COMPUTER METHODS IN ARCHITECTURAL PROBLEM SOLVING: CRITIQUE AND PROPOSALS

Per Galle

While the development of modeling and drafting tools for computer-aided design has reached a state of considerable maturity, computerized decision support in architectural sketch design is still in its infancy after more than 20 years. The paper analyzes the difficulties of developing computer tools for architectural problem solving in the early stages of design where decisions of major importance are made. The potentials of computer methods are discussed in relation to design as a static system of information, and to design as a dynamic creative process. Two key problems are identified, and on this background current computer methods intended for use in architectural sketch design are critically reviewed. As a result some guidelines are suggested for future research into computer-aided architectural problem solving. The purpose of the paper is twofold: (1) to encourage research that will take this field into a state of maturity and acceptance by practitioners, and (2) to provoke further debate on the question of how to do it.

This article has been reprinted with permission of the publisher from the Journal of Architectural and Planning Research, Volume 6, Number 1, Spring 1989, pp. 34-54.

INTRODUCTION

Computer-aided design in architecture can be viewed as a field comprising two rather disparate main areas: computer-aided problem-solving (CAPS) and computer-aided modeling and presentation (CAMP), i.e. representation and drafting of designs. Architectural problem-solving is thought of as the search for and selection of basic solution alternatives that takes place in sketch design; typicafly schematic space planning at the site or floor plan scale. While the development of practically useful modeling and drafting tools seems to proceed smoothly [Baer et al (1979); Christiansson (1984)], this cannot be said about CAPS which, after more than 20 years of experiments, still seems to appeal more to researchers than to practitioners. This awkward situation prevents us from utilizing the expected benefits of CAPS, vis. better analysis of design problems and their potential solutions; in short more well-founded design decisions.

The tasks to be solved by CAMP systems are well-defined and inter-disciplinary, as opposed to the tasks of CAPS systems. No doubt the relatively slow development of CAPS in architecture can be explained partially by the ill defined nature of problems that characterize architectural sketch design and a certain over-simplification of the problems to make them amenable to computer methods. Moreover, once the problems are submitted to computer processing their complexity reveals itself. Space planning, for instance, is sometimes formalized as a "quadratic assignment problem" which is known in computer science as a so-called NP-complete problem [Liggett (1980)]. This implies that on a sequential computer any known method for its exact solution requires time or storage that grows exponentially with the problem size. Other formulations also tend to require explosively growing computer resources.

But the lack of general acceptance of the results obtained so far is hardly due to these difficulties alone. Numerous CAPS systems for design of furniture layouts, floor plans, site plans and for general space planning tasks have been presented in the literature, tacitly assuming that these new "powerful tools" would be welcomed by practitioners. [For comprehensive surveys of the literature, and illustrations of various CAPS methods, see the excellent introductory texts of Mitchell (1977) and Steadman (1983)]. There has been, however, a tendency to neglect the crucial question: *What do we want these systems to do for us?* Proper discussion of this is obviously a necessary, though not a sufficient, condition for CAPS to become more generally accepted and practically useful.

Cross (1977) warned against the dangers of uncritical development of CAD in architecture and presented a general check-list of questions to be considered prior to the implementation of comprehensive systems. In a similar vein this paper will take up the above question, but in the narrower context of problem-solving in sketch design, i.e. the most creative phase of architectural design where major decisions are made. This is where the need for a debate seems most urgent.

The two sections to follow will consider two different aspects of architectural design (especially CAPS): The first section describes two main types of *information* involved in design: criteria and solutions, while the second section focuses on the process of design and the division of labor between man and machine.

The third section will criticize current problem-solving strategies in CAPS in terms of these ideas of design, and finally the fourth section concludes the discussion by suggesting some potentially fruitful directions for research into CAPS.

Throughout the paper examples are drawn from floor plan design, a fundamental architectural activity, but the discussion aims at architectural sketch design in general, viewed as problem-solving of a combinatorial nature. The recommendations made are intended not only as an aid in the development of specific CAPS systems with due respect to architecture as an art, but also as a subject of further debate.

DESIGN AS INFORMATION: CRITERIA AND SOLUTION SETS

Architectural design can be understood roughly as: a process which (1) creates one or more design solutions, i.e. geometrical configurations modeling some desired state of the physical world as specified by a number of design criteria (given a priori, or themselves created by the process), and (2) selects one such solution to be the best in terms of the (final) criteria.

Forgetting for a moment about design as a *process*, we first concentrate on the *information* aspect of design: What sorts of criteria and solution sets are involved, what are their mutual relationships, and what does it mean for the development of CAPS?

A Taxonomy of Design Criteria

Any problem of architectural design is determined by the architect's and the client's *architectural* criteria that characterize the solutions to be explored. Neither floor plans nor more complex architectural objects can be generated like the solution set to n linear equations with *n* unknowns, by methods leading directly to the desired result. To the author no systematic methods are known which generate solutions to realistic architectural design problems directly, without trial and error (though experienced designers may seem to "jump" directly to solutions, using their intuition); as far as systematic design goes, solutions must be *searched for* in a suitable class of geometrical configurations which are made accessible by the search method.

In traditional intuitive design architectural criteria are often implicitly agreed upon by client and architect, while the set of candidate solutions is only limited by the capacity of imagination. CAPS-methods, on the other hand, force the users to state several architectural design criteria explicitly, while typically limiting the set of candidate solutions, e.g. assuming floor plans to be rectangular mosaics of tightly packed, non-overlapping rectangles, each representing one room of the building.

In general, a computer system for design (as well as informal procedures residing in the head of a human designer) inevitably affect the "style" of the designs they produce, as argued by Simon (1975). [Cf. also Cross (1977, p 147).] That is, some design criteria are implicitly added by the system, without being derived from the user's problem defining criteria. Simon calls them "autonomous constraints". For the present purpose we prefer the term *system imposed* constraints.

TABLE 1. Classification of design criteria in CAPS.

Classes of design criteria in CAPS:		Examples from floor p1an design:
Desiderata:		
	Quantifiable:	* Minimal total pedestrian traffic. * Min. perimeter length. * Good daylighting conditions. * Min. total area.
	Non-quantifiable:	* Harmony of proportions. * Intimacy. * "Form follows function". * "Form follows fantasy". * Monumentality, order. * Visual variation. * Friendly entrance.
Architectural constraints:		
	Size:	* Min. and max. area of total plan or single rooms. * Min. and max. length of plan or single rooms. * Max. length/width ratio. * Walls on given modular grid. * Min. length of common wall between two rooms.
	Position:	* Room A adjacent to either room B or room C. * Min. 6 metres between rooms A and B. * Northern walls of rooms A and B aligned.
	Style:	* Solution is a terminal shape of a given shape grammar.
Resource imposed constraints:		* Max. CPU time and storage. * Max. number of solutions to be generated for the human designer to consider.
System imposed constraints:		* One sample plan per class of "equivalent" plans. * Equivalent plans share the same "rectangular dissection". * Equivalent plans share same "wall representation". * The solutions depend on the order of room placement. * Rectangular rooms only. * Rectangular boundary.

