A PERFORMANCE-BASED PARADIGM OF DESIGN

YEHUDA E. KALAY

University of California, Berkeley, USA

AND

GIANFRANCO CARRARA

Università degli Studi di Roma La Sapienza, Italy

Abstract. This paper proposes an alternative approach to existing design paradigms (and their CAD implementations) that are based on the traditional, causality-based notion that 'Form follows Function.' The proposed approach, which will be referred to as *performance-based design*, is founded on the argument that the relationship between form and function is *contextual* rather than *causal*. Hence, the expected *performance* of a given design proposal can only be determined by an *interpretive evaluation*, which considers the form (and other physical attributes) of the proposed design, the functional objectives (goals) that it attempts to achieve, and the circumstances under which the two come together (the context). The paper develops a performance-based design methodology and demonstrates its application in an experimental, knowledge-based CAD system.

1. Introduction

The quest for understanding how humans perform complex cognitive activities such as architectural design, and for developing methods and tools that can help them consistently achieve desired results, has been the *raison d'être* of design methods research for the past four decades. The formulation of such methods has followed, by and large, the scientific method of developing *theories* that explain the process of design, then casting them in *models* that can be represented explicitly and *implemented* by computational and other means. This endeavor has been mostly guided by the conventional wisdom that architecture, more than any other design activity, is a process that seeks a convergence of *form* and *function* within a particular spatio-temporal *context*: a physical container, or a stage, which will support certain human activities, subject to certain conditions and constraints, while being embedded in a particular social and physical context.

Following Thomas Kuhn's influential work concerning the nature of scientific inquiry (Kuhn, 1962), many design methods researchers sought a *causal* relationship between form and function, and a method for deriving *form* from *function*, or *function* from *form*. At the core of this quest lay two assumptions:

1. That a physical system's significant geometrical (and material) properties have some functional utility, and that one form is more suitable to fulfilling that function than other, alternative, forms.
2. That finding a causal relationship between form and function will lead to the development of a method, which can be applied with some assurance of success in every case where a form must be produced that will optimally facilitate and support a given set of functional needs, or that the function of a given object can be determined by closely examining its form.

Two fundamentally different paradigms of design have emerged, representing two different interpretations of the causal relationship between form and function. The first, attributed to Simon, Newell, and Shaw (Simon, 1979), attempts to explain design as a unique instance of general *problem solving*. It postulates that designers start with the sought *function* (i.e., the desired behavior of the system), which is often represented as a set of goals and constraints. The designers then attempt to discover a *form* that will support the desired function, using *deductive* search strategies.

The other paradigm, which is called *puzzle-making*, has emerged from the work of researchers like Alexander (1964) and Archea (1987). It postulates that designers begin with a set of *forms* (that include materials as well as geometry), which are modified and adapted until they achieve some desired *functional* qualities. This paradigm is based on *inductive* reasoning, and has been modeled with the aid of analogical inferencing methods (metaphors, symbols, and case studies).

While logically consistent and computationally convenient, neither one of these two paradigms, nor their many derivatives and permutations, has gained favor with architects themselves. The essence of the profession's criticism can be summarized as the failure of the proposed paradigms to account for the celebrated *intuitive leap*, that elusive but well-known moment when form and function seem to converge into a meaningful whole (Norman, 1987). From a logical point of view, the intuitive leap represents a *discontinuity* in the causal relationship between form and function, hence a stumbling block in developing a uni-directional (form-to-function, or function-to-form) design theory.

This paper proposes an alternative approach to understanding the process of architectural design, which attempts to account for the 'intuitive leap'

experience and fit it within a design paradigm that can be formally modeled and explicitly represented. The proposed approach, which will be referred to as *Performance-Based Design*, does not attempt to formalize the intuitive leap itself, only to *accommodate* it in the paradigm. It proposes to do so by promoting *interpretive evaluation* over *causality*. Instead of deriving form from function, or function from form, it advocates the development of means that measure *performance*—the interpreted utility derived from a given form and a given set of functional requirements within a particular spatio-temporal context.

The notion of *performance* is derived from the argument that the relationship between form and function is *context-based*, rather than *causality-based*. That is, the performance of a proposed design solution can only be determined by an interpretive evaluation which considers the *form* (and other physical attributes) of the proposed solution, the *functional* objectives (goals) that it attempts to achieve, and the *circumstances* under which the two come together. *Performance-based design* recognizes that different forms can successfully achieve similar functions, and that different functions can often be derived from similar forms. In addition, it accounts for performance variances of the same form/function combinations within different contexts. Consider the following example:

If all the chairs in a lecture hall are occupied, students will often sit on the floor. The shape (form) of the floor is not at all similar to the form of the chairs (nor is it usually made of the same materials). Yet, under certain circumstances and within certain limits, it is *functionally equivalent* to chairs. The same floor can also be used for many purposes other than sitting. For example, it can be walked upon, danced upon, and large objects can be placed on it. It can even be used to identify groups of spaces (as in 'first floor'). Included among all these varied functions is also the function of sitting. Obviously, the floor is not as well adapted for the function of sitting as are chairs, and most students will prefer to sit on chairs whenever possible. However, if chairs are not available, they will, in many cases, consider the floor *functionally adequate* for the purpose of sitting. The floor will *not* be considered functionally adequate for sitting when the context is an elegant cocktail party in the White House, as compared to a lecture in a university, and when the participants are foreign diplomats and not foreign students. In that case, the participants will prefer to stand, even if standing causes them considerable discomfort.

A computational implementation of this paradigm, which considers form and function to be equally important, requires that both be represented explicitly, along with the context of the particular design problem. The 'adequacy' of a particular combination of form-function-context will then be determined by evaluators (which may very well include the designer himself), which use a variety of means to predict the performance that will

ensue from the combination of form, function and context, then judge the desirability of that performance in comparison with alternative form-function-context combinations.

In the following pages, we develop the argument for performance-based design. Then we present a model of computer-aided performance-based design, and present an experimental knowledge-based CAD system that implements it. Initially, we will use the terms 'form,' 'function,' 'context' and 'performance' loosely, relying on the reader's intuitive understanding of their meaning. We will define these terms more precisely in the second part of the paper, where such rigor is needed.

