Computer Supported Creative Design: A Pragmatic Approach

by

Robert F. Coyne, Eswaran Subrahmanian

EDRC 05-46-89

Computer Supported Creative Design: A Pragmatic Approach

Robert F. Coyne and Eswaran Subrahmanian

Engineering Design Research Center, Carnegie-Mellon University
Pittsburgh, PA 15213, U.S.A.

Abstract. Creativity is recognizable only in a given context thereby making its definition elusive. In design, creativity in a domain may depend on characteristics of the process or the product, or both. In this paper, we argue that a better understanding of creativity can occur by focusing on the conditions that allow for creative acts. With respect to design research, we believe that it would be most productive to create *computer supported design environments* that stress the complementary nature of human and computer design capabilities. These capabilities and their relationships are reviewed to identify some of the critical issues in designing and building such environments. We argue for building such integrated design environments in order to empirically test and evaluate hypotheses about design processes and creativity in the context of real design tasks. In this regard we discuss LOOS: a partially automated approach for the design of layouts which illustrates some of the features desired for a computational design support environment.

1 Introduction

The idea that computers will actively aid design processes in various disciplines (or perhaps transform those processes) has been around for at least a couple of decades. Unfortunately, up to now reality has fallen far short of this promise. Each successive revolution in computing power and new paradigm for symbolic processing has been heralded by some as the needed potential that would allow Computer Aided Design [CAD] to fulfill its promise. Along these lines at least three successive waves of development of CAD can be crudely identified[1].

In the first wave, CAD tools were limited to the role of mechanizing the production and duplication of drawings and to some basic analysis programs. In the second wave, more sophisticated modeling and analysis tools and systems that use design grammars, logic and production systems were developed. These approaches can be roughly characterized as attempts to computationally apply mathematical formalisms and AI knowledge-based technology directly to design as problem solving. These capabilities, important though they are, only capture routine or repetitive design and do not support the average complex design

^ychener [Rychener 88] has independently identified three generations in computational design research that correspond very closely to what is presented here.

process in a fundamental way. The third wave of CAD recognizes that design processes have unique and significant aspects which are distinctly different from other types of problem solving. A clearer understanding of the requirements of design processes and the set of evolving computational technologies, such as knowledge-based systems, provides an opportunity for exploring sophisticated applications of these technologies and a promising agenda for current design research.

Perhaps the two most important aspects of design that distinguish it from other types of problem solving activity are the number and complexity of the representations required and the open-ended and exploratory nature of design processes. Our understanding of these as critical research issues is the key to building more effective design systems. Further, we believe that these issues have a fundamental connection to the nature of, and possibilities for creativity in design. We argue that a pragmatic and productive approach that design research can take right now is to build *computational design support environments* which combine human and computer design capabilities in a complementary, integrated manner. In this paper, we propose a conjecture that addresses design creativity in this context.

This approach to creativity in design is illustrated in the context of a partially automated design system for "generic" layout tasks. The system, LOOS, overcomes some of the limitations and compromises of former approaches to layout design specifically and knowledge-based design in general. The generative capability of the system is formally modeled as systematic enumeration of structural design descriptions. This is combined with a human-guided evaluative capability that is built-up incrementally, evolves and can be customized. This division of labor within an overall integrated design process is crucially supported by a representation that captures appropriate levels of abstraction. The observation and evaluation of partial designs in context stimulates the designer to reformulate the layout task in terms of performance requirements and their relative weights. Further opportunities for exploration and knowledge-acquisition will be afforded by a framework now under development for the flexible decomposition of layout tasks. We argue that the computational design model incorporated in LOOS begins to satisfy some of the key features required of an evolutionary computer supported design environment that will support the achievement of creative results.

2 Computer Supported Design and Creativity

Design and creativity go hand in hand in the sense that a designed artifact or a process of design is termed creative by its relative newness to the user, the domain or period in history. Creativity is recognizable only in a given context, thereby making its definition elusive. The context specific (individual, temporal and spatial) nature of what is considered creative is inescapable[2]. In this paper, we view humans and computers as information processing systems

[2]The phrase "reinventing the wheel" captures this idea very well. However, we also know that due to temporal constraints it is sometimes impossible not to reinvent the wheel; avoiding such repetition requires access to the long term memory of human (design) history, which, even if it were available would make searching for relevant insights, in the typical case, intractable.

and **argue that** with our current understanding of processes of creativity and current theories of design the most pragmatic question to ask is, "How can we enhance the potential for design creativity within a human-computer design environment?"\pagebreak

Within an information processing perspective, the objective of enhancing creativity leads us to the following conjecture:

> Creative solutions to a design task occur either when a new formulation of the task is generated providing new solutions, or when a solution is found within a given formulation in a region of the space of design solutions never examined before.

Given this conjecture, the ability to be creative can be enhanced when either or both of the above processes are systematically supported. We believe that currently computational systems can play the most effective role in enhancing creativity within human-machine design systems where the division of labor between the two participants supports exploration of problem formulations and solution spaces. We argue that this motivates investigating the *grounds(emironment)* for the occurrence of a creative act from which *the figure* of creativity itself may emerge.

Our approach to investigating the grounds for creativity is based on the current understanding of the participants, the human and the computer, in solving design problems. To this end, we examine cognitive models of design and computational models (largely implicit in systems that have been built) to summarize this understanding. We then identify an initial basis for integrating human and computational design strengths. We assert that the task of designing a human-machine environment for design is a design problem in itself that should be viewed from an evolutionary perspective. Further, such environments are needed to serve as an empirical testbed for testing conjectures about design and creativity. Design research structured around such environments will produce not only cumulative results, but also new goals in enhancing the occurrence of creativity in design.

2.1 Cognitive models

Cognitive modeling has its origin in information processing theories of human problem solving [Newell and Simon 72]. Modeling process behavior is the underlying objective of cognitive science. Use of methods from cognitive science, such as protocol analysis, in engineering is relatively new. Architects were the first to explore these possibilities [Eastman 69, Akin 86]; software design is another discipline that has used protocol analysis [Adelson and Soloway 84]. More recently,it has been applied in mechanical engineering and civil engineering [Ullman, et al. 88, Baker and Fenves 87]. Research in cognitive models of design encompasses two levels. Most common is the construction of fine grained models of problem solving in terms of design operators, heuristics and the design problem space; the granularity of analysis and models vary. A second approach attempts to characterize the "generic" aspects of design tasks that belong to the core of tasks commonly understood to be design tasks (such as mechanical design, architectural design, software design) [Goel and Pirolli 89]. This model is coarse grained and complements the fine grained models. Further research in these areas has

to be undertaken before complete cognitive theories of design are proposed. We present below a summary of assumptions common to cognitive models, the characterization of the design process that emerges from such studies, and the implications of these studies for understanding the abilities and limitations of the human problem solver performing (creative) design.

Assumptions

The following are the basic structural assumptions underlying cognitive models within an information processing theory of human problem solving(including design) [Newell and Simon 72, Simon 73, Goel and Pirolli 89]:

1. the structure of the information processor consists of an short term memory, a long term memory, and a processor for manipulating symbols; there exists a bottleneck in the transfer of information between short term memory and long-term memory.
2. the design problem-solving environment consists of a design problem space, design task environment and the information processing mechanism(human).
3. the design task must be solvable within reasonable time constraints and limits on resources.

The level of detail at which the design task environment is described or circumscribed for each study varies; this leads to the difference in granularity of the models proposed

Process of Design

Design tasks are prime examples of ill-structured problems where a large part of the problem appears to deal with the discovery of the very constraints or requirements that will be brought to bear on proposed solutions. This *exploratory* progression of a design task is captured nicely by Rychener in the following description of Simon's observations on the nature of ill-structured problems: [Rychener 88, Simon 73]

> " ..Architects [designers] were observed to formulate many items of information of importance to their designs only by recognizing their applicability while working out the details of a solution. They could not start out by making a list of criteria that they were seeking to satisfy with the newly-created layoutfdesign]. Thus the search for a good solution is also a search for proper information with which to evaluate it.

