21. BAU: A Knowledge-Based System for the Investigation of a Basic Architectural Unit

Herbert Kramel[1], Chen-Cheng Chen[2]

Department of Architecture
Swiss Federal Institute of Technology, ETH Hönggerberg
CH 8093 Zurich, Switzerland

The control of incremental complexities within an evolutionary design process has been a serious concern in both architectural education and practice. One method of examining this problem is to first define a "basic architectural unit" and a design environment which is composed of multiple units. Different levels of detail will be added to the unit as the design process continues. Secondly, a related computer program called BAU is introduced, which demonstrates that a computer is a meaningful tool for helping the architect to investigate the consequence of a design problem. Thirdly, both the domain expert's and the knowledge engineer's experiences during the development of BAU are described. Finally, the future direction of this research will be discussed.

Introduction

The notion of anonymous architecture (Moholy-Nagy 57) together with the concept of self-organizing processes (Alexander 79) as basic phenomena in architecture have changed the perception of architecture in the 20th century in a rather unobtrusive yet very fundamental way. A further expansion of these basic notions took place when the observation of basic types, and the consequent development of typology, were introduced into the field of anonymous architecture. Together, self organization and typology form powerful concepts for the understanding of human habitat and its settlement patterns as well as the generic processes of both phenomena.

The idea of a small, low key research test project, relating to anonymous architecture came into existence during a chance meeting of a group of people involved in CAAD. A suggestion that concepts developed in AI techniques may be useful for exploring aspects of anonymous architecture guided us in the initial formulation of this research project. It was assumed that similar to cells in a living organism, basic units could be defined in anonymous architecture which again like cells in an organism form clusters or aggregations of larger functional entities such as a house or habitat (Alexander 77, Steadman 79).

Starting with this notion, we defined for our own work a "basic architectural unit" in the form of a cube, since the cube is both in quantitative as well as qualitative terms highly defined and relates well to the CAD environment. In addition to this, the total number of units[3] in an aggregation, their functions - or attributes, as well as the environment (as a geometrically structured field) were defined. Following the definition of basic conditions, a process was set in motion which can be seen either as a design process based on AI

methodologies or as a simulation of a process leading to a typologically defined "population" of aggregates (comparable to the habitations we find in ecological niches, such as islands or mountain valleys).

This project, if the goals set out at its inception (described above) are pursued to their logical conclusion, will provide additional insight into the nature of self-regulating processes in architecture. It will provide a better understanding of the processes which generate the human habitat. Furthermore it will inform us about contemporary housing problems in developing nations since they can not be seen in the framework of a traditional projects environment. In an "architecture without architects" (Rudofsky 88) the principles of "classical" or "formal" architectural design do not apply. Finally, it can be expected that the project will provide new insights into the nature of possible CAAD application in an area which does not compete with the architectural profession but rather fills a gap where architecture is needed but the architect can not be afforded.

It is quite clear that there are no easy solutions. Yet we believe that this modest research project points in a direction in which, up to now, very little intellectual energy has been applied. The computer with its power of simulation could make substantial contributions in this area. Additionally, this project could show us a new direction in design education where, until now, only the area of "Architettura Maggiore[4]" has been addressed. "Architettura Minore" or Anonymous Architecture is the field which concern the majority of the urban population in developing nations today.

Methodology

Although design has been classified as a complex human activity, it is only recently that concepts and methods for capturing, organizing, and using design expertise have emerged from research in artificial intelligence and cognitive psychology. To generate new insights into the design process, this research incorporates *knowledge engineering* techniques (Rychener 88) to investigate the architectural design process. The primary purpose of this study is to explore the incremental complexity within a design process, the secondary purpose to specify the knowledge that the architect used for solving a design problem and the utilization of that knowledge in a design system (Hayes 83). BAU is a computer-aided design system which has been developed for these purposes. Design has been modeled within the framework of a general theory of problem-solving[5] (Newell 72). Several prescriptive models related to *design computation* have been proposed, such as, *near decomposable hierarchical systems* (Simon 69), *design model of functional reasoning* (Freeman 71), *generative systems* (Mitchell 77), *general design theory* (Yoshikawa 87), and *prototype refinement* (Gero 88). According to these approaches, design problem-solving can be characterized as a process of searching through alternative states of the representation in order to discover states that satisfy specified criteria. BAU is related to the idea of design problem-solving (Kalay 85). The development of BAU is aimed at defining a proper knowledge representation with which to manipulate the process of transforming a design problem into solutions.

This research is also related to *shape grammars* (Stiny 80) and *space synthesis systems* (Eastman 75), however, this research will only be described according to the knowledge engineering paradigm. In this section, we introduce the basic architectural design object, and

our methodology for controlling and investigating the design process within the BAU design environment.

The Basic Architectural Units

We defined a "basic architectural unit" as a cube that is initially 3 x 3 x 3 meters. A unit is a conceptual building module. The four vertical faces of the defined cube can be thought of as four walls, the bottom face as the base, and the top face as the roof of a building module. The allowable combinations of the units are governed by a set of geometric cubic relationships and design rules defined by the domain expert. As the levels of abstraction increase, the variations generated by the units becomes an extremely large sets of design alternatives, too numerous for a human designer using traditional methods to produce. BAU has been developed in order to assist the architect in the selection of satisfactory alternatives based on the design rules he defines, and to help him control the incremental complexity of the design process. The design process incorporated in BAU is a reflection of the domain expert's preferences. BAU is developed as a *rule-based design system* (McDermott 80). The development of BAU incorporates the following three alternate processes:

1. Possible design alternatives are generated based on the knowledge available about the current unit.
2. The domain expert's design rules are applied to the generated alternatives, and only those solutions, which comply to the rule are retained.
3. The unit is redefined as a more sophisticated unit for the next generation.