2. Causality-Based Design Paradigms

The notion that Form follows Function is derived from the assumption that a system's significant geometrical (and material)¹ properties have some functional utility, and that one form is more suitable for fulfilling that function than other, alternative forms. This notion, and its inverse (function is derived from form), have guided architects and engineers for millennia. Among other achievements it has also provided a convenient causal relationship between *form* and *function*, the two pillars of architectural design, hence for developing theories and methods intended to assist architects in performing their increasingly more demanding task of finding the "right" form-function combination.

Many formal theories and methods that were developed over the years to assist architects have been based on this logical foundation. They can, nonetheless, be classified into two general groups (Aksoylu, 1982): those that follow Simon, Newell and Shaw's (1979) Problem-Solving paradigm, and those that follow what Archea (1987) called "Puzzle-Making."

2.1. PROBLEM-SOLVING

Problem-solving is a general theory that attempts to explain the cognitive process of creative thinking. It was first formalized by Simon, Newell and Shaw in the late 1950s, and implemented in a computer program called GPS (General Problem Solver). Problem-solving assumes that the desired effects of some intellectual effort can be stated in the form of constraints and goals at the outset of the quest for a solution to achieve them. To find the solution, the problem solver uses a variety of search strategies to generate successive candidate solutions and test them against the stated goals, until one is found which meets them. The goals thus 'guide' the search for a solution right

¹ In the following, the term 'form' will be used to refer to all the physical attributes of objects, including their material composition, surface finish, and so on.

from the beginning of the problem-solving process. Problem-solving assumes that setting goals (i.e., knowing what should be accomplished) can be separated from the process of finding a solution that meets them, and that such knowledge can be acquired through an independent inquiry (analysis), which should be completed before the search for a solution has been initiated (Akin, 1978). For example, using this approach, selecting a structural system to span some opening will generally follow after an analysis of forces, cost, and other characteristics of the structure have been determined.

Since the characteristics of the problem, according to the problem-solving paradigm, are known prior to commencing the search for the solution itself, its proponents hold that the search for a “satisficing”² solution is goal-directed, and therefore that means-ends analysis can be employed to guide the search towards finding the desired solution. Thus, the skills that are employed when following the problem-solving paradigm are mainly *analytical*: the ability to compare the current “state” of the designed artifact to its desired “state” (in terms of its expected utility and behavior), and the ability to draw operational conclusions from this comparison so that the differences can be reduced.

Such goal-driven approaches have been computationally represented as deductive, backward-reasoning search strategies, where operators are applied to the goal statement in order to convert it into a set of sub-goals that are easier to solve. This method is applied recursively until a set of subgoals that can be solved without further reduction is found (Laird et al, 1986). Examples of tools based on this paradigm include space allocation programs (Armour and Buffa, 1969; Shaviv and Gali, 1974), and a large number of evaluation programs, such as wayfinding and energy (Gross and Zimring, 1992; Shaviv and Kalay, 1992).

2.2. PUZZLE-MAKING

The assumption that, in architecture, the characteristics of the desired solution can be formulated prior to and independently of the search for the solution that satisfies them was rejected by critics like Archea (1987) and Bijl (1987). They argued that such knowledge cannot exist prior to the search itself, since the sought solution is *unique*, and the process of finding it is characterized by *discovery* and has to contend with *uncertainty*. Kim (1987) and others have argued that the brief that architects are given by their clients, and which often constitutes the basis for the design goals, is much too vague, in most cases, to form a complete goal statement. Rather, it is merely a

² Meaning “good enough.” The term was coined by Herbert Simon in his book *Sciences of the Artificial*, MIT Press, 1969, pp. 35-36.

statement of *intents*, which defines a general framework for the sought solution, and some of the constraints it must abide by. Instead, they suggest that architects must gradually develop the statement of goals as they proceed with the design process itself. The additional information needed to complete the goal statement must either be *invented* as part of the search process, or *adapted* from generalized precedents, prototypes, and other relevant past experiences (so-called 'design cases'). Since the relationship between the newly invented information, as well as the precedents, to the particular needs of the problem can be discovered only as the problem becomes clearer, the adaptation itself is problem-specific, and cannot be accomplished prior to engaging in the search process itself.

Design, according to this view, is a process of *discovery*, which generates new insights into the problem. The design search process may, therefore, be compared to *puzzle-making*—the search for the most appropriate effects that can be attained in unique spatio-temporal situations through the manipulation of a given set of *components*, following a given set of combinatorial *rules*. Since architects cannot invent information from scratch in every case, they rely on design 'cases,' either from the architect's own experience or from the experience of the profession at large, to provide them with a rich pool of empirically validated information which has been refined through many years of practice and has gained society's or the profession's approval. This information comes in the form of proven solutions (Alexander et al, 1977), architectural styles, celebrated buildings, established metaphorical relationships, and recognized symbolisms (Venturi, 1972). How architects adapt this body of knowledge to the particular problem at hand is not known—it is the essence of architecture's celebrated 'intuitive leap' and creativity.

Therefore, rather than rely on a goal-driven strategy, the puzzle-making paradigm relies on adaptation of precedents, symbols, and metaphors. The main skills employed when following this paradigm are *synthetical*: the ability to compose given parts into a new, unique, whole. Such *data-driven* approaches have been computationally represented as forward-reasoning search strategies: operators are applied to the current state of the problem with the aim of transforming it according to pre-set rules. Example of tools based on this approach include generative expert systems, shape grammars, and case-base design systems (Coyne et al, 1990; Flemming, 1994; Oxman, 1992).

3. Different Kinds of Relationships between Form and Function

In this paper we argue that the relationship between Form and Function is much more complicated than implied by the causality-based notion of

‘Form Follows Function,’ and its inverse. Indeed, a particular *form* is often capable of many different functions, and a similar *function* is often afforded by many different forms. The following examples will serve to illustrate this argument.

3.1. MANY FORMS, SAME FUNCTION

The over-simplicity of the notion ‘Form Follows Function’ is evidenced by the multitude of different forms which essentially were designed to support similar functions.

Chairs provide one of the best examples of different forms that were developed to support exactly the same function (sitting). Figure 1 shows some of the many designs of dining/table chairs.

Figure 1. Chairs demonstrate how different forms can support the same function.

Design competitions, where competitors must respond to the same set of functional requirements within the same context, provide additional evidence that in architecture, form does not necessarily follow function. Each and every competitor will, invariably, produce a very different form.