Conclusions about the cognitive process(es) of design that emerge from cognitive studies support and elaborate on the exploratory characterization of the design process [Adelson and Soloway 84, UUman, et al. 88, Goel and Pirolli 89, Baker and Fenves 87]:

1. Design activity falls into three categories: generative, evaluative and patching. These permeate design processes in multiple ways, but some prominent examples are the generation of partial solutions, evaluation of proposed solutions (partial and complete), and patching incompatibilities in solutions that arise when synthesizing partially dependent decomposition modules.

2. Control of design processes by human designers is dynamic, flexible and not easily characterizable as algorithms. The decision making mode is largely one of least commitment intertwined with nested cycles of generate and evaluate[3].
3. Human designers use a variety of symbol structures to achieve their problem objectives and use external memory aids.
4. Abstraction hierarchies are extensively used and appear to function coherently in the context of a problem; however, there exist no formal theories of abstractions yet.
5. Extensive problem restructuring occurs in non routine design, and the problem structuring task and problem solving tasks are inseparable. More design time seems to be spent on structuring than on solving.

We will use these conclusions about the nature of human design processes as a basis for comparison with the nature of design processes currently captured in computational models. But first, we examine how this characterization of the design process reflects both the strengths and weaknesses of the human designer.

Limitations and capabilities of the human designer

The following limitations in the human designer's behavior are a natural consequence of the cognitive limits imposed by the structure of the human information processor:

1. The human fails to adequately manage the complexity of the design problem space through inadequate retrieval of relevant and useful design knowledge from memory. Here, memory may consist of design material from the current design session, such as partial alternatives generated, from previous experience or from design histories available to the designer. This failure is due to a limitation in the transfer of information between short term memory and long term memory [Ullman, et al. 88, Goel and Pirolli 89]. External memory aids are used to some extent to ova-come this limitation.
2. Human designers tend to make the first generated initial solution work or attempt to take a depth first approach to solving a design problem. How the generative process of human designers works is not discernible in the protocol studies of single designers. [Ullman, et al. 88]

Current external memory aids are inadequate in overcoming problems of bookkeeping required in a complex design task. The impact of this failure is acute in that it restricts the extent of the design problem space that is explored, and, even in problems with known structure, it restricts the extent of the solution space searched.

[3]We agree that human designers employ a least-commitment strategy for decisions but only within the scope of a particular solution scheme. Within that scope they seem able to practice least-commitment in component, constraint and value selection. However in the generation of solution schemes for development, human designers appear not to employ least-commitment but what might be called "greatest-commitment". Other findings in cognitive studies of design concur with this [Ullman, et al. 88], as we cite below. It is not entirely clear from Goel and Pirolli*s article what scope they intend , but they seem to imply that least-commitment is used broadly by human designers in the design problem space. We disagree and illustrate in this paper how computational design systems can be complementary to human design abilities in this regard.

Apparently, human designers overcome these limitations to deal with the complexity of design tasks through a variety of cognitive devices such as abstraction mechanisms for generalization and task decomposition, heuristics, analogies and metaphors to provide alternate formulations, and learning and knowledge compilation to evolve and improve the quality and efficiency of the design process. The skilled use of these types of cognitive devices has come to be associated with what is typically identified as expertise and "creativity" in human design practice. Thus, in a definitional sense, human designers employ creative processes to achieve better results *because* of their cognitive limitations, not *despite* them. But the use of a "creative" process, corresponding to one of the above cognitive devices, may not be a necessary or sufficient condition for producing a creative result (in the sense of a distinctly different or superior result). When we assess the potential that computers have for aiding the design process a more important concern is the achievement of "creative" results, not whether the means employed are creative in the same sense as human design processes. In spite of the cognitive mechanisms that human designers employ, the complexity of design tasks and the time constraints involved often restrict exploration and synthesis of "creative" (the best) solutions. Historically, creative solutions appear to have been proposed either by systematic study or by "stumbling upon a solution". As was pointed out, the solution generation process of human designers is still poorly understood in the context of their limitations.

12 Computational Models

The arrival of modern computers led to the widespread belief that they constitute a revolutionary tool that ultimately will enhance and transform the design process. However, our understanding of the strengths and limitations of computers and the relationship of these to creating an effective computational design environment has been progressing relatively slowly. As symbol processors, speed of processing information and an enormous capacity for memory were the obvious strengths of computers that were immediately recognized. Perhaps not obvious at first, it has been slowly recognized that a very significant limitation of computers performing design is their lack of capacity for evaluative judgment and selection, and their lack of what has been called "commonsense" knowledge. Without this broader and incisive knowledge it appears to be very hard for computers to employ strategies akin to the mechanisms for abstraction, analogy, etc. that appear to be key abilities of human performing design. On the other hand, since the proper goal of computer-aided design is to produce superior designs regardless of the process employed, then the expectation that computers imitate or duplicate human processes may be mistaken; entirely new processes for producing better designs may emerge that depend on utilizing the strengths of computers in ways that complement (or surpass) human designers abilities but bear little resemblance to human design processes. No doubt, for the above reasons, and because investigation of computers as design agents is in its infancy, few explicit computational models of design exist[4].

[4]**We are excepting here the deliberate attempts at computational realization and validation of human cognitive models of design; these models are not concerned with producing designs per se, but with the validation of cognitive theory.**

The early CAD systems that were used for detailing designs from a geometric point of view **were** based only on implicit and primitive computational models of design. Following these there have been numerous experiments in combining algorithmic and heuristic methods in design systems yielding more explicit and/or slightly less primitive computational models. For instance, automated design systems have approached design from a top-down model of refinement [Steinberg 87]. In this approach, the design problem is structured as a fixed top-down decomposition (a well-understood problem) in which the computational system manages complexity by synthesizing solutions from a set of alternatives available for each of the subproblems in the decomposition. This approach, and that of using constraint representations and processing, have primarily yielded a series of design systems with much more explicit models, but ones largely limited to capturing routine design, as acknowledged by the developers of these design systems

More recently design research has been directed toward using AI technologies, particularly in the form of knowledge-based systems. These systems succeed, after a period of knowledge engineering, in giving the computer some (impressive) narrow design expertise and evaluative power. However, they still suffer from limitations of rigidity and brittleness articulated as general criticisms of the "expert system" approach to problem solving [Doyle 84, Rich 84]. To counter this, another round of knowledge acquisition and refinement can always take place in principle; this is not usually planned as a continual and well-integrated part of mature systems but would be required for design tasks of even minimal complexity. To give these systems some range, flexibility and greater efficiency, various strategies for employing layered knowledge bases of meta-knowledge and meta-level reasoning and control have been incorporated with some success [Orelup 87, Takewaki 85, Genesereth 83, Stefik 81].

Under such arrangements the computational system performs in a less primitive and more sophisticated way, but the underlying model tends to be less clear especially when dependent on domain specific meta-level concerns. Implicitly these systems are based on the as yet unfulfilled hope that a limited number of domain independent meta-levels with corresponding inferencing strategies and knowledge will be discovered. But meta-level inferencing often seems to require guidance from yet another successive meta-meta-level and so on. Effective computational models of a complex task will have to cope with the problem of organizing and effectively applying large amounts of knowledge without requiring more knowledge at ever higher levels to be applied [Levesque 86]. Therefore, problems of knowledge acquisition, representation and retrieval in knowledge-based systems still pose major challenges if computational systems are to obtain an autonomous capability for design and creativity as we know it in humans.

Some recent design research is concerned with modeling computational techniques that focus on flexibility and innovation as more central issues. In attempting to overcome the limitations of predefined solution spaces and knowledge bases however elaborate, some researchers propose mechanisms that produce alternative formulations and sets of solutions.