The *resources* available constitute a third source of design criteria. In practice there are limits to the amount of computer time and storage that can be afforded, and human resources are also limited: The time an architect can spend on a project, and the amount of information he can cope with, e.g. the number of (computer-generated) solutions he is able or willing to compare.

Apart from the trichotomy of architectural, resource imposed, and system imposed, design criteria can also be divided into *desiderata* and *constraints.* Desiderata are satisfied to some *degree,* which is not necessarily quantifiable. (E.g. walking distances can be more or less short, or bedrooms more or less private.) A constraint, in our terminology, is either satisfied or not, and it can be objectively decided which is the case (e.g. whether two rooms are adjacent as required).

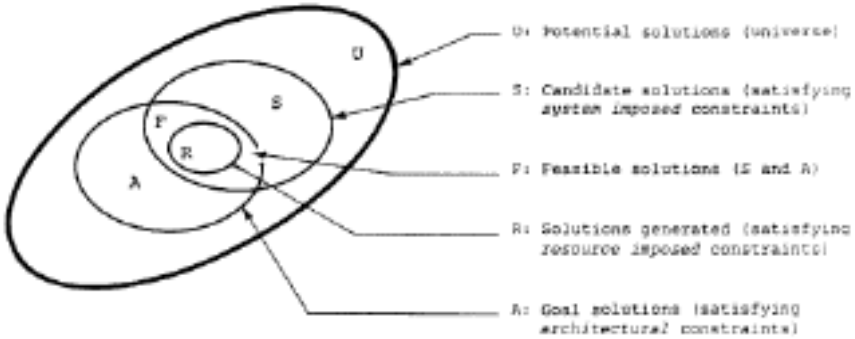

Figure 1. A Venn diagram showing the relations of goal solutions as defined by the architectural constraints given in the input to a CAPS system, candidate solutions satisfying the system imposed constraints (and thus available through the system), and solutions actually generated as part of the output. The class of generated solutions is delimited by constraints imposed by limitations of human and computer resources. (Notice that no specific sets cor respond to the desiderata listed in Table l.)

For the sake of completeness we also subdivide the architectural constraints into three types: *Size* and *position constraints,* and direct constraints on *style,* e.g. specified by a shape grammar [Stiny (1980); Stiny and March (1985)]. The ability of CAPS methods to deal with architectural style constraints was pointed out by Mitchell (1979) as highly desirable. Only recently have simple shape grammars found their way into actual CAPS systems; see e.g. Gero and Coyne (1985), Krishnamurti and Girauld (1986).

Combining the distinctions just outlined, we obtain the classification system in Table 1. The "rectangular dissections" of Mitchell et al (1976) and the "wall representations" of Flemming (1977, 1978, 1979) mentioned in the table are mathematical abstractions of solutions, both representing the structure of adjacency between rooms, but at slightly different levels of detail. Thus a set of generated solutions comprising a representative from each equivalence class of plans with the same dissection (or wall representation) will show all ways in which the adjacency constraints can be satisfied, but ignore solution variants that differ only with respect to, say, the size of rooms. [Refer to Steadman (1983) for further explanation.]

Subdividing the Universe of Potential Solutions

F'igure 1 shows the universe of *potential solutions,* i.e. all geometrical configurations that may be thought of as solutions to any problem of architectural sketch design whatsoever, and its subdivision into sets of solutions satisfying system imposed, resource imposed, and architectural constraints. [The model presented here is closely related to the models discussed by Mitchell (1975, 1979).]

The candidate solutions are the potential solutions which (theoretically) are available by means of a specific design method (computerized or manual). The class of candidate solutions of a method is delimited by the system imposed constraints of the method.

The goal solutions are those potential solutions which satisfy all architectural constraints of a specific design problem. Some of these may not apply to the candidate solutions of the method chosen, and will then usually remain implicit and without significance to the design process. E.g. the designer of a floor plan may not have any objections to circular rooms; however, by convention (or because it is supported by his computer!) he has chosen a design method which allows only rectangular rooms. Then of course he does not think of constraints on the minimum and maximum radius of rooms. But those architectural constraints which do not apply to the candidate solutions of a computerized design method obviously must form the core substance of its *input.* (E.g. the sample of architectural constraints on area and adjacency shown in Table 1 would typically occur in the input of a program for generating rectangular floor plans with rectangular rooms.)

The feasible solutions we define as the intersection of candidate and goal solutions. The more these classes intersect, the more suitable is the design method for the problem at hand. The ideal but unlikely situation of total suitability would occur if all goal solutions were candidate solutions.

The solutions generated constitute the core substance of the *output* of the system and are to be selected from the feasible solutions subject to the resource imposed constraints.

Over- Well- and Underconstraining

For a given design method (and hence a given set of candidate solutions available), the choice of architectural constraints requires a good deal of experience (or luck) if a suitable number of feasible solutions are to be delimited. Very often there will be none at all or too many with respect to the human resources:

Given the design method, a design problem is said to be *overconstrained* if the set of feasible solutions is empty. [1] If, on the other hand, there are too many feasible solutions to be generated and considered within the limits of the resources available, we call the problem *underconstrained* (with respect to resources and method). Finally, if the problem happens to be neither over- nor underconstrained we shall call it *wellconstrained.* (Cf. Fig. 2.)

Potential over- and underconstraining is probably inherent in all sorts of architectural design, usually blurred by the implicit or even subconscious use of criteria in traditional intuitive methods. But even a design problem explicitly stated for a CAPS system may be impossible to classify beforehand as either an over-, well-, or underconstrained problem. Therefore developers of CAPS systems should take great care to ensure that the systems will be able to cope with a problem in either of the three cases. As we shall argue in the next section existing CAPS (space planning) methods are unable to respond adequately to many under- and overconstrained design problems within their intended scope of application. This problem was recognized by Eastman (1975 b, p 10) more than a decade ago, so it is high time for us to pay attention to it.