Some scholars have tried to explain this apparent lack of causality by arguing that, typically, the functional requirements of a building do not tightly constrain its form, thus leaving the architect with much room to entertain ‘styles’ and other ‘non-practical’ considerations. Herbert Simon, for example, has defined style as “one way of doing things,” chosen from a number of alternative ways (Simon, 1975). Since design problems generally do not have unique or optimal solutions, says Simon, style can be used to select a solution from among several *functionally equivalent* alternatives, just as any other criteria can. He offers the following analogy:

Mushrooms can be found in many places in the forest, and the time it takes us to fill a sack with them may not depend much on the direction we wonder. We may feel free, then, to exercise some choice of path, and even to introduce additional choice criteria ... over and above the pragmatic one of bringing back a full sack (of mushrooms).

Most architects, however, would reject this notion that form is the result of less 'practical' functional considerations than other aspects of the building, and therefore an afterthought, something to be contemplated only when all the other 'important' aspects of the design have been dealt with. Rather, they would argue, it is something a competent architect will consider *before*, *during*, and *after* the development of solutions satisfying the functional needs. Moreover, the two issues cannot be separated, since each one informs the other, and influences its development.

3.2. MANY FUNCTIONS, SAME FORM

The notion that a given form can support many different functions is demonstrated well by designs of playgrounds, parks, and civic plazas. Joost van AnDEL (1988) observed that playgrounds for children between the ages 3 and 7 function best if the activities they afford are less structured, in terms of the equipment they contain. For instance, placing an old fire engine in a playground (a form) will direct the children's activities towards particular play patterns. Furthermore, van AnDEL observed that this particular form tends to create gender-biased play patterns, which appeal more to boys than to girls. On the other hand, a playground which consists mostly of a sandbox, some rocks, and a few trees or bushes affords less restricted play patterns, and is equally accessible to both boys and girls. He attributes this performance to the creative imagination of the children, who can translate the existing, generic forms into particular ones, as needed for playing games such as 'house,' 'cops and robbers,' or the landing of an alien spaceship.

Another example of architectural multipurpose (i.e., functional) spaces has been described by Elizabeth Cromley in her paper on the history and evolution of modern bedrooms (Cromley, 1990). In addition to providing a place for sleeping, bedrooms through the seventeenth century also functioned as parlors, dining rooms, and as places for entertaining guests. In the eighteenth century, the function of bedrooms became more focused, as a place for sleeping and dressing, for quiet retirement, and for socializing with close friends and family members. In the nineteenth century, bedrooms became a place to occupy only at night. In the 20th century, the definition of their function was broadened again, especially as far as children's bedrooms were concerned. Today such functions include sleeping, doing homework, reading, and playing with friends. Bedrooms for the adults (the so-called 'Master Bedroom'), have turned into 'suites,' which include full bathrooms, dressing rooms, and walk-in closets.

The ability of the same form to afford different functions is further demonstrated by what we now call *adaptive re-use*. The term designates the conversion of older buildings to meet modern needs. It is rooted in the

economic realities of the late 20th century, and the growing need for urban renewal and rehabilitation. This trend is characterized by corporations, shops, and even residential units moving into older buildings in the core of cities. Rather than tear down a building which may have some historical or cultural significance, new tenants may rehabilitate it while preserving its character. A typical case in point is Adler and Sullivan's Guarantee building in Buffalo, New York (Figure 2). In its 100 years history, this landmark building has served successfully as an office building, department store, and a variety of commercial, government, and other functions. While the building's interior has undergone some changes, the building has remained largely intact.

Figure 2. Same form, many functions: the Guarantee building, in Buffalo, New York.

3.3. OTHER KINDS OF FORM-FUNCTION RELATIONSHIPS

Peter Eisenman's structuralist approach to buildings, which derives from his own interpretation of Noam Chomsky's linguistic theories (as well as Jacques Derrida and other philosophers), demonstrates well the complexity of the possible relationships between form and function. His proposed design for the Max Reinhardt Haus in Berlin (a cultural center and office building), is modeled as a huge, three-dimensional Möbius strip—an abstract topological construct (Figure 3).

Figure 3. Arbitrary form: Peter Eisenman's proposed design for Reinhardt Haus, Berlin.

3.4. THE IMPORTANCE OF CONTEXT

The form of a building also depends upon the physical, cultural, social, and other contexts in which it is embedded, at least as much as it depends on the function it must serve. The form of the Sydney Concert Hall, which is depicted in Figure 4, is an example of a form derived from the physical context of the building (the Sydney harbor), as much as from its function (a symphony hall).

Figure 4. Context-influenced form: the Sydney Opera House.

Likewise, the shape of Le Corbusier's Ronchamp Chapel has been derived from its spiritual context, as much as from its functional and physical site considerations; and Gerrit Rietveld's colorful Schröder House in Utrecht, The Netherlands (1931), has been shaped as much by the neoclassicist cultural ideas of the De Stijl movement to which he belonged, together with painters like Theo van Doesburg and Piet Mondrian, as much as by functional requirements.

3.5. THE RELATIONSHIP BETWEEN FORM, FUNCTION AND CONTEXT

The position taken in this paper is that Form, Function and Context are linked through tri-lateral, mutual interdependencies. It would be futile, however, to look for causality among these relationships. The utility of the links can only be revealed by observing, measuring, and interpreting their overall, combined result, which is what we call *performance*. Performance evaluation is intended, therefore, to assess the desirability of the confluence of the three factors. It may reveal, for example, that a particular form is capable of supporting a certain functional need in a particular context, in which case it will be deemed 'successful.' On the other hand, it may reveal a need to modify the form to meet the desired function in the particular context, or to modify the desired function to meet the functionalities afforded by that form in that particular context.³

The complexity of this relationship is exacerbated by the fact that the *nature* of Form, Function and Context are dissimilar. Functional objectives are often *abstract*, expressed in terms of social, psychological, economic, and other behaviors. Forms, on the other hand, are often quite *specific*, and are expressed in terms of topology, geometry, and materiality. Context typically involves given social, cultural, and economic situations, in addition to physical ones (which include topography, climate, flora, etc.). To bridge the representational gap between the functionalities afforded by a given form, or the form that will afford a particular function in a particular context, the designer must rely on an *interpretive-evaluative* process, with its intrinsic fuzziness, value-laden biases and subjective belief systems. He must attempt to *predict* the functionality (behavior) afforded by the chosen form within the prescribed context, thus translating the form into a functional abstraction. At the same time, he must *envision* the form that might afford the sought function, thus translating the functional abstraction into a physical form. He must then *compare* the two abstractions to determine if they match (Figure 5). In addition, he must map the emerging composition onto the context, to determine if they match it too.