In the first phase, BAU generates configurations of basic units according to geometric relationships which allow units to be located, removed, or replaced within the design environment. At higher levels of development, the allowable combinations of units are governed by sets of design rules defined by the domain expert. Detailed operations are described in the following sections.

BAU: System Overview

To examine the complexities within a design process, we should discuss at least two basic phenomena of design: *design process* and *design object* (Simon 75). An architectural design process involves the transformation of one physical situation to another, which is a large sequence of smaller design processes. The sequence could conceivably never be complete. The efforts of these transformation focus on the "design object", which is the result of the design process.

Broadly speaking, BAU has been developed as a *top-down refinement* paradigm. Since the problem space generated by BAU is large and complex, it has to be decomposed into pieces or subproblems which are small enough to solve. Solutions to subproblems have to be integrated into an overall solution. The idea of top-down refinement is to start with the highly abstract initial problem specifications and to refine them by adding details and by decomposing them to the point at which primitive operators are available to complete all of the subtasks that are required (Dixon 89). However, because *generate-and-test* cycles are also necessary for each intermediate subtask, *bottom-up composition* is incorporated at the local level of the BAU process.

The design process of BAU can be considered as a state-space search process in which the designer attempts to locate satisfactory choices within an extremely large set of design alternatives. A hierarchy is established at the uppermost level and the general functional requirements for the design units are stated. The abstract refinement process is precompiled into a standard structure of parameterized components, and the design process consists of reasoning about constraints in order to determine appropriate parameter values.

Development of BAU

BAU involves a whole sequence of generate-and-test cycles. The first process generates certain design objects that are candidate solutions or components of solutions. The second process tests whether or not a given candidate satisfies a design constraint. In such a generate-and test process, the design is assembled component by component. Each generated component is added to the previous assembly and the new structure then tested.

BAU contains four subtasks, BAU.I, BAU.II, BAU.III, and BAU.IV. BAU.I designs two dimensional configurations for BAU. BAU.II extends a unit into a three dimensional unit which contains wall elements, slanted roofs and interior level changes. BAU.III introduces modularized elements to the units and associated construction rules to the environment . BAU.IV constructs four bau-houses in a small housing cluster. The following describes what each of BAU's subtasks involves.

Subtask I (BAU.I)

We begin by positioning the units on a plane. The chosen environment (building site) for this study, in order to be compatible with the concept of the "basic architectural units", is a 3 by 3 square grid, with each square measuring three meters by three meters (Figure 1.A.).

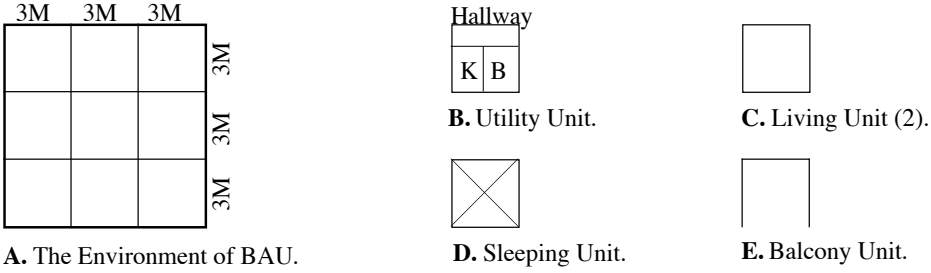


Figure 1. The Basic Architectural Units and their environment.

The unit is only considered as a two dimensional square in BAU.I. Four types of functional units, each three meters square, are chosen: a utility unit, consisting of a kitchen, bathroom, and hallway, represented by the U-Unit shown in Figure 1.B, a living unit, a sleeping unit, and a balcony unit, represented by the L-, S-, and B-Units in Figure 1.C-E respectively. There are eight possible U-Unit configurations, which can be obtained by rotating the standard U-Unit and its symmetric equivalent in ninety degrees increments. (Correspondingly, there are four possible B-Unit configurations.) The design task is to locate one U-Unit, S-Unit, and B-Unit, and two L-Units in the 3 by 3 grid.

Table 1. Rules for describing the BAU environment.

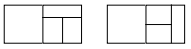
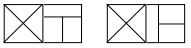
Rules for BAU and its Environment
Rule BE-1: Each unit must be located exactly on a single cell of the 3 by 3 grid.
Rule BE-2: Two connected units must share an edge.
Rule BE-3: All units must be connected.

The relationships between the units and their environment are governed by the three fundamental rules described in Table 1. Rule BE-1 and BE-2 confine the search space of BAU into a manageable size. Rule BE-3 imposes a basic formation on the house as a response to inclement weather. Based on these three BE rules, the architect develops more sophisticated two dimensional design rules for BAU.I. The design rules of BAU.I are classified into three categories. There are rules for: internal connections, external orientations, and ranking.

Rules for Internal Connections

The first category of design rules is "rules for internal connections", which specify the functional relations between units. Table 2. lists seven rules for internal connections. The sequence of these rules is determined by first the simplest units (L-Units), in order to reduce the search space of BAU.I. Rule IC-1, IC-2, and IC-3 specify spatial relations between units. The architect's contention that: a hallway between the living units and the sleeping unit is necessary for a house is the basis for Rule IC-4 and IC-5. Rule IC-6 and Rule IC-7 are imposed by the definition of B-Unit.