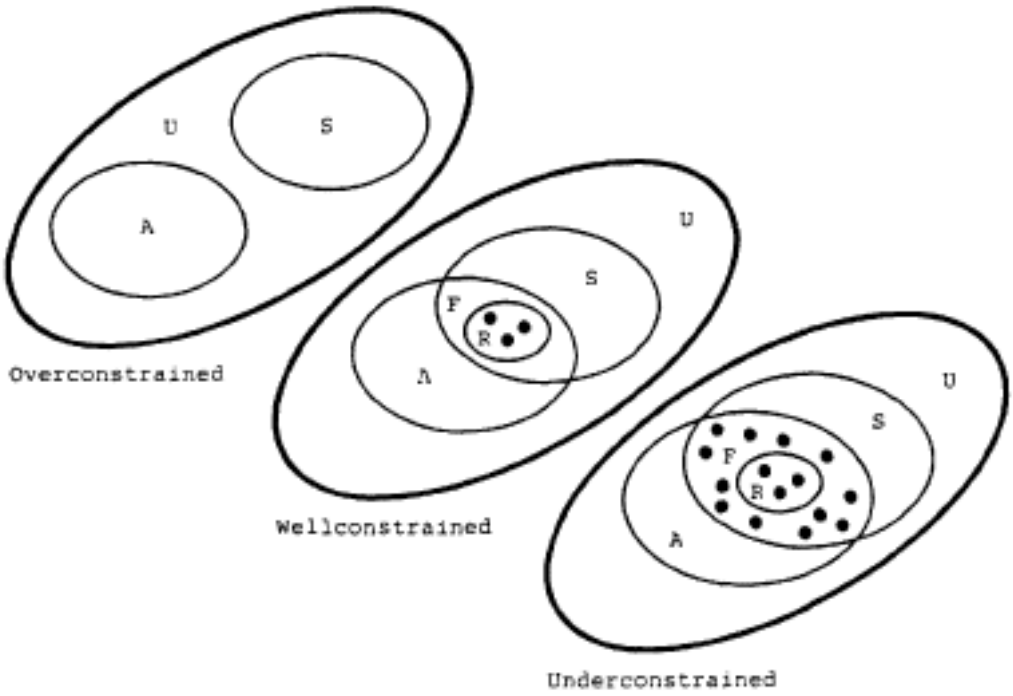

Figure 2. The three situations of over-, well-, and underconstraining illustrated in terms of the solution sets from figure 1. The dots symbolize feasible solutions.

Much of the difficulty of design arises from situations of underconstraining where the number of choices open to the designer appears practically infinite, rendering a decision very hard to make - and from situations of (unrecognized) overconstraining where the problem just looks increasingly intractable to him the more he works on it. If CAPS systems fail to be helpful in such moments of frustration, the designer is likely to discard them as useless.

DESIGN AS PROCESS: DIVISION OF LABOR BETWEEN MAN AND MACHINE

Another problem of crucial importance in further research into CAPS is the basic question of what parts of design as a *process* should be automated and what should be left to the human designer. [A point also stressed by Cross (1977).] In order to discuss this, we should elaborate a little on our crude definition of architectural design as stated above. Maver (1977) presents a model of architectural design, originating from The Building Performance Research Unit. According to this model, design proceeds through various stages with increasing levels of detail, each composed of four steps: analysis, synthesis, appraisal, and decision (with feed-back loops connecting them). Here we are interested in just a single early stage of the design process, vis. sketch design, but the four steps of such a design stage are useful for a discussion of the man / machine relationship. Table 2 contains in its left column a brief description of four design steps much like Maver's, but in terms of the concepts introduced in the previous section.

TABLE 2. Limits to computerization of architectural design, as proposed by the author.

	Division of labour between men and machine:	
Design Step	Designer does at least:	Machine does at most:
1. ANALYSIS: Analyze purpose of design. Choose accordingly the criteria (constraints, and desiderata) and design method.	Choose design criteria, e.g. according to the client's wishes. Choose method of design.	Reveal inconsistency in constraints, and possibly suggest how to relax them.
2. SYNTHESIS: Generate feasible solutions; if none exists, return to step 1.	Nothing.	Everything.
3. APPRAISAL Compare generated feasible solutions with respect to fulfillment of desiderata.	For each feasible solution generated: Evaluate fulfillment degree of non-quantifiable desiderata; determine relative importance of all desiderata.	For each feasible solution generated: Measure fulfillment degree of quantifiable desiderata.
4. SELECTION: Select best generated solution and stop; or return to step 1,2, or 3.	Everything.	Nothing.

(We see that some decision-making takes place in all steps except no. 2; hence the fourth step has been called SELECTION, rather than DECISION. It selects either a final solution or the next step to be repeated in the process.)

Limits to Computerization

The middle and right columns of Table 2, respectively, show what the author believes should at least be left to the human designer, and should *at most* be done by the computer. The Underlying (ethical and practical) principle is the following:

All activities involving value judgement should be carried out by the human designer, while purely logical matters (satisfaction and consistency of constraints) and purely numerical matters (measurement of the degree of satisfaction of quantifiable desiderata) may be left to the machine.

In each step requiring decisions, these are left entirely to the designer who thus retains his responsibility, while the task of the computer is that of providing the designer with adequate information that enables him to make his decisions. Clearly, each of these are also based on value judgements: In STEP 1 the formulation of criteria (to the extent that they are not given a priori), so as to make clear what a good, or at least acceptable, solution is like. In STEP 3 the subjective weighting of criteria. Finally, in STEP 4 the decision whether the generated solutions are good enough or search for better solutions should continue, also involves a value judgement. Typically this judgment will rely on a trade-off between the improvement expected from continued design efforts, and the resources needed (time, money, etc.).

How Close to the Limits?

Table 2 does not tell us how close to the Emits of computerization we should go. If we want the insight offered by a systematic exploration of the set of feasible solutions, as assumed in this paper, we should go as close as we can to these limits.

In cases of trivial routine design, automating STEP 2 could hardly jeopardize any substantial values of human creativity. As for the majority of non-routine cases it would seem unlikely that all goal solutions will be included in the set of feasible solutions available by computer. The search for solutions performed by the computer should then be supplemented by intuitive search in the set of goal solutions that are not feasible in terms of computer methods. The apparently controversial "Nothing" of Table 2 obviously applies only within the limited scope of CAPS methods. The solutions automatically generated should be taken as an inspiration for the architect to continue search for solutions beyond reach of the computer.

OPTIMIZE, SIMULATE, OR SATISFY?

Before we embark upon the future of CAPS, let us briefly review its past and present in the light of over- well- and underconstraining, and proper division of labor. Following Radford and Gero (1980), we divide CAPS methods into three main families:

* *(1) Optimization:* Generation of one or a few solutions which are not only feasible but also "best", e.g. in the sense of minimizing some measure of cost.

* *(2)Simulation:* Prediction of the performance of a given potential solution with respect to one or more criteria. Simulation methods do not themselves produce the solutions to be investigated; usually simulation is combined with a graphic user-interface allowing the designer to propose potential solutions interactively.

* *(3)Satisfaction:* [2] Generation of some or all feasible solutions, i.e. solutions that satisfy the given constraints, but without any automated ranking or evaluation of the solutions.