³ In some cases it may also be possible to modify the context itself, for instance by obtaining exemptions to zoning codes, modifying the socio-economic makeup of the inhabitants, or even the physical characteristics of the site.

Figure 5. Specification and abstraction in the design process.

Another complication arises from the multiplicity of *functions* that must be supported by the same form, often at the same time, as well as the multiplicity of *forms* that are needed to support a given function. Windows in a building offer a good example: they are needed to admit light and view into the building, but they interfere with the shelter-providing functions of the walls they puncture (from thermal, humidity, air infiltration, and other environmental control points of view). At the same time, they also contribute to the aesthetic appearance of the building, where their shape, pattern and rhythm may or may not correspond to their function. Consider, for example, the two functionally-equivalent, but aesthetically-different views of Le Corbusier's Villa Savoye (Figure 6). The task of the designer, in many cases, is to find a good compromise of form, function and context, rather than an optimal relationship between them (which may be the goal in certain engineering fields).

4. Performance-Based Design

Having denounced the causal relationship between Form and Function, which, as argued earlier, is the basis for the prevailing design paradigms, we must find another paradigm that will explain how designers can bridge the gap between Form and Function, and are thereby able to justify the selection of a particular Form to meet specific a Function within a particular Context.

Figure 6. Functionally-equivalent, but aesthetically-different views of Villa Savoye.

This paper suggests that such a paradigm can be formulated. We call it *performance-based design*: the specific confluence of *form* and *function* in a particular *context* (Figure 7). We define Function as the desired *behavior* of the building (or other artifact). This behavior can be quite specific (e.g., budget), or more abstract (e.g., provide a conducive environment for work). We use Form in its conventional connotation, as a physical manifestation of topology, geometry, and materiality. By Context we mean the physical, social, economic, cultural, legal, and other *settings* and *events* in which the building is located. By comparing the physical manifestation of a given building form with the conditions necessary to fulfill a desired set of behavioral characteristics of that building, within the particular context in which it is situated, we can determine the *performance* of both its form and function, relative to another composition of form and function within the same context.

This design paradigm also explains (and rationalizes) why designers may begin the search for a form-function-context composition either with a given function or with a given form, while progressing towards the other, as well as skipping around, back and forth, between the two (Figure 8). *Designing*, accordingly, can be considered an iterative process of search, where desired functional traits are defined, forms are proposed, and a process of evaluation is used to determine the desirability of the performance of the confluence of

forms and functions within the given context (Carrara et al, 1994). The search terminates when the designer finds a form that fulfills the function, or is satisfied by the functionalities afforded by the chosen form, within the given context. We call this condition *functional adequacy*: the instance when form and function come together to achieve acceptable *performance* within a given context.

Figure 7. Performance, as the confluence of form, function and context.

Figure 8. Design as a bi-directional search.

5. Implementation

To implement the proposed paradigm, a computable model of performance-based design had to be developed, which can represent explicitly *function* and *form*, as well as the *context* of the particular design project. Additionally, for the sake of computability, the *process* that helps to identify an acceptable convergence of form, function and context also had to be represented explicitly.

Through a succession of projects, beginning in 1985, we have developed such a model, which consists of four strongly related components:

1. A structured set of **goals**, representing the functional requirements that a design solution must meet.
2. A structured set of **solutions**, representing the physical and spatial components of the emerging solution (rooms, walls, windows, etc.).
3. A representation of the **context**, in which the project is embedded (physical, social, cultural, etc.).
4. A structured set of **evaluators**, whose purpose is to predict the performance of the form-function-context composition based on the physical attributes of the objects and the goals, within the particular context of the design problem.

These four components rely on different methods of representation. Goals are represented by sets of functionally equivalent *constraints*. Solutions are represented by specific *building elements* (e.g., walls, spaces, and materials). Contexts are represented through constructs we call *settings* and *events*. Evaluators are represented by a variety of methods, including simulation, case-based knowledge, and other computational means.

5.1. REPRESENTATION OF DESIGN GOALS

Many (but obviously not all) functional needs can be represented as objective *constraints*. For example, the nature, morphology, and sizes of spaces, their material composition, the equipment and furniture used in them, and the procedures for managing them can be represented as a class of *use* constraints. Desired temperatures, humidity, lighting, and other comfort parameters can be represented as a class of *environmental* constraints. The *structural* and *mechanical* behavior of buildings comprise additional sets of constraints, as do their behaviors under exceptional conditions such as fire and earthquakes, which can be represented as a class of *safety* constraints. Such classes can be further divided recursively into sub-classes, creating a hierarchical structure of increasingly more specific and detailed constraints (Figure 9). The classifications are, of course, highly arbitrary, and can be

tailored to the needs and preferences of each designer. They demonstrate, nonetheless, the ability to define design objectives explicitly, in terms of testable sets and subsets of goals and constraints. Some of these objectives are derived from the clients preferences (e.g., budget). Others may be derived from the context (e.g., views, temperature, size limitations). And still other objectives may reflect the architect's own aspirations, style, and ethical code.

Figure 9. Hierarchical, recursive relationship between goals and constraints.

A set of constraints can be used to indicate a particular combination of desired behaviors that must be accomplished by a candidate design solution in order to achieve a specific design objective. We call this set a *goal*. For example, the number of bedrooms in a single-family house determines the number of bathrooms it should have, because it is often indicative of the size of the family that will occupy the house. Likewise, the adjacencies of the rooms cannot be separated from their number and the types of activities they contain. The number of rooms and total floor area is directly related to the budget, which is also influenced by the quality of the construction.

The goals are considered to be achieved if all their constraints have been satisfied. The particular combination of constraints that is considered a goal is established when the goals are first introduced. This forces the designer (and the client) to consider and establish reasonable combinations of objectives, which then guide the design process. Additional goals may be added, or existing goals may be modified or deleted during the design process, thereby providing a measure of flexibility and a means for representing changing preferences as the design evolves.