These include techniques proposed and implemented for "creative" design such as generation of analogies [Zhao and Maher 87], prototype modification [Gero et al. 88] and modification operators [Addanki and Murty 87]. These proposals approach the problem of design and creativity by defining and modeling computational techniques for cognitive design devices such as analogy with the hope of understanding and generalizing them over a collection of tasks. Through these generalizations computational theories of design are expected to emerge (eventually) from the "bottom up".

Computational models of analogy, metaphor and other learning techniques can aid the designer with alternative problem formulation. However, postulating generative models alone of cognitive devices such as analogy is insufficient as their use may result in a combinatorial explosion (of analogies produced for instance). Any mechanism that generates a set of solutions has to be governed by means for selecting the useful instances out of the set; this selection mechanism depends on the performance criteria derived by trading-off problem objectives. This behavior has been observed in an empirical study [Kalagnanarn and Subrahmanian 89], and has also been articulated in a critique of domain independent learning techniques by Doyle [Doyle 88]. He argues that the domain independent learning methods are devoid of rational objectives thereby generating generalizations that are not useful. This point is well described for analogies by Simon in his essay on the architecture of complexity:

> Metaphor and analogy can be helpful, or they can be misleading. All depends on whether the similarities the metaphor captures are significant or superficial.

The discrimination of the significant from the superficial depends on evaluative criteria corresponding to the objectives of the problem being solved. Precisely for this reason, purely generative approaches to design innovation implicitly still require the human to participate in the process. While requiring the integration of human evaluative and behavioral design knowledge, these approaches have not (yet) delimited appropriate roles for the human designer and for the the machine; further, they lack a comprehensive perspective for integrating these techniques into a complete design process.

From our review of computational models, we identify two crucial points that we believe are necessary in developing computer-based models for design. The first point is to explicitly acknowledge and provide for a partial dependence on human design expertise and the careful integration of it with currently understood computer capabilities. Therefore, for the creation of design environments we adopt a design research strategy similar to what Rich [Rich 84] proposes as the gradual expansion of artificial intelligence. In describing the role Al-based systems play currently, she points out that even though part of the responsibility has been assumed by machines in solving problems they do not eliminate the need for the human in the system. Along these lines there are other significant approaches to understanding and aiding design based on the premise that design is an *exploratory* process and that the role of the computer is to aid the designer [Hemming 88, Smithers, et al. 89]. Flemming, in identifying the role of rule-based computer systems states:

> ..for tasks that are not well understood a rule-based system can serve as an effective vehicle to deepen our understanding and can thus lead to regularities, to generalizations and to the formation of theories that do not exist at the outset.

The **model** of Smithers, et al. goes beyond rule based systems and argues for designing systems based on a prototheory (design as exploration) to understand and formulate theories of design. This is a well thought-out, ambitious undertaking with the potential disadvantage, as the developers point out, that it requires large amounts of resources and time, because achieving a useful degree of intelligent support involves a greater range and capability than is typically required of approaches adopting automated design. We will have more to say concerning the relationship of such models to our conjecture in the next section.

The second point, as articulated by Doyle [Doyle 84], emphasizes the need for understanding the nature of apprenticeship versus that of constructing journeymen. He refers to expert and knowledge-based systems as instant journeymen who do not have a model of the apprenticeship process and hence do not have the capability to become masters; that is, they sometimes display impressive expertise but have no inherent structure or agenda for evolving and learning, for bootstrapping their knowledge and experience into more knowledge and better skills, or for ever acting in flexible and innovative ways. Even though his proposal is to study the apprenticeship process without computers, we believe (as the editors of his essay also note) that computers can play a role in understanding the apprenticeship process. In order to achieve this objective, we believe we have to endow computers with the facility to generalize, observe and experiment with problem solving theories in a domain. We view this process of transferring the nature of apprenticeship to computers as requiring a symbiotic human-machine system where there is a continual update of the division of labor.

23 An evolving design environment for enhancing creativity

Design of any system that is based on the premise that it display evolutionary behavior does not have an explicit final design goal. In articulating this point in the context of social planning(a not so well structured task), as an example of the design of evolving systems, Simon claims: [Simon 81]

> **A paradoxical, but perhaps a realistic view of design goals is that their function is to motivate activity which in turn will generate new goals.**

We view the task of building a human-machine environment for creative design from a similar perspective. This view is motivated by the limited understanding we have of the activity of design in information processing terms(both cognitive and computational) and the difficulty of identifying objective evaluative criteria for creative acts. This leads us to set minimal goals for the task of designing a *computer supported design environment(CSDE)* to experiment with creativity in design. In this approach we encompass the issues that we believe are currently the most important to support design and creativity, while not prescribing or restricting what concepts, mechanisms, structures, knowledge, etc. may have a place in such environments. We believe that this can be accomplished by building evolving human-computer systems that support the conditions in our conjecture and provide means for continually attenuating the limitations and expanding on the strengths of the participants in the design environment.

A design environment is constrained by the limitations of the participants in the environment. A computer supported design environment, with a single human and

computers) as participants, consists of two types of symbol processing systems: human and computer[5]. We have looked at the conclusions of cognitive studies for the case of the single human designer, and the so far persistent limitations of computers, which may be partially due to the limitations in our computational models for utilizing and instructing computers in design.

With these points in mind, we wish to focus on the potential for CSDEs, still characterized by the strengths and weakness of the two participants, but with a new twist on the role of the computer (and the human designer). In particular, their cooperative integration opens new possibilities for expanding the strengths of the computer when balanced by context sensitive human design knowledge that can be elicited, stored, refined and structured for dynamic application. Earlier, we mentioned that the generative processes of human designers are not easily discernible from protocols, and hence present a rich area for research in for proposing formal generative languages [Ullman, et al. 88]. Within a cooperative evolving environment with the proper division of labor and the right set of representations these can take the form of purely syntactic languages. Such languages can be postulated, developed and tested for completeness, and systematically applied using the main strengths of computers. Of course, the definition of designs through such languages must be completed and balanced by the exploration and application of task specific performance or behavioral requirements by the human designer in order to make a large space of solution possibilities converge and express the desired properties.

Similarly, with the right representations and structure computers can provide multiple abstractions and decompositions for managing and exploring partial design descriptions, and compiled design histories for bringing to bear on the present task, at the right time and level of abstraction, relevant design knowledge from previous design experience. These representations, languages, evolving structure and the design knowledge that they capture will ultimately become interwoven with computers speed and memory capacity as a more unified fabric or computational model expressing computers acquired strengths as design agents. With this perspective, the space of possibilities for the design of such evolving design environments is indeed large. It quickly becomes speculative and unproductive to discuss these issues further in general terms.

In the second half of this paper we choose to illustrate some possibilities concretely in terms of a specific class of design - layout design- and a unique computational model and system for exploring layout design tasks - LOOS. We will show that the LOOS approach takes advantage of existing technologies and human and computer strengths and integrates them in such a way that its architecture reflects an *exploration-based model of design* akin to

[5]In this paper we restrict the scope of our investigation into environments for creative design to two agents: a single human designer and a computer system (possibly consisting of more than a single machine.) Increasing the number of human designers in the design process adds additional complexities involving communication and social processes to investigating the environment of design.

that proposed with careful detail in [Smithers, et al. 89]. Insights from both of these groups confirm that determining the architectural features of a CSDE must be approached as a design problem itself with evolving goals. We believe that the major themes that must be addressed can be identified however and are as follows:

1. modeling design as an exploration process with human/computer cooperation
2. immediate enhancement of the creative potential of human designers coupled with the evolutionary enhancement of the role of the computer toward a more autonomous, creative partner
3. design theory in domains and insights into creativity should be accumulated experimentally and empirically through the normal use of design systems; this will be enabled if there exist sophisticated knowledge structures, representations and languages to elicit, accumulate and refine design knowledge and design process knowledge during single design tasks and across a set of tasks.