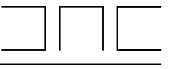
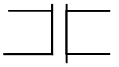
Table 2. Rules for internal connections.

Rules for Internal Connections	
Rule IC-1: The two L-Units have to be connected.	
Rule IC-2: The B-Unit has to be next to an L-Unit.	
Rule IC-3: The kitchen has to be next to an L-Unit.	
Rule IC-4: The hallway has to be next to the S-Unit.	
Rule IC-5: The hallway has to be next to an L-Unit.	
Rule IC-6: The openning of the B-Unit should not be connected to an L-Unit.	
Rule IC-7: The openning of the B-Unit should not be connected with an S-Unit or a U-Unit.	

Rules for External Orientations

The second category of design rules for BAU.I, "rules for external orientation", are listed in Table 3. These two rules specify the possible orientations of the balcony. The architect's reason for Rule EO-1 is that a balcony facing north cannot get any sunlight in winter. Rule EO-2 is based on the idea that proper orientation of the balcony will reduce undesirable noise from the street.

Table 3. Rules for external orientations .

Rules for External Orientations	
Rule EO-1: The opening of the B-Unit should not face to north.	
Rule EO-2: The opening of the B-Unit should not face the street.	

Ranking Rule of BAU.I

BAU.I produces a number of solutions. The development of ranking rules enables the architect to judge these configurations based on his own criteria. One of the ranking rule is shown in Table 4. Instead of specifying what should or should not be, this rule describes what would be better. This rule has been translated into a set of ranking values shown at the right side of Table 4. The program sums up these ranking value of empty cells, a configuration with a higher score indicates it is more proper according to the architect's criteria. The score ranges from 7 to 11. Moreover, multiple ranking rules could be applied to BAU, and optimal solutions could be found based on the architect's preferences (weighting factors and objective functions) for each ranking rule. However, we do not intend to investigate the problem of optimization in design, we'll leave the problem at this point. At the consecutive stages of BAU, different levels of detail will be added to the units based on the results of BAU.I.

Table 4. Ranking rule of BAU.I.

Rules for Ranking	
Rule RK-1: Locating garden at the south of site will be better.	2 1 2 2 2 2 3 3 3

Results of BAU.I

A tic-tac-toc like algorithm is applied to produce the first set of configurations based on the BE Rules. The total number of this set of configurations is 49. Figure 2. illustrates 14 representative configurations.

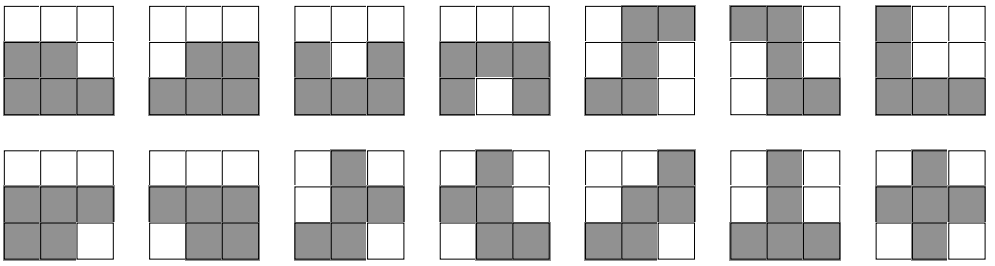


Figure 2.14 representative configurations generated by the BE rules.

A *significant configuration* is a combination of five units, representing the four functional units and the B-Unit, located on the 3 x 3 grid according to the internal-connection and external-orientation rules that were specified by the domain expert. BAU.1 generates 216 significant configurations, the first 72 of them are shown in Figure 3. Part of the results of Rule RK-1 are shown in Figure 4. The configurations at the top of the figure, which have a higher score (11), are ranked higher than the configurations at the bottom.

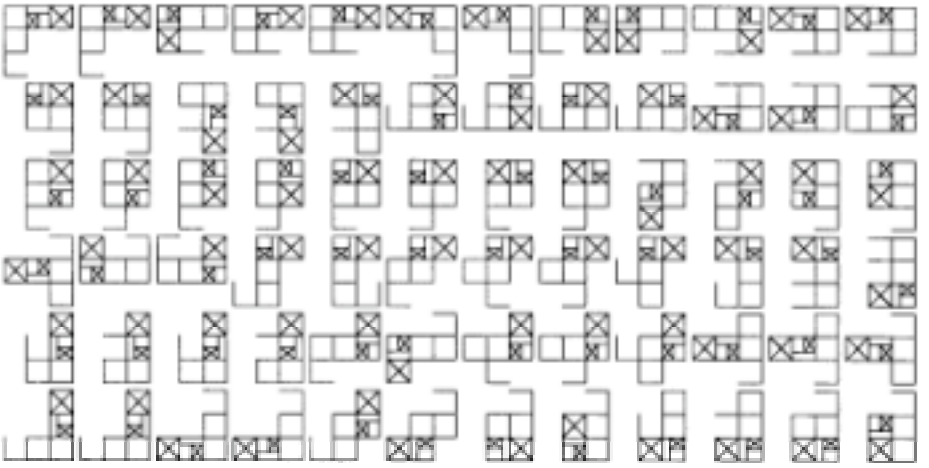


Figure 3. 72 of 216 significant configurations generated by BAU.I.