The specificity of design goals must not be confused with the specificity of the design solutions that satisfy them. As argued earlier, different design solutions may achieve the same goal, albeit each may satisfy the constraints that comprise the goal differently. The different performance levels at which alternative sets of constraints may be satisfied represent *tradeoffs* in the context of achieving a particular goal. The different windows on Le Corbusier's Villa Savoye (Figure 6) demonstrates such trade-offs with respect to aesthetical and functional considerations, although both provide the same basic utility of light, ventilation, and view.

While alternative goals represent acceptable combinations of performance levels, some combinations may be preferable to others. A prioritization of goals, reflecting a descending order of preferences, may be imposed by the designer or by the client, indicating which combination of performances the designer should attempt to accomplish first. Such prioritization is not only a common practice when architects and clients are faced with limited resources, but it also has a very profound effect on the direction of the design search process and on its results. This is due to the fact that all the decisions leading to the specification of a design solution are connected to each other, and decisions made earlier in the process may limit the options available to the designer in later design phases, sometimes to a degree where no options are available at all. For example, choosing a particular construction method early in the design process (e.g., wooden frame) imposes many constraints on the building, limiting the options available to the architect in designing its form, details, and construction schedule.

5.2. REPRESENTATION OF DESIGN SOLUTIONS

The stated constraints can be achieved by different, yet functionally equivalent solutions, comprising building *objects*. Computationally, objects can be defined in many ways.⁴ Recently, frame-based, object-oriented methods have been gaining popularity. In addition to their computational advantages, object-oriented programming methods appear to be intuitively similar to the building objects they represent. Frames make it possible to encapsulate many of the attributes constituting an object, and they can be organized into hierarchical classes and other types of relationships according to their properties.

According to the frame formalism, the relationship between an object and its attributes is fixed. The *values* of the attributes themselves, however, are not fixed: they are *variables*. Such fixed-attribute, variable-value

⁴See, for example "Computer Integration of Design and Construction Knowledge" (Eastman) and "Intelligent Systems for Architectural Design" (Watanabe), in *Knowledge-Based Computer-Aided Architectural Design* (1994), Carrara & Kalay, eds., Elsevier Science Publishers, B.V., Amsterdam.

relationships are known as *name-value* pairs. Attributes (also known as *slots*) can be thought of as 'place holders,' or as predefined properties that are associated with particular types of values. Values (also known as *fillers*), include the permissible range of numbers (and other types of values) that can be associated with a particular attribute, defaults, and even instructions (so called 'demons') that allow the attribute to calculate its value when it depends on values associated with other attributes. For physical objects of the kind used in architectural design, there would typically be an attribute called *shape*, whose value would be a particular topological/geometrical entity describing the form of the object and its location in space relative to some frame of reference. There would also be attributes for material composition, structural properties, cost, thermal properties, and so on.

Every type of entity in the database is accompanied by operators that can create, delete, and modify it, and associate it in various types of relationships with other entities. These operators help to maintain the consistency of the information in the database, by propagating changes caused by outside operators (e.g., by the designer). We have chosen to link objects with four types of relationships, as depicted in Figure 10. They include:

- *Classification relationships*, which provides the means for associating individual objects with classes of objects of the same kind, such that shared properties can be inherited along generalization hierarchies.
- *Instantiation relationships*, which provide the means for making *instances* of a *template* object, and maintain a measure of control over the instances by automatically changing them when certain key attributes of the template change.
- *Assembly relationships*, which connect instances to each other through links that form part-whole hierarchies. Assembly hierarchies allow propagation of changes from 'parent' objects to their dependent 'children.'
- *Aggregation relationships*, which bind together objects that share some common property, or objects that must be considered together when some database changes occur, but do not fall into one of the other categories. Aggregation relationships require explicit definition of the nature of the link that connects them. This explicit definition makes aggregation a more general type of relationship than classification and assembly relationships, where the nature of the link is implied by the type of the link itself. By adding a conditional component to the definition (in the form of a set of rules), the aggregation relation can exercise the link selectively, depending on the nature of the change and the nature of the affected objects.

5.3. REPRESENTATION OF CONTEXT

We consider *context* to comprise of project-independent information that the architect must respond to in his design, and over which he has little or no control. For example, the topography of the site, its climate and views, are such information. Likewise, the cultural environment, the economic and political makeup of the society in which the project is embedded, and often building and zoning codes, comprise project-independent factors that the building must respond to. Additionally, we also include in the term *context* the predominant activities that the building (or urban place) must support, which are typically implied by the nature of the project itself. For example, the medical procedures for treating patients in a hospital, the method of teaching in a school, and even the traditional habits of a family within its own house, are factors the architect must account for in his design.

Figure 10. The schema of objects and the relationships between them.

Context thus comprises two kinds of information: settings and events. *Settings* include physical (topography and additional characteristics of the site, such as its susceptibility to earthquakes, its climate, views, etc.), cultural (built environment, customs, etc.), socio-economic, legal (codes, etc.), political, and other factors. *Events* represent the nature of the activities that will occur within and around the building. For example, in case of a restaurant, events include cooking, bringing supplies, removing garbage, parking, serving food, dining, as well as hosting birthday parties, dancing, or political rallies.

Some of the information included in the context might be translated into functional requirements, and be represented as design goals and constraints. For example, construction in an earthquake zone carries many UBC

regulations and design constraints. Likewise, designing a hospital is subject to a long list of very specific requirements. However, many contextual facts are too subtle to be formalized into explicit goals and constraints. Architects are, nonetheless, aware of them, and do respond to them in their buildings. The result is what we call a 'good' building, which 'fits' within its context. Frank Lloyd Wright's Fallingwater in Bear Run, Pennsylvania, is a prime example of contextual awareness, which cannot be formalized through goals, but has nevertheless strongly influenced the design of the house.

By its very nature, *context* is inherently difficult to represent explicitly. Architects understand context by visiting the site, photographing it, rendering it in their sketchbooks, interviewing its residents, studying historical records, and, in general, spending much time there. We have proposed, therefore, an indirect representation of context, through the goals, the solutions, and primarily through the evaluators. Goals and evaluators, in particular, provide a convenient means for representing context-specific trade-off of needs, and for including problem-specific predictors and interpreters of performance. Still, we suggest that an explicit representation of context is desired, and ought to be the subject of further research. It can be achieved in the form of scripts, narratives, case studies, photo albums, and other media that have the ability to capture the above-mentioned attributes and qualities, as well as by the knowledge the architect himself brings to the design process.