Optimization

The usefulness of optimization methods in purely technical matters of operational research, construction engineering, and environmental or building design will not be questioned here. However, a large group [Gero (1973, 1983)] of computer methods based on optimization have been devised for use in genuinely architectural sketch design, vis. systems for automated floor plan layout. They automatically synthesize and select one or a few "best" solutions using various techniques to minimize an *objective function* of the layout. This function expresses the fulfillment degree of one or more *quantifiable* desiderata (often called *objectives).* Typically the objective function to be minimized is the sum of the traffic line lengths, multiplied by some measure of expected traffic intensity, e.g. the number of people walking along those lines. [The early system of Whitehead and Eldars (1964) even weighted the cost of walks according to the salaries of people walking.]

The problem of overconstraining is ignored by these methods, rather than solved. By simplifying architectural reality to essentially a matter of minimizing a function of distances and traffic intensity the relevant architectural constraints that might cause overconstraining are merely neglected. It should be noted, however, that Liggett and Mitchell (1981b) adopt a highly interactive approach to floor plan optimization in order to take a richer set of design criteria into account, including non-quantifiable desiderata. Their method incorporates, however, a substantial element of simulation.

Theoretically a design problem may have so many feasible solutions which are all optimal in the sense of the objective function (or are within an acceptable deviation from the optimum) that the problem would be underconstrained if they were all to be generated. Under the disputable assumption that the objective function expresses all criteria that matter, one solution optimal in the sense of the function is (in principle) as good as any other. Hence the underconstraining can easily be evaded by picking an optimal solution at random. From an architectural point of view, however, the price of such "elimination" of underconstraining is high, since in spite of more recent improvements, notably by Liggett and Mitchell (1981 a), *optimization methods are inherently unable to deal with non -quantifiable desiderata.*[3]

Admittedly, judicious use of optimization may allow the designer to have several solutions generated which are optimal or near-optimal or, as suggested by Radford and Gero (1980) and by Gero and Balachandran (1986), Pareto-optimal with respect to multiple objectives. (A solution S is said to be Pareto-optimal with respect to a set O of objectives if no other solution is better than S with respect to every objective in.) The designer can then compare the generated solutions in terms of qualitative and other criteria not taken into account by the optimization technique. But even such a procedure is biased toward the quantifiable desiderata. The solution which is "optimal" with respect to *all* desiderata (given their relative importance which is a subjective decision) is not necessarily one of the solutions that are optimal or near-optimal in quantifiable terms.

Although this is clearly unsatisfactory, (pure) optimization methods are primarily objectionable in architectural sketch design because they attempt a full automation of appraisal (STEP 3 in Table 2), tacitly assuming that the relative importance of non-quantifiable desiderata is negligible, thus indirectly making judgments on behalf of the designer. By generating only quantitatively optimal solutions, they also indirectly make decisions on behalf of the designer (STEP 4) as regards selection of the best solution, thus violating the limits to computerization outlined above [Cf. also Galle (1981) and Steadman (1983, pp 140-141).]

Simulation

The optimization techniques originate in the field of operational research; hence it is worth noting that the problematic application of them to architectural space planning tasks has long ago been criticized from within that field vis. by Krarup and Pruzan (1978), who recommend instead a simulation approach.

Simulation programs for architectural sketch design have been developed mainly at the University of Strathclyde in Scotland [Maver (1977, 1979)]. Today some of them have found their way into practice in Holland [Kraal (1983)], as well as architectural education in a diversity of countries such as Holland, Denmark [Agger (1982)], and Israel [Roth (1985)].

The basic task of such programs is to point out to the designer if his solution proposals do not satisfy the architectural constraints (STEP, 1), and to measure the fulfillment degree of quantifiable desiderata (STEP 3). According to Maver (1979) this enables the designer to investigate up to ten times as many potential solutions as he could do without the aid of the programs.

Thus augmenting the resources at the designer's disposal the programs render many problems wellconstrained that would otherwise be underconstrained. But since all tasks of synthesis are left to the human designer, there is no guarantee that any (let alone all) relevant feasible solutions are actually considered, even if the problem is wellconstrained. And there is no way in which the designer can know if this is the case.

As far as overconstraining gun the simulation approach has the advantage of allowing the designer to hypothesize unfeasible solutions whose violations of the constraints are then immediately discovered. Thus he may gain a useful insight into the causes of the overconstraining.

In general, the basically passive and un-creative role played by a simulation system in the design process ensures that the limits to computerization from Table 2 are certainly not violated; on the contrary it is a weakness of (pure) simulation that in the synthesis step it stays too far behind those limits. No systematic investigation of alternative solutions is achieved. On the other hand the simulation methods are able to utilize the potential benefits of computerizing the analysis and appraisal steps.

Satisfaction

The same objection regarding unsystematic search can be made against the class of non-exhaustive constraint satisfaction methods, i.e. methods that automatically generate one or some solutions, governed by ad-hoc heuristics or random number generators. Such methods have been developed by Weinzapfel and Handel (1975) (whose "IMAGE" system can also run in a simulation mode), Velez-Jahn (1971, 1973), Teicholz (1975), Ruch (1978) and have also been advocated by Willey (1978).

But several computer-based architectural satisfaction methods are (basically) exhaustive. That is, they generate systematically (within resource limits) all feasible solutions (e.g. schematic floor plans; cf. Figures 3 and 4). A pioneering effort was made by Grason (1968, 1970). Systems later developed by Mitchell et al (1976), Flemming (1977, 1978, 1979), and Roth et al (1982, 1985) impost various constraints in order to reduce the risk of underconstraining, vis. by selecting sample solutions representing certain classes of solutions which are considered "equivalent" (cf. the remarks explaining Table 1). These more recent systems are hybrids in the sense that exhaustive satisfaction is combined with optimization which is, however, used only as an auxiliary tool for selecting sample solutions. A new system based on such a principle is also suggested by Flemming (1986). The floor plan generator "FLOP 1" by Galle (1981, 1983) performs a purely exhaustive search under the additional system imposed constraint that the modular grids of the plans be as coarse as possible.

Although the system imposed constraints embedded in such sampling techniques dramatically reduce the number of problems that would be underconstrained, exhaustive satisfaction methods tend to be very sensitive to "combinatorial explosions": They may encounter a