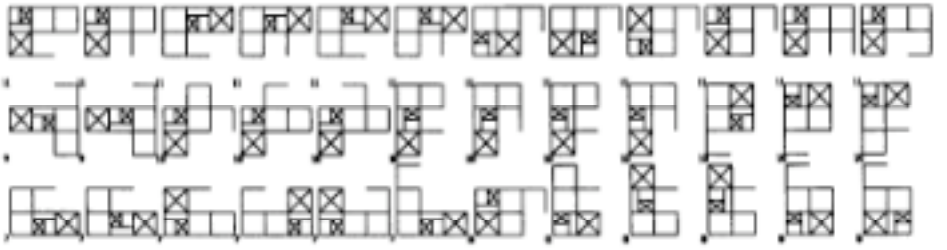


Figure 4. 36 of 216 ranked configurations based on Rule RK-1.

Subtask II (BAU.II)

The next step, once an effective method of generating significant two dimensional configurations had been developed, was to establish another set of design rules which would allow us to explore the significant configurations in three dimensions. In BAU.II, the unit is extended into a three dimensional cube. The interior level changes and slanted roofs are introduced to define a second type of unit, which is shown in Figure 5.

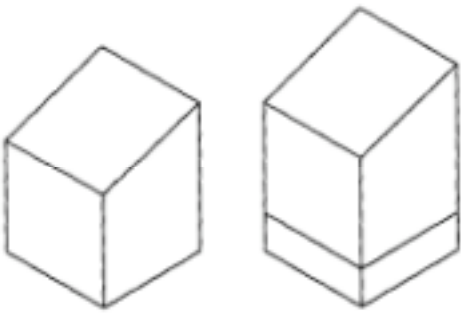


Figure 5. Second type of basic architectural unit.

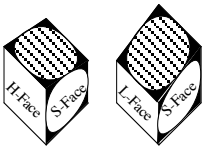
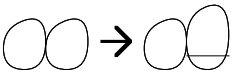
There are eight possible configurations for this set of units, which can be obtained by rotating the standard unit by ninety degree increments and shifting between the interior level changes of one meter and none. The interior level changes make the interior space and the design problems for BAU more interesting.

Rules of BAU.II

The complexities of a unit increase dramatically when the design tasks move from a two dimensional plane to three dimensional space. There are a few important repercussions resulting from the transformation into 3D. First, the problem space BAU encounters becomes extremely large. Each single significant configuration of BAU.I has 1024 (4^5) alternatives for roof combinations. Secondly, we are able to discuss the semantic level of the design problem , e.g., the visual impact of the design, which is no longer restricted to a syntactic level. Thirdly, the rules for satisfactory combinations of roofs are much more difficult to describe and to derive, and translation of the design rules into computer code is more complicated. These issues also suggest that design in the third dimension is one of the most significant aspect of architectural design. These implications are explored in the following subsections.

This set of rules is based on the need to provide satisfactory water run-off between roofs and visual appearance. These rules are not easy to describe. During the *knowledge acquisition* of BAU.II, the domain expert drew these rules graphically in order to explain his ideas to the knowledge engineer. To make the rules more comprehensible, a notation about the second type of unit is provided at the top of Table 5.

Table 5. Design rules for roofs.

Rules for Roofs	
Notations of 2nd type of bau.	
Rule RF-1: If two units are adjacent, then the L-Face of one unit should not attach to the L-Face or S-Face of another unit.	
Rule RF-2: If three units are in a row, then the L-Face of the middle unit should not attach to the H-Face of another unit, and the H-Face of the middle unit should not attach to a L-Face of the other unit.	
Rule RF-3: If two units are adjacent and one unit's L-Face is attached to another unit's H-Face, then the first unit should be raised 1 meter.	
Rule RF-4: If two adjacent roofs have the same height, then they should share either an H-Face or an S-Face.	

Results of BAU.II

After the completion of the cycle related to Rule RF-3, each single significant configuration of BAU.I has about 100 to 200 alternatives for roof combinations. Partial results of Rule RF-3 are shown in Figure 6.

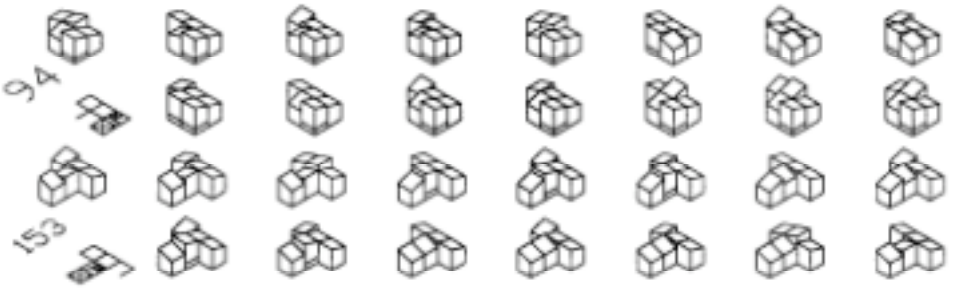


Figure 6. Partial results generated by Rule RF-3.

Although Rule RF-4 is stated quite succinctly, it was the most complex and difficult rule to translate from the domain expert to the knowledge engineer and then to the machine. The architect knew what represented satisfactory and unsatisfactory roof combinations, but had trouble distilling a complete rule to describe his criteria. We worked a couple of weeks to solve this problem, and to derive Rule RF-4.

Figure 7. Results of configuration No. 158 generated by Rule RF-4.

In Figure 7, only 15 significant roof combinations are generated out of 1024 possible combinations. The architect could contrast these results with the two dimensional configuration produced in BAU.I (lower-left configuration in Figure 7) to derive the next level of design reasoning. This research, however, has not continued to this level. This problem will be investigated in our future research.

Subtask III (BAU.III)

The third type of unit is extended into a more complicated and comprehensive building unit, which has different types of windows, interior walls, exterior walls, base, columns, and roofs, depending on the different functional types of the units. One of these unit types is shown in Figure 8.