5.4. EVALUATION

We consider *evaluation* to be a process that compares what *has been* achieved (or is *projected* to have been achieved) to what *ought to be* achieved. Evaluation, therefore, can be defined as *measuring the fit between achieved or expected performances and stated objectives, within a given context*. The process of evaluation can, however, only be applied to a given, specific set of performance characteristics (form, function and context), such as the form, composition and location of a building within a given site, and intended to meet the needs of a specific client, much like medical diagnosis can only be applied to the physical condition of a particular patient, under particular circumstances. When evaluating hypothetical design solutions where performances are not yet in evidence and cannot, therefore, be assessed directly, evaluation must be preceded by *prediction*. Prediction is the process whereby the expected performance of buildings (or other artifacts) is simulated, hypothesized, imagined, or otherwise made evident, so that it can be subjected to evaluation. For example, the rate of heat loss through a given building envelope must be predicted, often by way of simulation, before an evaluative procedure can determine whether this rate is

acceptable for the activities that will take place in the building at a particular location. Likewise, the fire resistance properties of a wall or a door must be determined before its behavior under emergency conditions can be evaluated. Some building behaviors can be predicted by using established scientific methods, based on first principles. These include energy, structural, lighting, and other physical phenomena. Other behaviors lack such a scientifically rigorous base, and depend upon experience, rules of thumb, and sometimes sheer guesswork. These include color schemes, building *parties*, proportions, and other psychological and behavioral phenomena. Evaluation and prediction are, therefore, often value-based and dependent upon judgment, taste, and other subjective variables.

Such variables depend not only on the attributes of the solution itself, but also on the *context* in which it is embedded. The context-specificity of evaluation and prediction are, in fact, their most valuable characteristic for the development of a performance-based design paradigm. However, they are also the least computable aspect of the proposed implementation of the paradigm. Context, in the general sense that was discussed earlier, is hard to represent computationally and, therefore, difficult to incorporate in the evaluation procedures.

Moreover, evaluation refers not only the general suitability of the project as a whole to the stated goals and its context, but also to the suitability of the developing solution to goals and contexts that are particular to specific *phases* of the design, construction, and use of the building, and the concurrent relationships and influences of certain criteria on other criteria. For example, the disruption caused by the construction of a building (or a freeway) to its neighborhood may outweigh its benefits once completed. Likewise, the materials from which the building is made may be harmful to the environment or to the inhabitants of the building (asbestos and lead are prime examples).

Thus we distinguish between *Multi-Criteria* evaluation and *Multi-Phased* evaluation, where Multi-Criteria is an evaluation modality that examines a given design solution from *several different* points of view (e.g., energy, cost, structural stability, etc.), and Multi-Phased evaluation is a modality that examines how the design solution, or a succession of design solutions, satisfy a *particular design objective* (e.g., energy) throughout the study period (typically, the life-span of the building). To complicate matters even further, the designer must often engage in *both* evaluation modalities at the same time. Each modality informs the other, as well as the process as a whole. It is, unfortunately, very difficult to develop computational tools that can perform both kinds of evaluation, and at the same time, be cognizant of the particular context of the design problem. Most, if not all of the evaluation programs developed to date have chosen one of the two modalities (Kalay, 1992).

Nonetheless, as a methodology, we suggest that such multi-modal evaluation tools ought to be considered (Shaviv and Kalay, 1992).

6. Case Study

We developed a system called KAAD (Knowledge-based Assistant for Architectural Design), which is intended, among other things, to be a proof-of-concept for the proposed design paradigm and its implementation model. KAAD was designed to help architects specify design objectives, adapt existing or create new design solutions, predict and evaluate their expected performance, and compare them with the stated objectives within the specific context of designing health-care facilities for treating infectious diseases in Italy. KAAD is founded on a knowledge-base, comprising prototypical design solutions, which includes much of the information pertinent to generic building objects, such as walls, windows, doors and fixtures, as well as information specific to nursing units, infectious diseases suites, and other hospital-related data. The knowledge-base includes not only *syntactical* information (form, materials, cost, etc.), but also *semantic* information which explains the meaning and contextual relevance of the information (e.g., that a particular door in the proposed design violates infection-containment protocols). Particular solutions are derived from prototypes by adaptation to the specific context of the problem.⁵

To demonstrate the concepts discussed in this paper, we will consider the design of a small nursing unit intended to house two patients in the infectious diseases suite of a hospital located in northern Italy. A typical solution to the problem is depicted in Figure 11, which shows a collection of spaces superimposed with some access-control constraints: the gray arrows represent physical accessibility, while the white arrow represents visual access, but not a passage. For the sake of simplicity, the adjacencies with the outside and with other units of the hospital have not been represented here.

This case study, albeit of reduced complexity, is significant because hospital suites for treating infectious diseases must respond to a considerable number of constraints, many of which are often crucial. These include support for specific treatments of symptomatic seropositive or AIDS-infected patients, as well as the guarantee of adequate protection of the patients against the risk of crossed or opportunistic infections. At the same time the designer must also guarantee an adequate level of protection to the visitors and the staff by carefully considering paths, entrances, filter and reclamation areas and dressing rooms. Some for the stated goals of the design of such a nursing unit are:

⁵ A full description of KAAD is beyond the scope of this paper, and is irrelevant for its purpose. The interested reader is referred to (Carrara et al 1994) for more details.

Figure 11. Layout of a typical nursing unit for treating infectious diseases.

- Proper connectivity to other hospital units
- Size limitations
- Environmental constraints, such as the number of air exchanges, air velocity, air purity, relative humidity, etc.
- Specific fixtures and furniture
- Social and personal welfare requirements.

Additionally, such suites are context-specific in term of medical procedures, and culture-dependent in terms of care and visiting patterns. The particular contextual setting of the case study included the following data:

- Location—Northern Italy, not far from the Adriatic sea, implying certain climatic and soil conditions, as well as a particular socio-economical profile of the patients and their visitors
- Date of construction/use—first decade of the 21 century (which implies certain medical procedures)
- Legislative context—an infectious diseases hospital is subject to certain rules and regulations that must be observed, in addition to customary construction methods and practices.

The event-contextual data included the following:

- Patient visiting habits, implying certain amenities for non-care-givers/receivers (e.g., parking lots, lounges, cafeteria, etc.).
- Treatment methods, including contact between staff and patients, with all the attendant risks (to both parties) that are incurred from treating patients in an infectious diseases hospital suite
- Emergency egress procedures for patients, staff, and visitors.