Besides being evolutionary in its behavior, the environment proposed is also evolutionary in terms of the history of development of computational design tools; a step that integrates formal problem solving and knowledge-based methods in a manner that allows it to display a continual improvement in its behavior.The CSDE framework that we propose for conducting design research accommodates both "creativity in the large" and "creativity in the small." What we mean by creativity in the large is the overall potential for producing superior results (innovative, creative and custom design) that emerges through the integration of multiple human/machine processes and representation capacities, both long and short term. By creativity in the small we mean the focus on computationally modeling individual processes or mechanisms that potentially produce creative results such as mutation and modification operators, analogy, case-based-reasoning, etc.

We see these two levels of exploration and potential for creativity as perfectly compatible, indeed necessary to one another. A significant and important part of the evolution of a CSDE will be the introduction, refinement and incorporation of more and more individual computational mechanisms for creativity comprising the gradual shift of design expertise from human to machine. At the same time our belief is that the focus on and development of CSDEs as a whole is necessary as an empirical testbed where such mechanisms can emerge as ideas, take-shape and be tested.

2.4 Testing conjectures about design creativity

We now reexamine our conjecture specifying conditions for the occurrence of creative acts in design. Briefly restated, our conjecture again is that the potential for creative design occurring is increased when designers are provided the capability to explore an expanded formulation space or more of a given solution space for a task. In motivating and describing a computational model consisting of evolving CSDEs, we have also proposed the appropriate experimental environment within which to test our conjecture. The construction and planned evolution of such environments will provide the exact conditions necessary to promote our understanding of creativity, and achieve creative results for real design problems. We believe

that any design research proposing conjectures about computational models of creativity must also specify the experimental environments in which to test them[6].

The next section will describe how we can support a minimal CSDE which provides the conditions by which to evaluate our conjecture in the class of layout design tasks. We don't claim to verify the conjecture in the strong sense, but we clearly illustrate it in terms of examples of systematic exploration and task reformulation in layout design. Of course, we realize that in order to fully test the conjecture, we should validate it across classes and domains of design tasks; layout design is just a beginning. Our proposal at the experimental level - constructing evolutionary CSDEs - will allow us to test for each of the conditions of the conjecture, and follow this approach in a number of design domains. Below we add a few further comments about each of the conditions in our conjecture by identifying some types of systematic exploration and task reformulation; these are by no means exhaustive but represent an interesting collection of possibilities. These examples will be further illustrated in the next section.

Expanding the number of potential solutions examined

We note again the limitations of human designers in exploring and keeping track of multiple alternatives and the importance this attaches to constructing computational design systems that enable a designer to explore different options in parallel [Mittal et al. 86, Mostow 85]. As the following quote suggests, there is a connection between the ability of designers to explore a given design space more thoroughly and the potential to discover innovative solutions to a typical problem within a given problem space: [Rychener 88]

> Past design work (which is largely done by people, with few ideas as to how to automate even its more routine aspects) has often been hampered by a failure to consider more than one or two main alternatives. This certainly excluded most truly innovative approaches. It has also meant that the search for alternatives has not been systematic and thorough.

In essence we advocate systematic examination through generation of all solutions, at appropriate levels of problem abstraction, as a means to expand the scope of exploration[7]. Systematic generative capability effectively filtered through performance knowledge, and intelligent design history management will provide means to explore larger portions of a potential solution space than humanly(!) possible.

---

*This is in contrast to approaches that propose models for creativity and use computational implementation as a sufficient condition for verification without any(or very limited) verification of their purported behavior [Ohlsson 83,Bundy83,Sharkey85]

[7]Of course, such a design strategy is potentially cornbinatoric. However, there exists at least one model for implementing exhaustive enumeration of solutions that is tractable if certain conditions are met. *Hierarchical gentraU-und-test* permits pruning of candidate solutions that are only partially specified, and when a partial solution is pruned, an entire class of solutions corresponding to the description is eliminated from the generation process [Stefik et al. 83]. The LOOS system, presented in this paper, is an example of the implementation of this model in the domain of layout. This approach is potentially so important as a computational means to produce superior (creative) designs that we recommend that other domains of design attempt to meet the conditions allowing its use; this requires a representation enabling the specification of partial designs and an architecture that supports their evaluation with certainty.

Expanding the exploration of task formulations

If purely syntactic generative mechanisms arc proposed, the number of potential solutions can be large because they are missing an important part of the design task specification: performance constraints. Two sets of performance constraints with the same structural design descriptions correspond to two different formulations of the task. Performance constraints are an important means by which humans designers manage the complexity of exploring the solution space. By allowing a designer to incrementally define and manage sets of performance constraints, support for expanding the set of possible formulations of design tasks can be achieved. As the shortcomings in partial designs are detected, as a result of incomplete or faulty specification of the current set of constraints, the system could effectively aid the user in correcting the problem formulation. In this regard, design histories can also play a vital role in connecting the knowledge from previous attempts and solutions to the problem at hand. Modification of performance requirements is only one method of expanding the scope of exploration of the problem formulation.

Another important means by which the designer manages complexity is through problem decomposition using abstraction hierarchies. Design and performance constraints can be expressed at different levels of abstraction of the task. Any representational language for design, including generative ones, will have to provide mechanisms to allow for specification of task decompositions. The designer using these languages should be able to express decompositions and allow the computer to generate solutions at these levels. Alternative decompositions would also provide the opportunity for experimenting with differing formulations of a design task. Aid in managing possible sets of design task decomposition at various levels of detail can potentially allow for overcoming the tunnel vision behavior imposed by cognitive and temporal limitations.

In the above discussion of our conjecture we initially assign the roles for blind generative capability and algorithmic solving methods to the computer due to its computational strengths while assigning the role of evaluator and specifier of rational objectives to the human. Continual evolution of this integrated design environment is based on the ability of powerful computational variants of context sensitive human cognitive devices (abstraction, analogy and learning), or purely computational techniques with no cognitive equivalence finding a stable use in the design environment. Techniques that impose additional limitations on the ability to perform design or are not used will fall out of the environment while others that enhance the power of the designer will find a niche in the environment sometimes proving useful beyond their originally intended use. We believe that we can use design environments constructed around these principles to enhance and understand the design process. As our understanding of the design process progresses the role of the computer may evolve from that of an apprentice to a creative partner while in the initial stages they serve as amplifiers of human creative powers.

3 Layout design: a testbed for computer supported creative design

Layout design deals with many of the complex issues that typically arise in the design of artifacts that have to satisfy specified constraints and are composed of parts that have shape and take up space. Because layout involves the two-dimensional composition of objects into floor plans, site plans, etc. it is in some respects relatively less complex than some other design tasks. But layout design is interesting and important because it is a generic phase or process of many design disciplines. A design method for layout should capable of dealing with many or all of these domains if its representation and overall architecture are general enough.

Layout design is also inherently difficult because of the large (potentially infinite) number of location and orientation combinations available for placing any single object. Furthermore, there are multiple interdependencies among the design objects imposed by their shapes, sizes and the spatial relationships required to meet multiple performance requirements. For these reasons there is no known direct method which is guaranteed to produce feasible solutions without trial and error. Depending on the difficulty of the problem some amount of exploration of problem structure and search for candidate solutions is required whether in human or computer based approaches.

Recurring in the literature on layout design is the observation that due to cognitive limitations human designers do not have the capability of making systematic explorations of alternative arrangements [Liggett and Mitchell 81, Galle 86]. A distinctive feature of layout problems is that decisions must simultaneously satisfy global requirements (e.g., usage of space) and local requirements (e.g., adjacency); an acceptable spatial arrangement results from a complex pattern of tradeoffs. Ideally, what is desired is a structured method for producing multiple alternative layout solutions where each proposed layout embodies tradeoffs that can be understood and justified, location decisions are made explicitly, and possible directions for variation are indicated[8].

Because of its cross-disciplinary importance, its inherent complexity, the fact that only restricted 2-D representations are required, and the need to augment human methods, layout design is a remarkably suitable design task for the initial implementation and study of a CSDE. In this section we describe LOOS, an approach to supporting layout design, and its properties in relation to our conjecture and the requirements for an evolving CSDE.