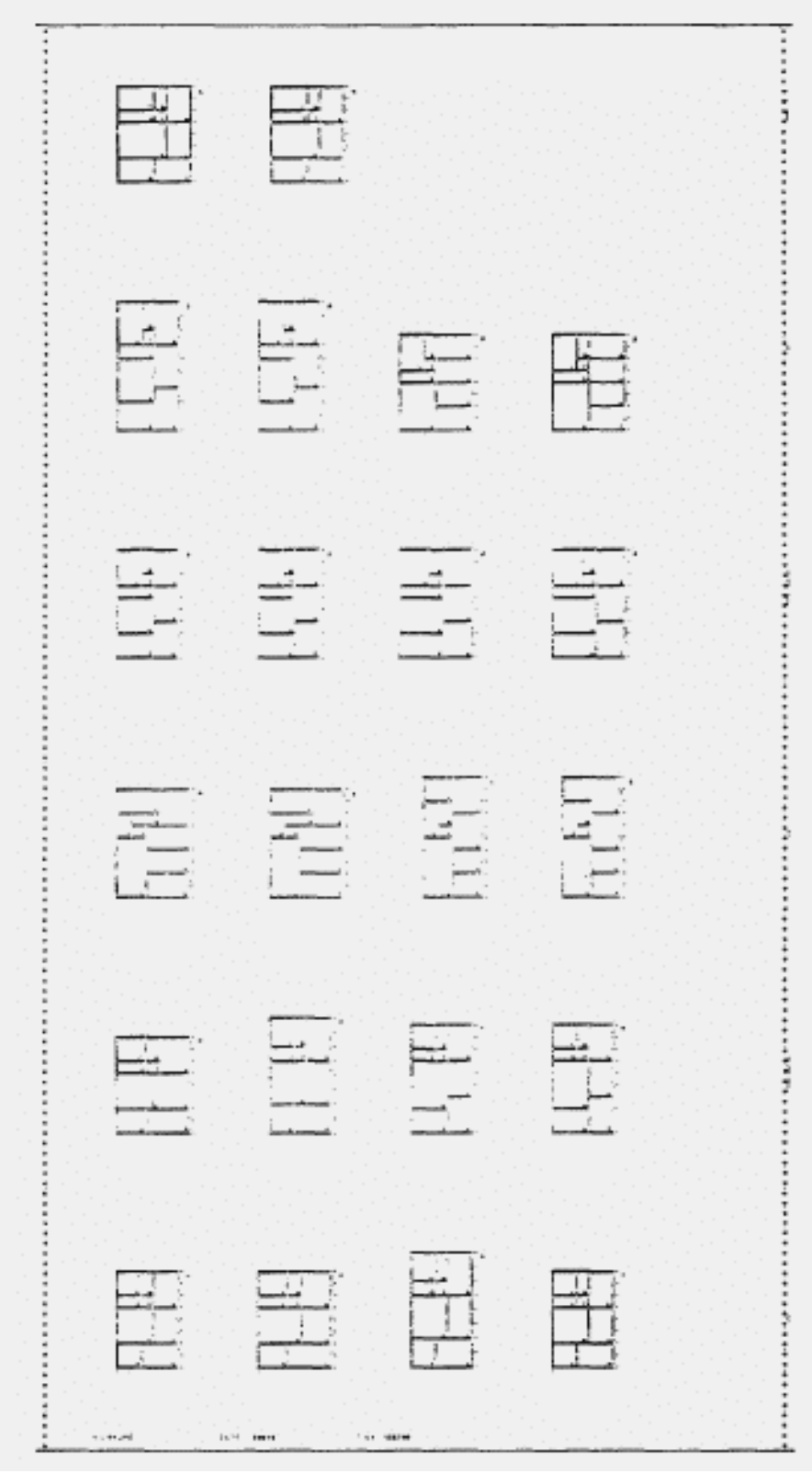

Figure 3. Sample output from the author's FLOP 1 system (Galle (198l): Exhaustive list of all 22 solutions to a simple house design problem. The system was restricted to rectangular floor plans.

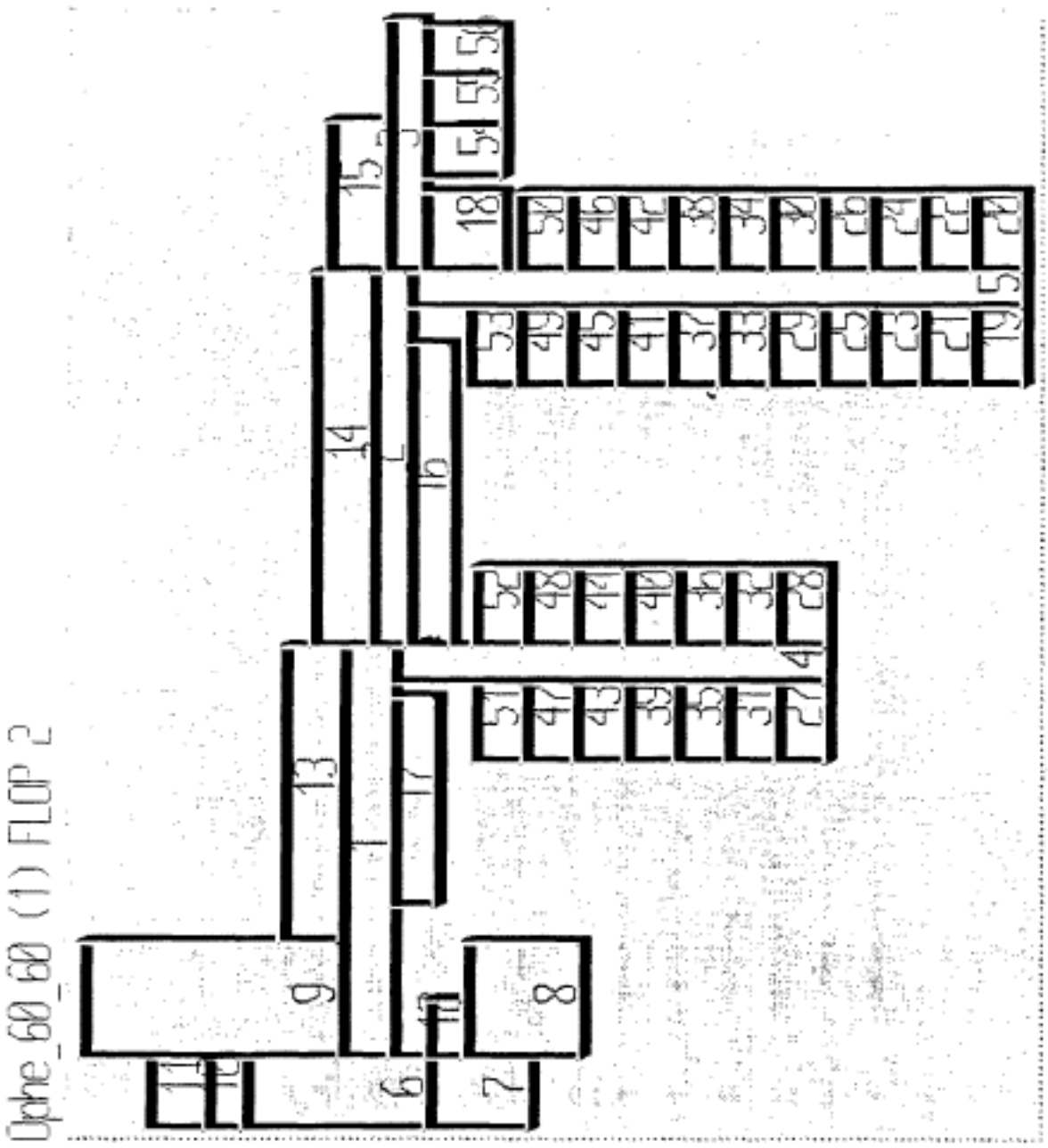

Figure 4. Design for an old people's home, generated by the FLOP 2 system [Galle (1987 a, b)] in a test run creating new variants of an existing building by the Danish architects Buhl & Klithoj.

problem which (in spite of the carefully devised system imposed constraints) is underconstrained, and so forces the generation process to stop before a systematic overview of the entire set of feasible solutions has been obtained. (E.g. a sample solution may not have been found in all equivalence classes.)