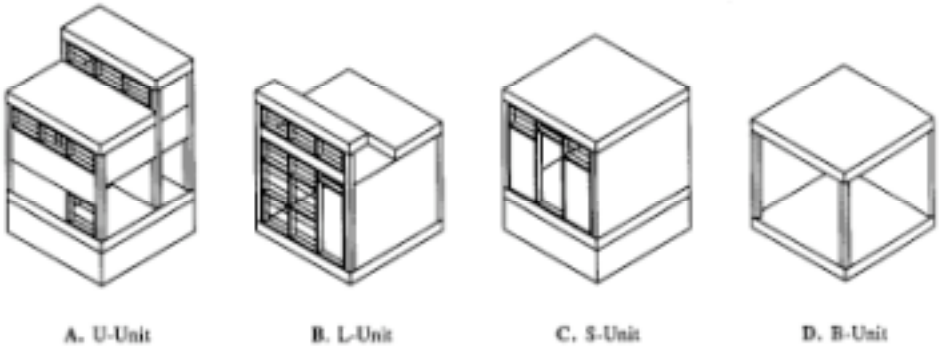


Figure 8. Third type of basic architectural unit.

The major difference between BAU.II and BAU.III is that the units have been pre-compiled into a set of parametric objects in BAU.III. The primary effort of BAU.III is focused on first parameterizing the units, then reasoning about the relations among different parameters of the units. Since the third type of unit contains more building components, the parametric objects are pre-stored in a graphical database. BAU.III generates a descriptions of bau-houses based on the relationships between units defined by the architect. The generated description activates some parameters of the component units and produces results, which can also be used for the structural analysis, cost analysis, and the prediction of energy consumption or for further design analysis.

Rules of BAU.III

Rules for BAU.III address the relationships between adjacent walls. The design decisions are based on the internal circulation and the locations of windows, doors, and interior walls.

Table 6. Design rules for adjacent walls.

Rules for Adjacent Walls
Rule AW-1: Two adjacent L-Units have to share an S-Face.
Rule AW-2: U-Unit has to share an S-Face with an L-Unit.
Rule AW-3: The opening of the S-Unit should be across from the face attached to the U-Unit.

Results of BAU.III

For each significant configuration of BAU.I, 1 or 2 three dimensional combinations are generated by BAU.III. In the first phase of BAU.III, we were interested in developing different types of roofs and skylights. Figure 9. presents part of the generated results.

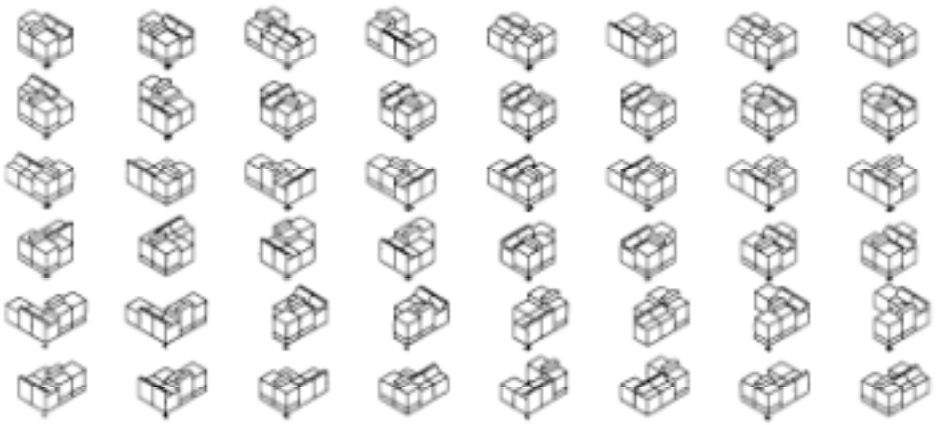


Figure 9. Partial results generated by BAU.III.

In the second phase of BAU.III, detailed information is added, e.g., columns, beams, stairs, doors, and windows. Figure 10. presents four representative combinations.

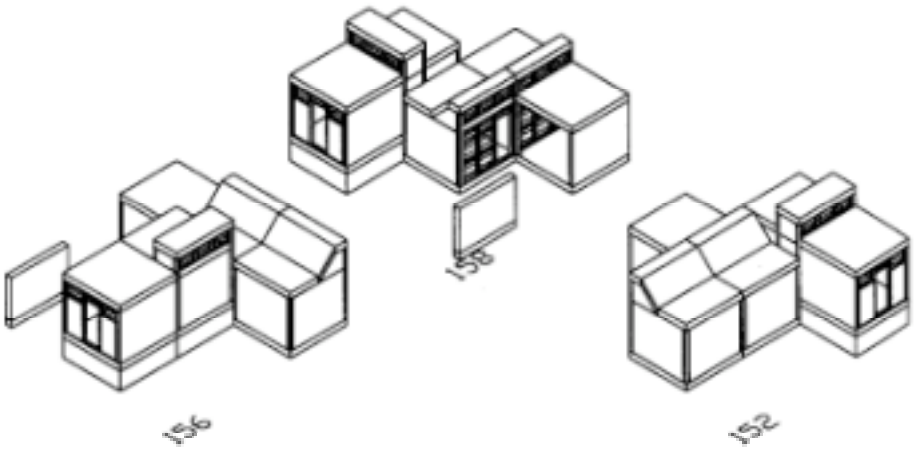


Figure 10. Detailed bau-house samples generated by BAU.III.