---

[8]It is for these reasons that there have been severe drawbacks to the representation of spatial allocation problems in mathematical formulations such as quadratic-assignment or mixed-integer-programming. Typically, not enough is known up front about the constraints to formulate layout problems for this type of solver. These approaches may be suitable where the set of requirements is strictly limited (such as where distance or material flow is the overriding concern) but performance requirements impacted by considerations of adjacency, shape, size, alignment, relative location, etc. and their tradeoffs are very difficult or impossible to model aphorL Once the correct formulation for the problem is known these solving methods are probably the most efficient to employ. They can no doubt be used to optimize for certain requirements after a rough or aggregate layout is known.

3.1 The LOOS system: human and machine cooperation

The following description of the LOOS system does not emphasize technical details or the current realization of all of its components (which are evolving). Rather, the intention is to provide the minimal explanation of the approach that will allow us to examine its basic organization of the design process. For further technical detail and substantiation, a recent full description of its implemented status (at that time) and the important theoretical basis for its representation and generator can be found in [Flemming, et al 89]. The primary design goal for LOOS was to build a partially-automated system combining human expertise with the systematic generative capability of computers. An initial system realizing that goal is implemented and running on real problems from several domains.

The core of LOOS is a process that composes a set of 2D objects into one or more layouts of rectangles. This process is designed around a graph-based relational representation that facilitates the systematic generation of all candidate solution layouts and a separate tester which supports evaluation of candidate solutions over a wide range of criteria. Application of the system to a layout design task requires acquisition of a designers' knowledge and provides them with a set of alternatives that capture tradeoffs. The systems' components are a generator, a tester or evaluator, and a controller. Each was designed to be an independent module so that various evaluators and controllers are easily experimented with and replaced irrespective of and without disturbing the other components.

3.1.1 Representation and generation

The representation on which the approach is based uses the basic spatial relations above, **below, to the right** and **to the left** to define the structure of a layout and represents this structure formally through a directed graph whose nodes represent the rectangles in a layout and whose arcs represent spatial relations between pairs of rectangles; Figure 1a shows an example.

A set of generation rules has been developed to insert "rectangles" into a graph, one at a time, and to construct in this fashion every set of realizable relations (for a given number of rectangles) as an alternative layout structure; a sample sequence of rule applications to generate partial layouts is shown in Figure 1b. The rules constitute a *generator* that is a formal grammar for the generation of 2-dimensional syntactically correct layouts of rectangles, where the rectangles themselves are unattributed.

Spatial relations, as defined here, formulate a type of intermediate abstraction for describing layouts that reflects a separation of structural from dimensional concerns. Each representation suppresses the continuous dimensional properties of potential layouts and expresses the discrete spatial relations between objects. Each candidate layout thus represents a class of feasibly dimensionable layouts and the possibly infinite set of solutions is divided into a finite set of subsets that can, at least in principle, be enumerated. This representation captures a suitable notion of structure that can be used to define critical differences between alternative layouts and to generate alternative structures; this allows the generation of a well-

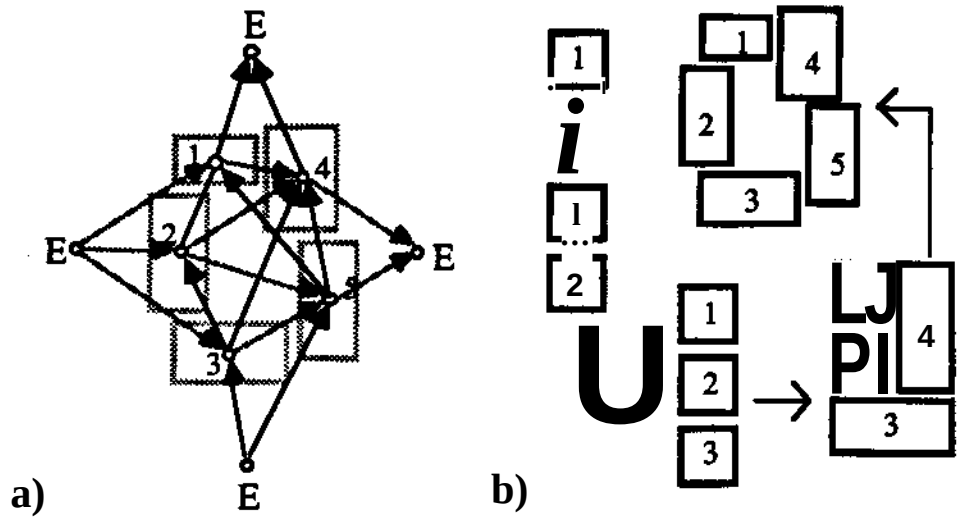

Figure 1: Formal representation and generation of arrangements of rectangles

defined space of potential solutions. Higher-level concepts regarding the quality of layouts can be formulated based on this representation so it also serves as an explicit structure for queries, analysis and evaluation.

The representation also supports the incremental monotonic specification of designs in terms of their structure. The formal properties of the generator insure that each partial layout generated is formally complete and distinct and that the spatial relations between objects already allocated will not change with the addition of further objects; alternative relationships between a given pair of objects will be found on other branches in the space of possible structures.

A restriction of the representation is its limitation to a rectilinear world. However, the representation achieves significant formalization and simplification given the present state of the art, and it permits numerous interesting applications at various levels of complexity across many design disciplines. Another potentially more serious restriction is that the representation, by itself, does not guarantee that the exploration of the potentially vast space of possible solutions will be efficient and computationally feasible. However, here again the representation is extensible to support hierarchic recursive processing of tasks in order to offset the possible combinatorics of generation. This extension, described in the next section, supports the representation and modification of the decompositional structure of layout tasks.

3.1.2 Layout task decompositions and their representation

In this section we briefly describe the extended representation framework that will support an expanded design process. A prime motivation for the proposed framework is to deal with the problem of scalability and the combinatorics of the generation algorithm; the intent is to manage the complexity of problems both in the numbers of objects handled and in the amount of interactions and detail involved with those objects. To handle complexity, the

experience of both human and automated methods suggests that a very important abstraction mechanism in layout design is a hierarchical decomposition of the objects it handles and of the tasks (or goals) it pursues. An approach, described below, is currently being implemented [Coyne 89] to achieve the decompositional structuring desired within the LOOS approach; this research is also converging with many of the other goals for creating a CSDE.

The system, as first developed, treats the underlying representation *as flat:* each node represents a single, physical object at the same level of abstraction. But this need not be the case. Loosely-packed arrangements of rectangles can represent layouts not only in different domains, but also at various scales and decompositions. A rectangle might represent a single unit or a cluster of rectangles each of which might again represent a cluster. The formal representation used in LOOS supports this recursive structure. Graphs are particularly amenable to this kind of treatment because their nodes can easily be expanded (or contracted) through appropriate operators. Furthermore, the generation rules employed by LOOS can be used recursively not only to insert nodes, but also to expand nodes formerly inserted.

In extending the representation to structured objects, each node, or layout component, also defines a layout *goal* or *task,* namely that of allocating the parts it contains. Decompositions in the domains of layout design can be understood in this dual function: they partition both the *object* and *process* of design. The merging of goals and design structure in terms of both function and hierarchy leads to the concept of a powerful integrated abstraction, a *goal-object* (GOB). Each GOB is self-contained and represents a complete design task, and the knowledge and representation structures and set of mechanisms with which to accomplish it These include (but are not limited to) startup and terminating conditions, design method and control selection, evaluation criteria, and the definition of the further unfolding of its own abstraction subtree (subGOBs). A layout task can now be planned as a decompositional hierarchy of subtasks corresponding to a nested tree of GOBs. The overall task is specified as a single top-level GOB. The decompositional structure for the layout task will partition this goal into subgoals at various abstraction levels. The design process will unfold as the *recursive, hierarchical* satisfaction of the layout goals through the placement of GOBs. We will elaborate on how this structure can be utilized in dynamically controlling the problem decomposition and the solution process in the next section.