The obvious approach to this effect of underconstraining has been taken by Flemming (1978) and by Roth et al [Roth (1984)] who allow the human designer to guide the generation process interactively. Indeed, resorting to interaction may be necessary but is likely to re-introduce the element of randomness that was characteristic of the simulation methods.

TABLE 3. A subjective evaluation of current methods of CAPS in architectural sketch design. with respect to overwell- and underconstraining. (Applications of the methods as auxiliary techniques in "hybrid" methods are not taken into account.)

	Evaluation of method depending on type of design problem:		
CAPS Method:	Overconstrained:	Wellconstrained:	Underconstrained:
Optimization	Weak: Overconstraining eliminated by oversimplifying design problem	Weak: Solutions explored systematically (in terms of objective function) but nonquantifiable desiderata suppressed	Weak: Underconstraining largely eliminated but nonquantifiable desiderata suppressed
Simulation	Strong: Diagnostic power	Weak: No systematic exploration: Many or all feasible solutions may be missed	
	Weak: Simple input control tests;		
Non-exhaustive satisfaction	Strong: Diagnostic power as in "IMAGE"		Weak: No systematic exploration of feasible solutions
Exhaustive satisfaction	Weak: At most simple input control tests	Strong: Systematic exploration of feasible solutions (provided reasonable systems imposed constraints)	

As for overconstraining, some of the methods incorporate input control tests that may reveal simple cases of inconsistency in the constraints specified by the designer, notably constraints on room adjacency that are impossible to satisfy in two-dimension layouts; cf. e.g. Ruch (1978). However a complete set of size and position constraints may be impossible to satisfy simultaneously, without their inconsistency being *a priori* detectable by any simple test. The literature has nothing much to report on the behavior of the systems in such cases, presumably because they are, e.g. like FLOP 1, unable to give any helpful information at all when the design problem is overconstrained in this complex way.

One remarkable exception is the non-exhaustive IMAGE system mentioned above. It manipulates box-shaped volumes representing e.g. furniture, rooms, or buildings, trying to satisfy a number of constraints e.g. on area, distance, alignment, non- overlapping, and visual access. It currently checks for each box whether it satisfies all constraints pertaining to it. If not, the system calculates a suitable displacement of the box so as to reduce the constraint violations of that particular box. As pointed out by Eastman (1975b), the basic assumption of IMAGE is that the design problems are usually overconstrained. Therefore the system effects a *compromise* so as to minimize the overall violations of constraints (and hence may fail to find

feasible solutions even if they exist). An important feature of the compromise generator is that it is able to report the constraints that have been (most) violated, thus giving the designer potentially valuable information.

Generally speaking, however, the weakness of current satisfaction methods is their inadequate response to over- and underconstrained problems. The strength of exhaustive satisfaction methods is that for wellconstrained problems they give us exactly the systematic overview desired for decision-making, going close to but not across the limits to computerization.

Comparison

While, as we have argued, the optimization family of methods fail to solve the general division-of-labor problem properly in the architectural context, simulation and (exhaustive) satisfaction seem to complement each other nicely in this respect: Simulation is able to utilize the automatization potentials of analysis and appraisal (cf. Table 2) but not of synthesis -- and vice versa for exhaustive satisfaction.

Turning to the problem of over- well- and under constraining, we notice a similar complementarity of strengths and weaknesses between simulation (together with the IMAGE system) on one hand and exhaustive satisfaction on the other hand. The pattern becomes clear from Table 3 which summarizes the above discussion of this problem. An obvious conclusion to draw is that the best choice of CAPS tools for typical architectural sketch design, given the broadly published techniques of *today,* is a combination of simulation (or an IMAGE-like approach) and exhaustive satisfaction. But as the table shows, we still need tools for proper handling of under constrained design problems.

It should be admitted at once that the statements contained in the table, especially when condensed and simplified in this way, may seem rather dogmatic. Since the premises of the above argument may not be generally agreed upon (e.g. the limits to computerization, the desirability of systematic exploration of solutions, or the importance of non-quantifiable desiderata), one cannot expect the conclusions to be so.

CONCLUSIONS AND RECOMMENDATIONS

We have identified two key problems in the development of CAPS methods for architectural sketch design:

* (1) Proper handling of over- well- and underconstraining: It was pointed out that CAPS methods should be able to respond to all three situations; particularly methods which fail to help the designer avoid the frustrations of under- or overconstraining are unlikely to become generally accepted.

* (2) Proper division of labor between man and machine: An attempt to outline "the limits to computerization" of design was summarized in Table 2, and we argued that for the potential benefits of CAPS to be better utilized the limits should be approached -- but not violated.

Since, according to the discussion summarized in Table 3, none of the methods currently available i satisfactory with respect to both of the key problems, these issues should obviously be addressed in future research.

More specifically we reached the conclusion that simulation and exhaustive satisfaction wer preferable to (pure) optimization [4] with respect to (2) and probably could be combined in order to solv it. Recommending therefore that optimization be generally abandoned (or reserved for auxiliar purposes), how could we suggest improvements of the remaining methods with respect to (1)? - A reasonable approach would be to concentrate on those of the weaknesses listed in Table 3 which seen most likely to be remediable.

Overconstrained design problems, for example, imply a need for improved diagnostic power o satisfaction methods. Although satisfying a set of inconsistent constraints is impossible by definition certain approaches to overconstraining have already emerged in the context of satisfaction. One is systematic relaxation of constraints so as to eliminate the inconsistency, followed by generation of sample of the corresponding, not strictly feasible, solutions. If the designer knows which constraints ar relaxed, and how, those solutions may give him enough insight to decide upon a revised set o constraints. The non-exhaustive IMAGE system actually performed constraint relaxation along thes lines. A principle based on (potentially exhaustive) generation of *abstract, approximate solutions* wa suggested by Galle (1986), and tested in the "FLOP 2" system which has only very recently bee implemented. A preliminary report on the results is given in [Galle (1987a, b)]. Another techniqu introduced in FLOP 2 for handling overconstraining is generation of *maximal incomplete solutions;* e.g floor plans with as many of the required rooms as possible, but with certain rooms omitted whos constraints cannot be satisfied. But more ambitious approaches are needed that would enable the system to suggest explicitly how to modify overconstrained problems.

