An Experiment with a Frame, Two Lines and Two Rectangles,” in K. Kensek and D. Noble (eds.), *Computer Supported Design in Architecture*, ACADIA '92, Charleston, SC.

Kirkpatrick, S., et al., 1987. “Optimization by Simulated Annealing,” in M.A. Fischler and O. Ferschein (eds.), *Readings in Computer Vision*. Los Altos: Morgan Kaufman.

Kripke, S., 1982. *Wittgenstein on Rules and Private Language*. Cambridge: Harvard University Press.

Kroll, L., 1987. *An Architecture of Complexity*. Cambridge: MIT Press.

Nissenbaum, H.F., 1985. *Emotion and Focus*. Stanford: Center for the Study of Language and Information.

Papazian, P., 1991. “Principles, Opportunism, and Seeing in Design,” MIT Artificial Intelligence Laboratory, Memo No. 1309.

Papazian, P., 1993. “Mixtures of Simples: Towards a Design Technology,” to appear in A. Tzonis and I. White (eds.), *Automation Based Creative Design (ABCD)*. Amsterdam: Elsevier.

Schön, D., and Wiggins, G.,1992. “Kinds of Seeing and Their Functions in Designing,” *Design Studies* Vol. 13, No. 2.

Suchman, L.A. 1987, *Plans and Situated Actions*. Cambridge: Cambridge University Press.

Visser, W., 1991. “Planning in Routine Design: Some Counter-intuitive Data from Empirical Studies,” presented at the AI in Design Workshop of the 12th IJCAI, Sydney, Australia.