6.1. REPRESENTATION

For the sake of simplicity (as well as other constraints, such as time and human resources), we have chosen to combine the solutions, the goals, and the context within a single frame-based representation. Furthermore, some of the evaluators have also been represented through the same formalism, by means of procedural attachments. Two kinds of generic frames have been developed:

- *Space Units* (SU), representing classes of objects that meet requirements associated with individual rooms (or their equivalents) in a hospital, such as dimensions, type of use, environmental conditions, and so on.
- *Functional Elements* (FE), representing the physical components of the building that delimit spaces and define safety and environmental comfort levels.

These entities form two hierarchical structures: Building Units (BU), which represent the spaces in the building, and Functional Systems (FS), which represent the structure of the building.

As an example, consider the implementation of a simple SU (Space Unit) prototype of a nursing room in a hospital (Figure 12). The frame includes slots that establish the spatial relationship between this particular SU and other SUs; slots that define the maximum and minimum values for certain variables; as well as slots that define the current values associated with different variables.

6.2. PROCESS

From the user's point of view, the design process in KAAD is similar to other design processes using CAD. The user may begin by drawing lines, representing walls, doors, and other building elements, or he can begin by specifying the desired goals (constraints) the building ought to achieve, in the form of bubbles representing the functional units (rooms, corridors, outdoor spaces, etc.) and the adjacencies between them. Such drafting and modeling activities are supported by KAAD in a manner similar to other CAD systems.

The bulk of KAAD's action happens in the background. Each action of the user causes KAAD to do one of three actions:

1. search its knowledge-base for a prototype FE (Functional Element) or SU (Space Unit) which matches the characteristics of the object specified by the user, and instantiate it;
2. check to see if the modification made by the user conforms to the constraints associated with the affected FEs and SUs, and report back to the user if it does not;

3. initiate a prediction/evaluation process that will provide the user with information concerning the designed artifact as a whole (e.g., energy performance, emergency egress, etc.).

(dg3 (ako value su))

(description (value "space unit for patient's
nursing room"))

(com (value conn1))

(adj (value dg6 dg7))

(far (value dg2 dg11 dg15 conn2))

(ims (value hfur3 ite))

(sup (min 22)

(unit mq)

(max 28)

(unit mq)

(description "minimum and
maximum net area")

(wtemp (range 19 21)

(unit °C)

(description "interior winter temperature")

(stemp (range 25 27)

(unit °C)

(description "interior summer temperature")

(rewh (range 40 60)

(unit %)

(description "winter relative humidity")

(resh (range 40 60)

(unit %)

(description "summer relative humidity")

(vent (value 2)

(vela (value 0.2)

(unit m/sec)

(description "air velocity")

(pura (value 4)

*Communication (part of
the routes between SUs)*

*This SU must be adjacent
to the specified SUs*

*This SU must be far from
the specified SUs*

*SU is an instance of speci-
fied prototype*

*Boundary values for sur-
face areas*

*Range of acceptable inte-
rior winter temperatures*

*Range of acceptable inte-
rior summer temperatures*

*Range of acceptable
winter relative humidity*

*Range of acceptable sum-
mer relative humidity*

*Desired number of air
changes*

Desired air velocity

Desired level or air purity

Figure 12. An example Space Unit (SU) of a nursing unit in a hospital.

The first two actions (instantiation and checking compliance with the constraints) are transparent to the designer, triggered by the expression of goals (constraints) or the specification of design solutions. Typically, not all the information needed to completely define an FE or SU instance is given

at once. KAAD follows a progressive slot-filling process, using the values specified by the designer as they become available. It automatically calculates many of the values needed to complete the instantiation, using default geometrical and material information, such as adjacencies, paths, areas, costs, and so on. Since FEs and SUs represent both goals and solutions, the two actions are conveniently similar from a programming point of view (which is, of course, why we chose to combine them in the first place).

When KAAD detects a conflict between the input and the stated constraints, it notifies the user by opening a 'warning' window (Figure 13). The user has the option to modify the input, override the constraint temporarily, or modify it (the user, in this case, is assumed to be the 'expert').

Figure 13. A typical screen of KAAD, showing a warning window.

Similarly, the user can override any number of KAAD's automatic options (including the ones normally hidden from the user), defer them, and otherwise control the actions of the system. Additionally, KAAD supports many typical CAD functions, such as grids, snapping, 2D and 3D views,

automatic handling of drawing details such as insertion of doors and windows, two-, three-, and four-way wall intersections, and a full range of geometric modeling operations. Nonetheless, every *form* that the user can see on the screen (other than drafting aids), is the graphical representation of an *instance* residing in KAAD's database, along with its full *functional* description.⁶

KAAD has been implemented by a team of 12 programmers in Italy and in the USA over a five year period. They were supported by a \$1.25 million grant from CNR (Italy's national research council). The knowledge-based parts of the system were implemented in Allegro Common Lisp 3.0 from Franz, Inc. The graphical and the database management components of KAAD were implemented in C. The user interface was developed under X11R3. The first prototype of KAAD was developed in the UNIX operating environment, on MicroVax and Tektronix workstations. Several PC versions, developed under Microsoft Windows 3.1 using Allegro CLI/PC 1.0 and Borland C++, are in final stages of development.

7. Conclusion

The development of computational tools that can truly assist humans in performing complex activities such as architectural design relies upon developing a deep understanding of the process that is to be assisted, and on casting this understanding into a model that can be represented explicitly (and thus can be translated into a computer program). Having identified the two main characteristics of architecture as Form and Function, the search for formal theories that can explain the process of design tended to converge on causality-based paradigms. Hence the attraction of statements such as 'Form follows Function,' and its converse 'Function follows Form.' These statements provided a convenient logical foundation for design theories, much like other causalities have formed the foundation of many engineering and practically all scientific paradigms.