Wellconstrained design problems should obviously be solved by exhaustive satisfaction wheneve possible, if one accepts the conclusions in Table 3. The weakness of simulation and non-exhaustiv satisfaction in the case of wellconstraining stems from the very principles of these methods. Henc attempts to improve them would be likely to lead towards either optimization or exhaustive satisfactio As for the latter, an important area of research is the continued effort to render as many problems a possible wellconstrained, e.g. by developing efficient combinatorial search algorithms [Mackwort (1977); Kornfeld (1982)] and inventing useful sampling techniques in terms of system impose constraints [Simon (1975, p 300); Flemming (1979; 1986); Galle (1986; 1987a)].

Underconstrained design problems are evidently the most difficult ones to master. Again, improvin the weaknesses of simulation and non-exhaustive (heuristic or random number based) satisfaction seem impossible *a priori* because the search is unsystematic by definition. But at least in principle nothin prevents exhaustive satisfaction (i.e. exhaustive within resource limits) from improvement with respect t underconstraining. No doubt the poor behavior of most exhaustive methods in cases o underconstraining results from their underlying unwarranted optimism: They rely on the hope that th problem at hand (perhaps due to system imposed constraints) will turn out to be wellconstrained, so tha all feasible solutions will eventually be generated. Encouraging results have been obtained by designin FLOP 2 on the more pessimistic assumption that each generated solution may turn out to be the last on permitted by scanty resources: A simple combinatorial search strategy called *Branch & Sample* [Gall (1987c)] was developed which ensures at any stage of the search that the solutions generated so

far are representative of the entire set of feasible solutions, and are uniformly scattered over it. (Technically, representativeness and scatteredness are defined in terms of a so-called ultrametric distance function over the search tree).

These years the idea of *"knowledge based systems" is* being intensely investigated by some CAPS researchers, and one can hope for a break-through in that direction leading to more 1nteWgent" and user-friendly systems. For example Coyne and Gero's (1985a, b) attempts to transfer so-called "planning" techniques from other areas of artificial intelligence to floor plan design, and their suggestions for further research, show some promise. But even so the fundamental and serious problems of CAPS which have been the subject of this paper are likely, for quite a long time, to remain fundamental and serious.

NOTES

[1] Thus to say that a design problem is overconstrained strictly speaking amounts to stating the non-existence of an object with certain properties. (A feasible solution.) But we know from Goedel and others that no general algorithm exists to prove or disprove such a statement in finite time. Therefore in practice we shall consider a problem overconstrained whenever the resources and the rnethod at hand do not allow us to find any feasible solutions to it.

[2] Radford and Gero (1980) use the term generation

[3] Gero and Oguntade (1978), as well as Oguntade and Gero (1981), have proposed Zadeh's fuzzy set theory as a formalisrn which might enable CAPS methods to deal properly with subjective evaluation, but to the author's knowledge no systems with such an ability have yet been implemented.

[4] For an argument concluding in favor of optimization, see the paper by Radford and Gero (1980).

REFERENCES

Agger K (1983) Aktiviteter i EF-regie. EF-sambarbejde om edb-uddannelse of arkitektstuderende. In Sigrist F (ed.), *EDB i byggesektoren.* SBI-meddelelse 27. Report (in Danish) from SBI (the Danish State Building Research Institute). Horsholm, Denmark: SBI, 31-36.

Baer A, Eastman C, Henrion M (1979) Geometric modeling: a survey. *Computer Aided Design; 11: 253.*

Christiansson P (1984) Datoerstodd Projektering, CAD; Besoek vid Foeretag och Universitet; USA Hoesten 1983. Report (in Swedish) TVBK-3019. Lund, Sweden: Tekniska Hoegskolan i Lund (Technical University of Lund).

Coyne RD, Gero JS (1985a) Design knowledge and sequential plans. *Planning and Design: Environment and Planning B; 12: 401.*

Coyne RD, Gero JS (1985b) Design knowledge and context. *Planning and Design: Environment and Planning B; 12:* 419.

Cross N (1977) *The automated architect.* London: Pion Ltd.

Eastman CM (1975 a) (ed.) *Spatial Synthesis in Computer-Aided Building Design.* London: Applied Science Publishers.

Eastman CM (1975 b) The Scope of Computer-Aided Building Design. In Eastman (1975 a) 118.

Flemming U (1977) *Automatisierter Grundrissentwurf. Darstellung, Erzeugung and Dimensionierung von dicht gepackten, rechtwinkligen Flaechenanordningen.* (Diss.) Berlin: FB 8 der Technischen Universitaet Berlin.

Flemming U (1978) Wall representations of rectangular dissections and their use in automated space allocation. *Environment and Planning B; 5:* 215.

Flemming U (1979) Representing an Infinite Set of Solutions through a Finite Set of Principal Options. In Seidel AD and Danford S (eds.), *Environmental Design: Research, Theory, and Application. (EDRA 10)* Washington D.C.: Environmental Design Research Association, 190-197.

Flemming U (1986) On the representation and generation of loosely packed arrangements of rectangles. *Planning and Design: Environment and Planning B;* 13: 189.

Galle P (1981) An Algorithm for Exhaustive Generation of Building Floor Plans. *Communications of the A CM; 24:* 813.

Galle P (1983) A Theorem Relating to Exhaustive Generation of Floor Plans. *Bulletin of Computer Aided Architectural Design;* 48: 30.

Galle P (1986) Abstraction as a Tool of Automated Floor Plan Design. *Planning and Design: Environment and Planning B;* 13: *21*

Galle P (1987a) *A Formalized Concept of Sketching in Automated Floor Plan Design.* DIKU report 87/3. Copenhagen: Datalogisk Institut (Institute of Computer Science) University of Copenhagen.

Galle P (1987b) A *Basic Problem Definition Language for Automated Floor Plan Design.* DIKU report 87/4. Copenhagen: Datalogisk Institut (Institute of Computer Science) University of Copenhagen.

Galle P (1987c) *Branch & Sample: Systematic Combinatorial Search without Optimization.* DIKU report 87/5. Copenhagen: Datalogisk Institut (Institute of Computer Science) University of Copenhagen.

Gero JS (1973) *The Application of Operations Research to Architecture - A Review.* Computer Report CR *21.* Sydney: Dept. of Architectural Science, University of Sydney.

Gero JS (1983) Computer-Aided Architectural Design - Past, Present and Future. *Architectural Science Review*; 26: 2.

Gero JS, Balachandran M (1986) Knowledge and Design Decision Processes, in: Sriram D, Adey (eds.) *Applications of Artificial Intelligence in Engineering Problems.* Berlin: Springe *343-352.*

Gero JS, Coyne RD (1985) Logic programming as a means of representing semantics in desig languages. *Planning and Design: Environment and Planning B* ; 12: 351.