3.13 Evaluation and control

Controlling the syntactic generator

Given that a well-defined space of potential solutions can be produced, LOOS employs a domain-dependent knowledge-based approach for evaluation. It discriminates among the generated candidates based on multiple criteria that capture the semantics for the quality of layouts in the particular domain and problem. This tester, which we also call a *semantic filter,* is designed to be built up incrementally as designers are stimulated to add, change or withdraw criteria when confronted with partial layouts graphically displayed and rated.

The semantic filter in LOOS serves two critical purposes. First it completes the

definition of partial design descriptions by adding behavioral interpretations to the structural alternatives proposed by the generator. Second, the filter must provide adequate knowledge for discriminating among alternatives so that the space of possible solutions is sufficiently reduced to make its exploration feasible and efficient. As is the case in most design tasks, the challenge is to fashion a set of criteria, neither overconstrained or underconstrained, that balances the set the requirements or behavioral qualities that the designer is seeking; tradeoffs may be necessary.

Figure 2 shows the expansion of a selected node (partial design) by the generator producing a set of alternatives, structurally defined, which represent the placement of another object in the layout in all possible ways. The figure also shows that with the same structural expansion the substitution of a different semantic filter changes the descriptions of the partial alternatives produced. Changing the evaluative criteria may also change the resultant partial designs that may be chosen for expansion at the next level of development

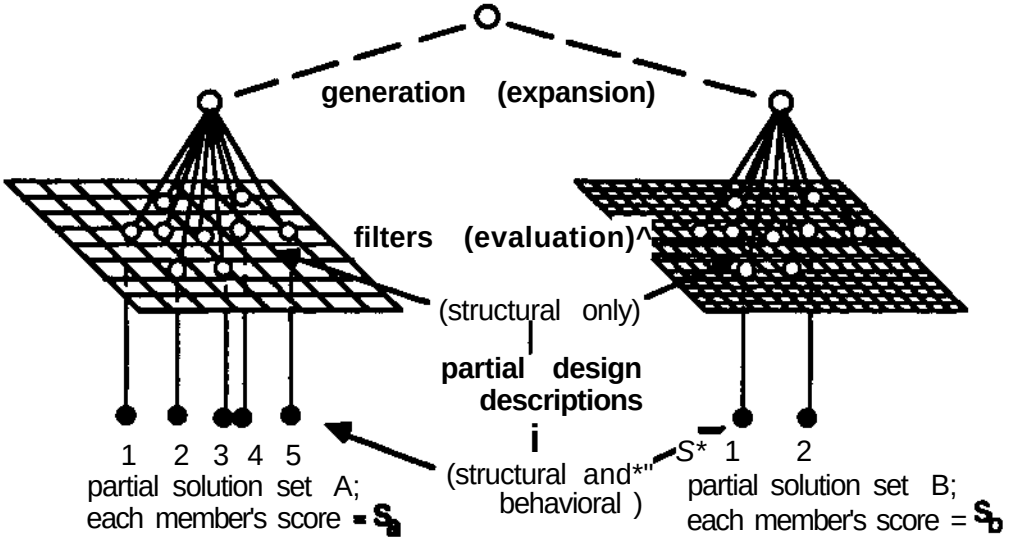

Figure 2: Generation and interpretation of partial solutions

The results of evaluation are stored in an *evaluation* record. In the current system this an ordered three tuple consisting of the number of constraints, strong criteria and weak criteria violated. It also captures the relative ratings for partial designs in terms of individual requirements. Constraints indicate a requirement that a solution *must* satisfy in order to become acceptable; strong and weak criteria indicate desired properties of decreasing importance. Evaluation scores for partial designs are interpreted lexicographically using the ordering in the tuple or the evaluation record; that is, there is a relative left to right ordering of importance. The arity of the tuple and assignment of importance of criteria to the elements in the tuple is up to the judgment of the designer using the system and is a matter for

experimentation and knowledge-acquisition. Tradeoffs among design criteria are accommodated by their relative weighting as reflected in assignment to a category in the evolution record.

A *controller* uses the results of evaluation and a simple branch-and-bound(BNB) strategy to steer the traversal of the space away from less promising alternatives deferring their development until it is certain that nothing better can be found. Using this strategy, the controller expands those and only those intermediate designs which are at least as good as any other design generated before (independent of its level of development in terms of number of objects placed), and prunes the space of candidate solutions.

The structure of the evaluation record, and the rating of partial designs in terms of it, facilitates the ordering of solutions with respect to *failing* requirements. The idea of a filter captures the notion of applying the semantics of the domain in question as a negative screen to the alternatives produced by the syntactic generator. This notion of guiding the expansion of the space of alternatives based on failure of requirements is an important one in design tasks involving multiple complex requirements with both global and local interactions. As a result of this strategy and exhaustive enumeration, LOOS is able to manage decision making and commitment in a unique and efficient way. As noted, it asserts the structural definition of candidates *monotonically* to produce new points in the potential solution space. Since all performance requirements are evaluated in terms of this structural definition, this guarantees that once a partial solution fails in terms of some performance test, it can never get better through further development and its performance is monotonically rated (as an increasingly negative score.) Taken together,the representation, generator, tester and controller of the LOOS approach implement a form of *hierarchical generate-and-test* [Stefik et al. 83 ,p. 72]. The architecture of LOOS and the domains for which it is intended satisfy the conditions that make this approach workable; in particular, the generator partitions the solution space in ways that allow early pruning, and partial solutions may be evaluated with certainty.

Therefore, LOOS avoids making uncertain commitments(in the form of predictive decisions about performance), by maintaining a "currently globally best" set of candidates and always expanding that set in the minimal way in the next round of generation[9]. This is in contrast to design approaches which intertwine performance requirements in a generator and employ a least-commitment approach in all choices; their problem is that when least-commitment breaks down they still have to commit to a choice in proposing an alternative (and later backtrack if it is wrong.) LOOS makes no commitment in choosing which of the current globally best partial designs should be expanded and their children examined - it examines them all. In this way the system can again commit with certainty to a new set of current globally best candidate solutions, with no backtracking. This *least-commitment of resources* approach is effective in the face of multiple, complex interactions among the goals of layout,

[9]This sounds very memory intensive but since each new insertion only changes a parent layout graph locally we inherit much of the graph structure through the hierarchy.

the **global** uncertainty of outcome until all objects are placed, and the need to systematically explore tradeoffs among multiple candidate solutions.

Controlling problem decompositions

The extension of the flat representation to GOBs provides an opportunity both for decomposing an overall layout into components and for controlling the complexity of the representation at various levels or scales. All objects in a domain of layout, whether primitive or structured, will be represented as GOBs. Viewing GOBs (structured objects) alternatively as primitive or as subdesign tasks to be further developed provides a uniform protocol and drastically simplifies and decentralizes control. It will also permit the overall task to be decomposed at non-uniform levels of abstraction, where some GOBs are slated for development and others are not, even if of the same category; in this regard a decomposition scheme suited to a layout task may be arbitrarily complex. For instance, in building layout placing "rooms" within a building may be the final level desired, while in another design task the detail of the layout within each room may be needed; or, internals of some rooms may need to be placed while others are left undetailed. This provides designers great flexibility to represent (and experiment with) the degree of detail and level of completion desired in layout tasks. The structure of a given layout task can be represented and is customizable through selecting the development level of the GOBs that comprise the task. This decomposition strategy supports both top-down and bottom-up control of the design process, such as moving from coarser to finer scales or allocating parts that have been built from smaller components.