Different Levels of Abstraction

We have shown the sequence of how a basic unit is transformed from a two dimensional square into a three dimensional building object, and the compositions of different levels of units in the design environment. Although the units have been compiled into a complicated building block, different levels of information about each unit have been stored in different layers during the design objects compilation. This information is available to the architect, it allows the architect to examine the design object taking advantage of various level of abstraction (Figure 11).

Current Development of BAU (BAU.IV)

Our current research interest is the investigating of the concept of "a basic architectural unit" in a larger scale design environment. The design environment of BAU.IV is defined as a 9 by 9 grid, and the design task is to locate four generated bau-houses on the grid. This extension emphasizes the relations among multiple bau-houses. In addition, new design elements are introduced, e.g., compound walls, trees, streets, street lamps, fountains, etc. The development of BAU.IV enables us to re-examine the potentials and limitations of our methodology. Retrieval and combinations of bau-houses are the most crucial parts of this stage of work.

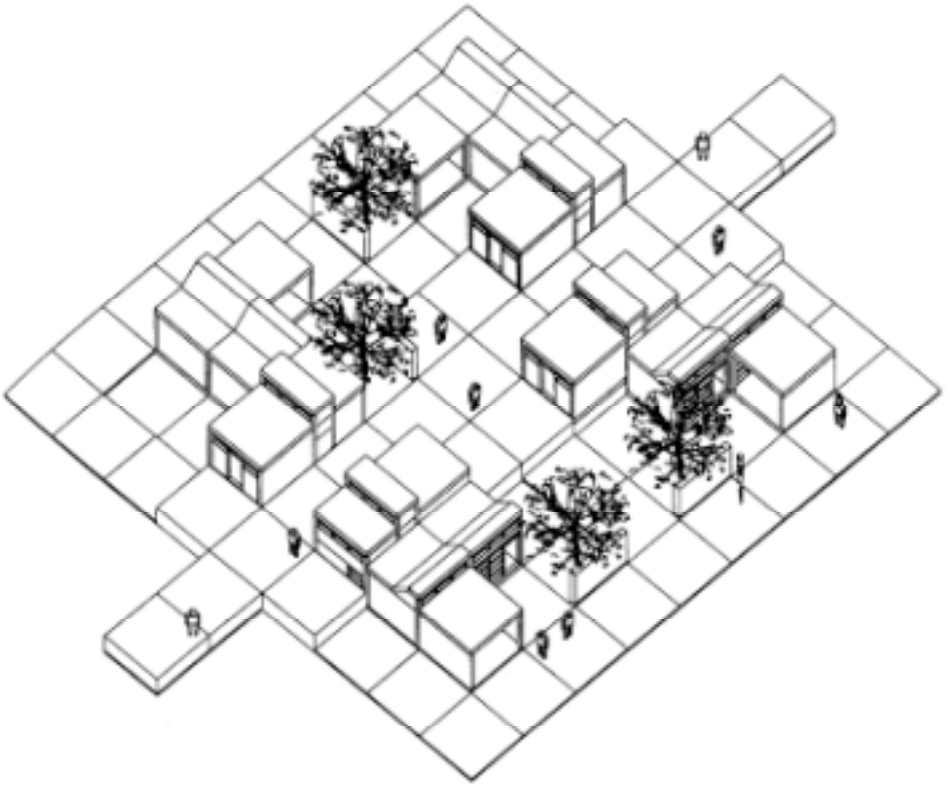


Figure 11. Current development of BAU.IV.

Since BAU.IV is still under development, only partial results of BAU.IV are presented in this paper. Figure 12. shows a bau-cluster generated by BAU.IV.

Experiences Related to BAU

The present section discusses our experiences in the development of BAU. Knowledge representation, control strategies, programming hardware and software, and the project history of BAU are addressed.

Knowledge Representation

The units and their relationships which comprise the design environment are defined explicitly and can therefore be translated easily into descriptions which a computer can manipulate. Two types of information are important for BAU In order to express the information about the units at different levels and the relationships between the units, a declarative form of representation is necessary. Additionally, a uniform representation for symbolic and graphical

representation is an important issue. *List* representation is selected because of its flexibilities in knowledge representation.

Each configuration of BAU is represented as a list containing nine elements. The first element of a bau-list occupies the south-west corner of the environment, and the last element of a bau-list occupies the north-east corner of the environment. At the primitive level, each element of the bau-list contains a binary number, 0 or 1, "0" indicates an empty cell and "1" indicates an occupied cell. As the process continues, each element which is "1" in the bau-list is replaced by the symbol "B", "L", "S", or "U" to indicate that the occupied cell has been replaced by a functional unit. In the later refinement process, each symbol can be extended into a more complicated sub-list which contains more information about the unit.

Table 7. Knowledge representation of the basic architectural unit.

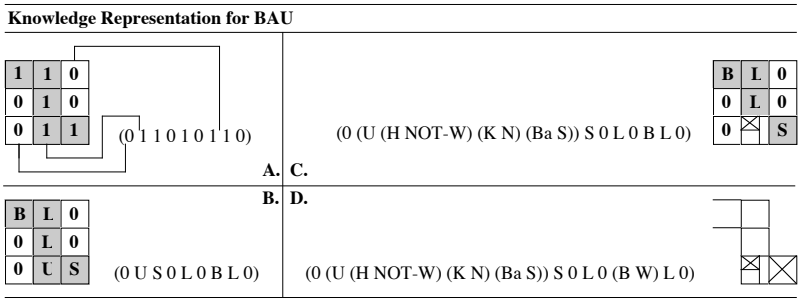


Table 7. presents the manipulation of a bau-list during a BAU.I cycle, for a two dimensional configuration. The graphical routines read in the bau-list and display the graphical results on the screen (Table 7.D). The same concept of list representation can be extended to the third dimension, information about roofs, and columns, etc, is added to each sub-list. The knowledge representation of BAU also imposes rules about how an abstract design element can be converted into a real building entity and visualized.