Many architects found these logically-convenient statements inadequate to describe what their experiences taught them. These experiences were often characterized by a discontinuity in the relationship between form and function, which they called 'the intuitive leap' (Norman, 1987). This leap occurs when architects, engaged in the search for a form that will facilitate some desired function, actually find the 'right' form. The paradigm presented in this paper attempts to recognize this experience, and use it as a

⁶ As a concept-demonstration program, KAAD lacks many features that would be desirable in a more fully developed program. Particularly, it lacks means to easily extend and update its knowledge base. Changing or extending KAAD's knowledge base currently requires extensive knowledge of programming in LISP, as well as knowledge of KAAD's specific data structure. Since we have not intended KAAD to be a production CAD system, these limitations are not considered unacceptable by its design team.

basis for an alternative formal model of the design process, which can be implemented by computational means. It did not attempt to formalize the intuitive leap itself, only to accommodate it in the model. This accommodation takes the form of contextuality: the convergence of form and function in a particular context. It strives to eliminate the precedence of either form or function, and hence of the causal relationship between the two. To compensate for this, it develops the notion of *context-based performance*, as a means for interpreting and determining the confluence of the two entities.

The novelty of the proposed approach lies in considering form and function equal, hence deserving explicit representation when implemented in computational design systems, and in striving to explicitly represent the context of the design project. The particular implementation described in this paper chose to represent form, function and context in bundles called Space Units (SUs) and Functional Systems (FSs), following object-oriented, frame-based programming practices. It stands to reason, however, that a more radical separation is conceivable, where the three primary entities will be represented by entirely different means. In that case, the interpretive mechanism will have to be separated as well.

The proposed design paradigm fits well with our view that computers ought to be *partners* in the design process, tools the designer can draw upon when developing forms, specifying functions, and interpreting their confluence (Swerdloff and Kalay, 1987). The partnership approach is intended to *facilitate* design but not to fully automate it. It is based on the observation that designers are able to cope with and manage complex design processes, and have, for centuries, achieved outstanding results doing so without the aid of computers. It also eliminates the immediate need to deal with difficult and (so far) intractable computational problems such as representing the processes of *learning*, *creativity*, and *judgment* as overt knowledge structures. Rather, the partnership approach, combined with the performance-based paradigm of design, permits the designer to provide these hallmarks of architectural design himself, while drawing upon those aspects of the design process that have already been successfully computed, such as a host of analyses, visual presentations, and even certain solution-generating algorithms.

8. References

- Akin, O.: 1978, How do architects design, in J. Latombe (ed.), *Artificial Intelligence and Pattern Recognition in Computer-Aided Design*, IFIP, North-Holland, New York, NY.
- Aksoylu, Y.: 1982, Two different systematic approaches to design, *Technical Report*, University of California, Berkeley, CA.

- Alexander, C.: 1964, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, Massachusetts.
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S.: 1977, *A Pattern Language*, Oxford University Press, Oxford.
- van Andel, J.: 1988, Expert systems in environmental psychology, *JAPS10 Conference*, Delft, The Netherlands.
- Archea, J.: 1987, Puzzle-making: What architects do when no one is looking, in Y. E. Kalay (ed.), *Computability of Design*, Wiley Interscience, New York.
- Armour, G. C. and Buffa, E. S.: 1968, A heuristic algorithm and simulation approach to relative location of facilities, *Management Science*, 9(2), 294-309.
- Bijl, A.: 1987, An approach to design theory, in H. Yoshikawa and E. Warman (eds), *Design Theory in CAD*, North-Holland, Amsterdam.
- Carrara, G., Kalay, Y. E. and Novembri, G.: 1994, Knowledge-based computational support for architectural design, *Automation in Construction*, 3(2-3), 123-142.
- Coyne, R. D., Rosenman, M. A., Radford, A. D., Balachandran, M. and Gero, J. S.: 1990, *Knowledge-Based Design Systems*, Addison-Wesley, Reading, MA.
- Cromley, E. C.: 1990, Sleeping around: A history of American beds and bedrooms, *Journal of Design History*, 3(1), 1-17.
- Flemming, U.: 1994, Case-based design in the SEED system, in G. Carrara and Y. E. Kalay (eds), *Knowledge-Based Computer-Aided Architectural Design*, Elsevier Science Publishers, Amsterdam.
- Gross, M. D. and Zimring, C.: 1992, Predicting wayfinding behavior in buildings: A schema-based approach, in Y. E. Kalay (ed.) *Evaluating and Predicting Design Performance*, Wiley Interscience, New York.
- Kalay, Y. E. (ed.): 1992, *Evaluating and Predicting Design Performance*, Wiley Interscience, New York.
- Kim, M. K.: 1987, Development of machine intelligence for inference of design intent implicit in design specifications, in Y. E. Kalay (ed.), *Computability of Design*, Wiley Interscience, New York.
- Kuhn, T.: 1962, *The Structure of Scientific Revolutions*, University of Chicago Press, Chicago.
- Lenat, D. B. and Feigenbaum, E. E.: 1991, On the thresholds of knowledge, *Artificial Intelligence*, 47(1-3), 185-250.
- Laird, J., Rosenbloom, P. and Newell, A.: 1986, *Universal Subgoalting and Chunking*, Kluwer Academic Publishers, Boston, MA.
- Oxman, R.: 1992, Multiple operative and interactive modes in knowledge-based design systems, in Y. E. Kalay (ed.), *Evaluating and Predicting Design Performance*, Wiley Interscience, New York.
- Norman, R. B.: 1987, Intuitive design and computation, in Y. E. Kalay (ed.), *Computability of Design*, Wiley Interscience, New York.
- Simon, H. A.: 1969, *The Sciences of the Artificial*, MIT Press, Cambridge, MA.
- Simon, H. A.: 1975, Style in design, in C. Eastman (ed.), *Spatial Synthesis in Computer-Aided Design*, John Wiley, New York.
- Simon, H. A.: 1979, *Models of Thought*, Yale University Press, New Haven, CT.
- Shaviv, E. and Gali, D.: 1974, A model for space allocation in complex buildings, *Build International*, 7(6), 493-518.
- Shaviv, E. and Kalay, Y. E.: 1992, Combined procedural and heuristic method to energy-conscious building design and evaluation, in Y. E. Kalay (ed.), *Evaluating and Predicting Design Performance*, Wiley Interscience, New York.
- Swerdlhoff, L. M. and Kalay, Y. E.: 1987, A partnership approach to computer-aided design, in Y. E. Kalay (ed.), *Computability of Design*, John Wiley and Sons, New York.
- Venturi, R., Scott-Brown, D. and Izenour, S.: 1972, *Learning from Las Vegas*, MIT Press, Cambridge, MA.