Many design systems advocate and depend on the strategy of decomposition or subgoaling to deal with complexity; however, very few deal with problems of recomposition, or the smooth integration of subgoals to solve an overall goal. To provide a method that allows dynamic, and flexible resolution of interaction between subtasks the GOB architecture will allow some limited bottom-up processing to occur with the overall top-down control. For instance, this could happen opportunistically by pushing down a level of abstraction and dealing with the articulated subcomponents of a GOB in order to retrieve information or resolve an interaction at the higher level. The use of GOBs in this manner offers some opportunity for bounded negotiation and communication between "leaky" subgoals to capture the advantages of subdividing a task while providing for realistic integration of design problems that are only "nearly decomposable". Also, related to recomposition, there will be the possibilities for the specialization of GOBs in context (based on a prestored-stored hierarchy or parameterized instantiation), and for generalization through the aggregation of components (freezing a portion of the design) into a single object at a higher level of abstraction.

3.2 The conjecture illustrated

We now discuss the critical capabilities for the support of layout design in LOOS that begin to give it the character of CSDE. They serve as a foundation upon which to build a design environment that will evolve and support creativity in an incremental and cumulative way.

Expanding the examination of a given problem space for potential solutions

We have pointed out the limitations of human designers in exploring a large space of solutions for a given problem. To overcome this, LOOS enables a designer to thoroughly examine the potential solutions for a given formulation of a layout problem. It does this by combining systematic generative capability and the abstract representation of partial designs with an evaluative knowledge-base. Leaving aside the development of the knowledge base(s) for a particular domain, once they exist, designers are enabled to easily look at many more potential solutions, and find distinct alternative solutions faster[10] for a typical problem than they could using only their own resources. Depending on the complexity of the problem this could substantially reduce their expenditure of resources (conserving these for more creative application to the design task at hand) while significantly increasing the number of distinct solutions considered. We believe that the capability for systematic examination, though not creative in itself, underlies the creative potential at a basic level; that is, it provides the grounds for its emergence rather than being the creative act itself.

An expansion of the number of problem spaces explored for a given layout task

Within LOOS*s design process designers are also able to explore potential solutions for a given layout task in a variety of problem spaces. They are able to change the problem space within which they are working by changing the requirements or behavioral description of (partial) layouts in three ways:

1. by adding, changing, and withdrawing evaluative criteria.
2. by adding/changing measures of importance for particular evaluative criteria
3. by adding or withdrawing categories of importance in the evaluation record (to which to assign the individual evaluative criteria)

Each of these can contribute to modifying the semantic filter through which the structurally defined and generated partial solutions are passed, as described and illustrated in Figure 2. Examples of evaluation requirements that might be added for a kitchen layout task are "The back of the refrigerator cannot be placed against a wall" and "The sink has no space on either side for a work counter"[11]. The following list suggests additional levels of importance and corresponding categories of performance requirements that might be added to achieve finer grained discrimination of alternatives for a typical architectural layout task:

1. well-formedness tests, such as dimensional fit
2. code tests
3. good practice
4. client preference
5. office style or policy
6. designer preference

[10]Just as an example, eight distinct solution alternatives are found by the system in less than a minute, real time, for a real kitchen layout task within a non-rectangular boundary. This performance was on a Sun 3/60 with 24 megabytes of memory with LOOS running on top of Unix (Mach), Lucid CL 2.1, PCL and XI1. .

"Remember that these are framed as negative tests and scoring reflects failing requirements.

This possible extended hierarchy of requirement levels would be reflected in the extension of the evaluation record tuple. The significant thing about the elaboration of levels of possible evaluations is that those that reflect somewhat arbitrary constraints or criteria (designer preference) could contribute to balanced discrimination on problems that would otherwise be overconstrained or underconstrained. The LOOS approach will readily accommodate such customization, and its applicability will depend on the domain in question.

For particular domains of layout, the construction of a hierarchy of evaluative knowledge-bases, corresponding to levels such as the above, would provide a means for individuals or firms to create and store their "intellectual design capital." This possibility reflects the opportunity for designers to conveniently intermix a variety of personal and institutional evaluation and stopping rules in design. The same iterative generate-evaluate-modify cycle may be used to develop these knowledge bases or customize an existing one. We will have more to say about the role that such knowledge bases might play as part of design histories later.

The interaction of designers with a graphic display of partial designs is an important example of raising the abstraction level at which computer tools can provide external memory aids to the designer. The exploration of alternative problem spaces in a developing design task, with the close interconnection of long term memory aids (from internal and external sources through a noticing and evoking process), might be considered a *creative process* in itself. At the very least it will have significant direct impact on the potential for producing *creative results* in the overall human/computer design environment.

Expanding the exploration of problem spaces through decomposition

Each GOB has a set of constituent objects, other (sub)GOBs, whose placement make up its layout task. A designer may experiment with alternative decompositions by assigning an object as a constituent of one GOB as opposed to another, or by making an object active in the layout at a different level of abstraction, for instance at the same level as a former "parent" GOB. In this way designers are afforded the flexibility to capture and experiment with the granularities of objects and layout tasks within a domain until they find a suitable decomposition relative to the constraints of their present task. This is a familiar process in many design domains where there is frequently the option to choose a standard structured object from a library or design database, or to custom build in context from a more primitive set of components. For specific critical GOBs it may be desirable to build up and record ordered alternative decompositions (sets of constituents) so that successive ones may be tried dynamically in the event that sufficiently good evaluations are not achieved by the prior one.

Initially within a domain, the human designer will provide the decomposition knowledge and enter it into the system in the form of GOBs, their development levels and constituents. This will involve another level of knowledge acquisition and the potential for exploration of alternative problem spaces in terms of the structure of the task. Over time and problem runs a knowledge base of GOBs will be accumulated in a domain and constitute accessible long term memory usable to help structure any given task, and of course customizable to the task.

Supporting Diagnostic or patching activity.

The LOOS architecture, in addition to supporting generative activity, also supports patching activity by serving as a diagnostic, evaluative system (in domains where an evaluation knowledge base has been developed.) Designs, produced by either manual or automated means, are easily converted into LOOS's internal representation, submitted immediately for evaluation and the results displayed with an explanation of any violations. It would also be possible to remove a design object or objects deemed most responsible for a failure (by backward application of generation rules) and to re-submit the reduced configuration, along with the object(s) to be inserted, to the generator in order to obtain additional alternatives.

Along these lines, it is easy to imagine how an interactive environment for using the LOOS system in multiple ways could be constructed. The device of removing a (critical) piece from the layout and re-inserting it in all possible ways could be developed into a form of *prototype adaptation* [Gero et al. 88][12], and bears a strong relationship to the heuristic planning mechanism of *debugging an almost correct plan.* [Sussman 75]. This capability could be used in developing, learning and teaching patching expertise.

GOBs: an evolving structure for knowledge, mechanisms, and interaction

From the above description it should be clear that each GOB will be a relatively independent task with considerable local autonomy and ability to respond flexibly and dynamically to accomplishing its task. In a sense, each GOB represents the entire core LOOS system -and more - by providing an appropriate place to acquire, store, retrieve and reason about the application of knowledge of layout design in many forms. For instance, each GOB will store its own semantic filter appropriate to its layout task and set of constituent objects; this will save in a structured way at the right location an evaluative knowledge base that a designer develops and wants to store at the proper level of abstraction. Similarly, each GOB will have a slot for methods to achieve its task, of which the core systems[9] generative method will be only one option (albeit the prime one initially)[13]. Over time the system may learn to use particular methods in the context of particular layout subtasks. As knowledge and experience accumulate over design tasks in a domain one would expect a shift to take place from generate-and-test to more direct methods, such as instantiating pre-stored generalized solutions. One can envisage that skeleton or prototypical layouts (for instance, generalized context-invariant layouts satisfying internal relationships only) are stored as a result of previous design sessions and used as a basis for creating or completing a desired layout in response to a current context.