Control Strategy

BAU separates knowledge according to the way it is to be processed. Two sorts of knowledge are retained: knowledge for generation and knowledge for testing. Both types of knowledge can be expressed by rules: constructive and restrictive rules, corresponding to generators and filters (Takala 87). The first sort knowledge is important for maintaining and handling the additional information as the levels of detail increase. The second sort contains the heuristics of the domain expert. The distinction between these two types of knowledge is similar to the distinction between the working memory and production memory elements of a production system[6]. The generate-and-test method is applied to generate the new working memory element and then to test it against the production memory. It is, however, uneconomical to use the exhaustive method to generate the problem space, especially when each problem state contains not just a single symbol. To avoid an exhaustive generate-and-test process, a more efficient search for the significant results is obtained by imposing constraints on the intermediate stages as they are generated. The application of constraints is the most significant part of this project.

Problem Space and Solution Sets

Although it is hard to estimate how much information an architect maintains within a design process and how complex a design problem is, the systematic investigation of BAU may allow us to examine these problems based on the generated problem space and derived solution set.

The possible problem space[7] of BAU.I is ((9! / (5! x 4!)) x (5! / 2) x 8 x 4), (= 241,920). The number of acceptable solutions for BAU.I (according to the rules defined by the domain expert) is 216, the acceptable solutions represent a percentage of 0.089% (1 / 1,120). The possible problem space of BAU.II and BAU.III is (241,920 x 4^5 x 2), (= 495,452,360). The number of acceptable solutions for BAU.II is 5,765, representing a percentage of 0.00116% (1 / 85,941.43). The number of acceptable solutions for BAU.III is 270, which is 0.0000055% (1 / 1835008.74). It is surprising that an architect can find a design solution in such a huge problem space, and how quickly the constraints prune down the search tree into a manipulatable size.

Table 8. Problem space generated by BAU.1.

Rules	No. of Solutions	Rules	No. of Solutions
Rule-BE-1	C (9,5) = 126	Rule-IC-4	608
Rule-BE-2	102	Rule-IC-5	224
Rule-BE-3	49	*B-Unit is assigned.*	224 x 4 = 896
Units are assigned.	49 x (5! / 2) = 2940	Rule-IC-6	896 x (3 / 4) = 672
Rule-IC-1	1272	Rule-IC-7	592
Rule-IC-2	840	Rule-E0-1	444
U-Unit is assigned.	840 x 8 = 6720	Rule-EO-2	216
Rule-IC-3	2048	Rule-RK-1	216

Software and Hardware

Before BAU was developed, it would have been difficult to estimate accurately the problem space and the proper knowledge representation. There is a trade-off between using a procedural language (C Programming Language) and a declarative language (LISP), the former has its advantage in terms of computational efficiency and the latter is good because of its flexibility in knowledge representation. Since incremental complexity is the most significant factor in our problem, LISP was chosen.

The design kit for BAU was first developed using COMMONLISP. AutoCAD and AutoLISP, the computer graphics software system currently being used for the second phase of our development, enable the programmer to define instances, create modules by combining transformed instances, and display the results in two or three dimensions. BAU runs on both IBM RT and SUN4 workstations, under the UNIX operating system. The results of BAU.IV are sent to a special 3D graphics program called STALKER written with

the GL graphics library, that runs on Silicon Graphics Workstations. This allows the architect to study the volume, material, and color of bau-houses, and to "walk through" the building site in real time.

Project History

We started the discussion of this project in March 1990. Since then, we have had weekly meetings between the domain expert and knowledge engineer. Each meeting lasted about one hour. During the meetings, the domain expert drew his ideas on paper, and explained his ideas to the knowledge engineer. After the meetings, the knowledge engineer tried to translate the domain expert's ideas first into rules, and then into computer code. Each subsystem of BAU took approximately two to three months to develop.

An earlier project (Roger 78) had first addressed some issues relevant to this project. Two major conceptual problems in the approach taken earlier were (1) generation of the problem space, and (2) acquisition, formalization, and integration of domain knowledge.

Conclusions

BAU has proven that the computer is a meaningful tool for investigating design problems which can be specified by well-defined objectives operating on pre-determined variables or features. Through the development of BAU, the design process becomes directly accessible for detailed examination. It also suggests that there could be a possibility for architectural education to incorporate systematic research-based theory and technique within the studio system. Research and education are linked by an exchange in which the researcher offer theories and techniques applicable to practical problems and the educational environment, in return, offers researchers new problems as well as practical tests of research results. There are several important lessons that we have learned from BAU:

1. There are some areas of architectural design, in which problems could be clearly defined, goals could be relatively fixed, and phenomena lend themselves to the categories of available theory and techniques.
2. The design requirements for solutions can be expressed by some combination of constraints (design rules) and criteria (design goal). In this case, the design problem-solving process can be viewed as a state-space search process.
3. In design, exhaustive search can be avoided through a proper ordering of constraints. This is an important factor of design expertise.
4. Through the study of the architect's problem solving behavior, the acquired design model and strategies can be used in automating part of the design process.