Gero JS, Oguntade 00 (1978) *Fuzzy Set Evaluations in Architecture: Preliminary Studies.* Comput Report CR 29. Sydney: Dept. of Architectural Science, University of Sydney.

Grason J (1968) A Dual Linear Graph Representation for Space-Filling Location Problems of the Flo Plan Type. In Moore GT (ed.): *Emerging Methods in Environmental Design and Plannir* Proc. of The Design Methods Group, lst Int. Conf. Cambridge, MA: The Design Metho Group, 170-178.

Grason J (1970) Fundamental Description of a Floor Plan Design Program. In Sanoff H and Cohn (eds.): EDRA [1]. Proc. of the lst ann. Design Research Association Conf. held at Chap Hill, North Carolina. Stroudsburg, Pennsylvania: Dowden, Hutchinson & Ross, Inc 175-180.

Kornfeld WA (1982) Combinatorially Implosive Algorithms. *Communications of the ACM* ; 25: 734.

Kraal L (1983) Implementation Experiences of Design Applications. In *PARC 83* Proceedings of tl International Conference on Computers in Architecture. Oxbridge, Middelsex: Onli Conferences Ltd: 65-79.

Krarup J, Pruzan PM (1978) Computer-Aided Layout Design. *Mathematical Programming Study* ; 9: 75

Krishnamurti R, Girauld C (1986) Toward a shape editor: the implementation of a shape generatic system. *Planning and Design: Environment and Planning B* ; 13: 391.

Liggett RS (1980) The quadratic assignment problem: an analysis of applications and solution strategie *Environment and Planning B* ; 7: 141.

Liggett RS, Mitchell WJ (1981 a) Optimal space planning in practice. *Computer-Aided Design* ; 13: 277

Liggett RS, Mitchell WJ (1981 b) Interactive graphic floor plan layout method. *Computer Aided Desigr* 13: 289.

Mackworth AK (1977) Consistency in Networks of Relations. *Artificial Intelligence* ; 8: 99.

Maver TW (1977) Building Appraisal. In Gero J S (ed.) *Computer Applications in Architecture.* Londo Applied Science Publishers; 63-94.

Maver TW (1979) Models and techniques in design. *Design Methods and Theories* ; 13: 173.

Mitchell WJ (1975) The theoretical foundation of computer-aided architectural design. *Environment and Planning B* ; 2: 127.

Mitchell WJ (1977) *Computer-Aided Architectural Design.* New York: Van Nostrand Reinhold Company.

Mitchell WJ (1979) Synthesis with style. In *PARC 79 Proceedings.* (Int. Conf. on the Application of Computers in Architecture, Building Design and Urban Planning). Oxbridge, Middelsex: Online Conferences Ltd.: 119-134.

Mitchell WJ, Steadman JP, Liggett RS (1976) Synthesis and optimization of small rectangular floor plans. *Environment and Planning B* ; 3: 37.

Oguntade OO, Gero JS (1981) Evaluation of architectural design profiles using fuzzy sets. *Fuzzy Sets and Systems* ; 5: 221.

Radford AD and Gero JS (1980) On Optimization in Computer Aided Architectural Design. *Building and Environment* ; 15: 73.

Roth J (1984, 1985) Private Communication.

Roth J, Hashimshony R, Wachman A (1982) Turning a Graph into a Rectangular Floor Plan. *Building and Environment* ; 17: 163.

Roth J, Hashimshony R, Wachman A (1985) Generating Layouts With Non-Convex Envelopes. *Building and Environment* ; 20: 211.

Ruch J (1978) Interactive space layout: a graph theoretical approach. In Proc. of *Design Automation Conference No. 15.* New York: ACM/IEEE, 152-157.

Simon HA (1975) Style in Design. In Eastman (1975 a), 287-309.

Steadman JP (1983) *Architectural Morphology.* London: Pion Limited

Stiny G (1980) Introduction to shape and shape grammars. *Environment and Planning B* ; 7: 343

Stiny G, March L (1985) Spatial systems in architecture and design: some history and logic. *Environment and Planning B: Planning and Design* ; 12: 31.

Teicholz E (1975) The Computer in the Space Planning Process. In Proc. of *12th Design Automation Conference.* New York: ACM/IEEE, 331-339.

Velez-Jahn G (1971) Rectangular Meshes: Their Uses and Control in Computer- Produced Architectural Schemes. In Proc. of *ACM 1971 Annual Conference.* New York: Association for Computing Machinery, 745-755.

Velez-Jahn G (1973) Hacia un Control Geometrico de la Forma: Construccion Automatizada de Tramas Orientadas a la Creation de Espacios Arquitectonicos. Caracas: Separata del *Boletin de la Academia de Ciencias Fisicas, Matematicas y Naturales,* Ano XXXII, Tomo XXXII, Nos. 94 y 95.

Weinzapfel G, Handel S (1975) Image: Computer Assistant for Architectural Design. In Eastman (1975 a) 61-97.

Whitehead B, Eldars MZ (1964) An Approach to the Optimum Layout of Single-Storey Buildings. *The Architects' Journal;* (June) 17: 1373.

Willey DS (1978) Structure for automated architectural sketch design. *Computer-Aided Design* ; 10: 307.

For additional information, please contact the author directly at the Department of Computer Science University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen 0, Den-mark.

ACKNOWLEDGMENTS

This work was supported by a research fellowship from Datalogisk Institut (The Institute of Compute Science) at University of Copenhagen. Thanks are due to Nils Andersen from Datalogisk Institute fo reading and commenting on drafts of this paper.

AUTOBIOGRAPHICAL SKETCH

Dr. Galle was born in Copenhagen, Denmark in 1950. He graduated as an architect in 1975 at the School of Architecture, Royal Academy of Fine Arts, Copenhagen. In 1987 he received a Ph.D. in Computer Science at University of Copenhagen, where he is currently engaged in research on knowledge based systems for support of architectural sketch design.

CAAD: Education -Research and Practice
ECAADE Conference 1989
School of Architecture in Aarhus Denmark

Architectural Design and Drawing

Lee Jenkinson[1] , André G.P. Brown[2] and Frank Horton[3]

1. Research Assistant
2.& 3. Lecturers

School of Architecture and Building Engineering
University of Liverpool
Liverpool L69 3BX
U.K.

Keywords: Drawing, design process, Le Corbusier, Ronchamp, CAAD

Abstract:

This paper focuses on the function of drawing in architectural design. It does so by taking an in-depth look at the drawing material produced for the design of the chapel at Ronchamp. Within architectural design there is more than one type of drawing. The objective therefore is to determine what exactly these different types of drawing are and furthermore what their function is for the architect. For we believe that questioning, at this basic level, the function of drawing within the design process provides the basis from which it is possible to go on to question the function of computer-based drawing within the design process, and consequently it's function in CAAD.