There are several questions that remain unanswered in this research: How does a designer accept a design solution in such a large problem space? How can a designer tune a mediocre design scheme into a good design solution? How does a designer structure his design problem before he starts his design problem solving? What are the differences between experts and novices (students) in solving this type of design problem? How does a novice learn design? How does one develop a general design tool for this type of design tasks? These questions are interesting and probably could be examined by incorporating BAU with cognitive psychology studies of design.

There are several possible directions for our future work with BAU. Aside from that already mentioned in 3.3.4., i.e. combining several bau-houses in a larger environment, the indexing and retrieval of existing results of BAU are important. Additionally, we'll introduce 2 by 4 standard wooden structures into the BAU system, to investigate BAU at more detailed levels. Moreover, as part of the techniques of system architecture, computer-aided manufacturing can be incorporated into BAU.

The final consideration is how to extend the acquired experiences of BAU to other designs problems. In this regard, two problems should be addressed: First, given a design problem, the issue arises of utilizing solutions of previous problems which are known to be similar to the current problem. Secondly, once a solution for a specific design problem has been found, the question of how to apply this process knowledge to a class of similar design problems appears (Mostow 85). *Analogical* and *inductive reasoning* are suggested respectively for those two problems in our future research of knowledge acquisition and machine learning in BAU.

The results of this investigation demonstrate some of the possibilities offered by computers for exploring more fully the consequences of design decisions. We anticipate that BAU can be augmented to address new research issues in the incorporation of knowledge-based design systems in architecture, computer-aided architectural manufacturing, and robotic task planning.

Acknowledgements

We wish to express our sincere thanks to Bharat Dave, Sheron Refvem, Shen-Guan Shih and Prof. Gerhard Schmitt for their recommendations during the development of BAU. This research is supported by the Department of Architecture, at the Swiss Federal Institute of Technology.

References

Alexander, C., Ishikawa, S. and Silverstein, M. 1977. *A Pattern Language*. New York: Oxford University Press.

Alexander, Christopher. 1979. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.

Dixon, John R. 1989. "On Research Methodology Toward A Scientific Theory of Engineering Design". In Newsome S., Spillers, W. R., and Finger S. (Eds), *Design Theory 88*'. New York: Springer-Verlag.

Eastman, Charles M. 1975. *Spatial Synthesis in Computer-Aided Building Design*. London: Applied Science Publishers.

Freeman P. and Newell, A. 1971. "A model for Functional Reasoning in Design". In *Proceedings of the Second International Joint Computer Conference on Artificial Intelligence*, British Computer Society, London.

Gero, J., Maher, M.L. and Zhang, W.G. 1988. *Chunking Structural Design Knowledge as Prototypes*. The Architectural Computing Unit. Department of Architectural Science, University of Sydney, Sydney.

Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. 1983. *Building Expert Systems*. Reading: Addison-Wesley Publishing Company.

Kalay, Yehuda E. 1985. "Redefining the Role of Computers in Architecture: from Drafting/Modelling Tools to Knowledge-Based Design Assistants". Computer Aided Design.Vol. 17, No. 7, September.

McDermott, J. 1980. *R1: A Rule-Based Configurer of Computer Systems*. Tech. Rept. CMU-CS-80-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh.

Mitchell, William J. 1977. *Computer-Aided Architectural Design*. New York: Petrocelli/Charter.

Moholy-Nagy, Sibyl. 1957. *Native Genius in Anonymous Architecture*. New York: Horizon Press.

Mostow, Jack. 1985. "Toward Better Models of the Design Process". *The AI Magazine*, Spring.

Newell Alan and Simon Herbert A. 1972. *Human Problem Solving*. New Jersey: Prentice-Hall, Inc.

Roger, Gray. 1978. *BAU/2*. Technical Report, Department of Architecture, Swiss Federal Institute of Technology, Zürich.

Rudofsky, Bernard. 1988. *Architecture without Architects: A Short Introduction to Non-Pedigreed Architecture*. NM: University of New Mexico Press.

Rychener, Michael (ed). 1988. *Expert Systems for Engineering Design*. New York: Academic Press.

Simon, Herbert A. 1969.*The Sciences of the Artificial*. Cambridge: The MIT Press.

Simon, Herbert A. 1975. "Style in Design". In Eastman, Charles M. (ed), *Spatial Synthesis in Computer-Aided Building Design*. London: Applied Science Publishers.

Stiny, George. 1980. "*Introduction to Shape and Shape Grammars*". Environment and Planning B7:343-351.

Steadman, J. P. 1979. The Evolution of Designs: Biological Analogy in Architecture and the Applied Arts. Cambridge: Cambridge University Press.

Takala, Tapio. 1987. "Theoretical Framework for Computer Aided Innovative Design". In Yoshikawa, H. and Warman, E. A. (eds), *Design Theory for CAD*. Amsterdam: North-Holland.

Yoshikawa, Hiroyuki. 1987. "General Design Theory and Artificial Intelligence". In T. Bernold (ed), *Artificial Intelligence in Manufacturing*, Amsterdam: North-Holland.

---

1. Herbert Kramel is a professor in Dept. of Architecture, ETH, who is acting as the architect (domain expert) of BAU.
2. Chen-Cheng Chen is a graduate student in Dept. of Architecture, ETH, who is acting as the knowledge engineer of BAU.
3. From now on, we will use "unit" to indicate the "basic architectural unit".
4. The term is used here as employed by Stanislaus Von Moss in 1964.
5. Problem solving is a process in which one starts from an initial state and proceeds to search through a problem space in order to identify the sequence of operations or actions that will lead to a desired goal.
6. A production system is a database of production rules and some control mechanism that selects applicable production rules to reach some goal states.
7. All the possible states that could occur as a result of intersections between the elements and operators within a problem-solving process.