GOBs will serve as structure to capture layout design knowledge at the right levels of abstraction (evaluation, decomposition, appropriate method and control invocation, etc.) across

---

[12] **In layout design GOBs will have characteristics similar to prototypes in regard to storing generalized design solutions suitable for modification operations and patching.**

[13] **PosriMe alternative methods are selecting from a pre-enumerated set, refining a prototypical arrangement, modifying an almost correct arrangement, constraint-directed generation, or heuristic or interactive constructive methods.**

a number of design tasks within a domain, but remain modifiable and extendible within individual design sessions. Through GOBs an understanding of a class of design problems will accumulate in the form of design histories. In this way, the process of abstraction and generalization of knowledge from particular design tasks will no longer be just implicit in the practice of human designers alone, but will become explicit in the evolving architecture and knowledge bases of a CSDE. Because of their knowledge content and key place we believe that GOBs will also provide the opportunity to introduce and experiment with a variety of mechanisms for learning, the use of analogy, modification and patching operations, and other localized methods possibly related to design creativity or innovation in context.

The structure of GOBs, and the overall structure of a CSDE to accomplish the goals outlined, is itself a design problem, the formulation of which requires experimentation and testing in context to fully understand. The study of the required structure can be identified with the aim of understanding how to build design apprentices who accumulate experience and apply it to new tasks; this includes the compilation of expertise while retaining the flexibility to be creative.

We have shown that the LOOS system satisfies the basic requirements posed in our conjecture and begins to address the critical issues involved in the realization of an expanded computational model and experimental testbed for design and creativity - an evolving CSDE. It is obvious that the continual improvement of a CSDE will require careful consideration of all the issues involved with creating a well-structured interactive environment with graphical display, and high level languages. These will give a designer an understanding of and access to the developing content of knowledge structures and design representations without concern for their internal representation and implementation.

33 Computer Supported Design and Creativity in LOOS

In the LOOS approach to layout design the computer supplies the generative expertise and "horsepower" while the human provides the evaluative judgment to formulate the interpretation of the syntactically proposed layouts and to structure and guide the overall process. Through this balance of capabilities the overall architecture of the LOOS system effectively addresses some of the long-standing fundamental problems of supporting layout design. In this section we summarize how LOOS is an evolving CSDE that encompasses capabilities that incrementally and cumulatively enhance creativity in design. In doing so we answer, in large part, the following questions that should be asked of any computational design system that purports to enhance or replace in part the creative capacities of human designers.

1. How does the environment extend/improve the process of a human working alone?
2. How does a human complement and provide essential knowledge and judgment that an automated system alone would find difficult (impossible) ?
3. How does this integrated environment evolve over time? what does each agent gain? and how do their roles change, etc.?

To support a computational design model that answers the above questions, LOOS provides the following design support features:

- **The facility to expand the amount of the potential solution space searched for a given formulation of layout problem, by systematically exploring alternatives and the tradeoffs among them.**
- **The facility to expand the number of problem spaces explored for a given layout problem. It achieves this by supporting incremental reformulation through the addition and modification of constraints and constraint importance levels, and by supporting reformulation through alternative decompositions in a dynamic, flexible way.**
- **An independent syntactic solution generation capability. The strict separation of evaluation from formal syntactic generation enables a critical division of labor between machine and man, and also enables the approach to be applied to "generic layout tasks" across multiple disciplines.**
- **The incorporation of semantic concerns in a tester within a well-defined context for evaluation. Spatial relations provide an explicit structure for evaluation of partial designs in terms of a current full set of applicable criteria.**
- **The means for domain experts to make explicit their knowledge when confronted with a concrete solution in a familiar graphical display, as opposed to formulating their knowledge in general terms unrelated to a concrete case.**
- **Support for generative, evaluative and patching activities and assignment of appropriate roles to the agents, human and computer, for each of these. As mentioned earlier, it achieves this by dividing the labor of generation and evaluation between man and machine.**
- **The use of multiple representations and abstraction techniques and levels to mediate decisions and the processing of design goals to design specifications. Evaluations are enabled at different levels of abstraction.**
- **Combined top-down and bottom-up reasoning with limited dynamic, automatic transitions between these modes on demand. Control of the design activities is based on a limited commitment of resources mode of control. Control can also be managed interactively or automatically at all significant choice points.**
- **The computer performs constraint management and required bookkeeping as the design evolves, and can not only signal when constraints are violated, but is capable of keeping track of tradeoffs.**
- **Enables the use of subjective evaluation and stopping rules(tradeoffs). This ultimately results in the compilation of this knowledge into evolving customizable knowledge bases. The compilation of knowledge bases helps the designer in experimenting with multiple sets of evaluation rules.**
- **Provides for a continual update of the division of labor. For example, a change in the division of labor will occur with the evolution over time from (interactive) generate-and-test to more direct methods for proposing partial solutions, such as pre-stored solutions. The system acquires new design abstractions and evaluations from the interaction with the designer over large classes of problems. This would allow the system to propose design abstractions based on the specifications of the problem.**

Though we have not mentioned search explicitly, it is clear that LOOS incorporates search in a careful limited way in its design process. The generative component can be viewed as producing a well-defined search space of structurally defined alternatives. However, this space describes neither complete design alternatives nor the complete process. Here search, combined with a method for acquiring and applying evaluative expertise, underlies and supports a systematic process for exploring the overall problem space. Iterations of generate-evaluate-modify converge on the desired formulation of the design task and produce not only the enumeration of all possible solutions given that formulation, but an implicit theory of layout design for that domain in the form of the evaluation knowledge-base developed. Iterations over multiple problems in a domain can refine the knowledge base and theory produced.

Therefore, to simply classify the LOOS approach as a search-based design system, or as a knowledge-based system is a mistake. It has elements of both and they are integrated in such a way that the system more accurately reflects an *exploration-based model of design* as cited earlier in section 2.3. The implemented LOOS system and structured agenda for continuing research offer the singular advantage of an experimental environment in which to continue to test and refine our conjecture(s) regarding creativity with empirical results. In this approach we propose using the computational medium as a partially automated assistant to deal with real design tasks (layout), and intend to study the process of layout design and attempt to develop domain theories of layout design through the ordinary exercise of the system on design tasks.

4 Conclusion

In this paper we described an approach to the design of computational environments that can be used to study the conditions under which creativity can occur. We proposed a conjecture on creativity within an information processing theory of design. This conjecture, briefly stated, predicates the occurrence of creative design when a designer is provided the capability to explore an expanded problem formulation space or more of a given solution space for a task. We then characterized the realization of design environments to enhance creativity as a design task itself with evolving goals. Through LOOS, an architecture for layout design, we illustrated how the conditions of the conjecture could be supported in an evolving computer supported design environment We also described how this architecture allows for the development of domain theories of design. Finally, we discussed the significance of our approach as a foundation for design research and its relation to existing areas of research in creativity and design.

5 Acknowledgements

The authors are grateful to Ulrich Hemming and David Steir for their helpful comments on earlier drafts of this paper. Support for this work has come from the National Science Foundation through its funding of the EDRC.

References

[Addanki and Murty 87]
Addanki, S., and Murty, S.
Prompt* an innovative design tool.
In *Proceedings of the Sixth National Conference on Artificial Intelligence.* AAAI87, Morgan Kaufmann, Los Altos, CA, 1987.

[Adelson and Soloway 84]
Adelson, B., Soloway, E.
A cognitive model of software design.
Technical Report 342, Department of Computer Science, Yale University, 1984.

[Akin 86] Akin, Omer.
Psychology of Architectural Design.
Pion, England, 1986.

[Baker and Fenves 87]
Baker, N., Fenves, S.
A knowledge acquistion study of structural engineers performing design.
Technical Report EDRC 12-19-87, Engineering Design Research Center, Carnegie-Mellon University, 1987.

[Bundy 83] Bundy, A.
Nature of AI: reply to Ohlsson.
Artificial Intelligence and Simulation of Behaviour Quarterly (Summer):24-25,1983.

[Coyne 89] Coyne, Robert F.
Planning in design synthesis: abstraction-based LOOS (ABLOOS).
PhD. Proposal EDRC 48-14-89, Engineering Design Research Center, Carnegie-Mellon University, 1989.

[Doyle 84] Doyle, Jon.
Expert systems without computers or theory and trust in artificial intelligence.
AI Magazine 5(2):59 - 63, Summer, 1984.

[Doyle 88] Doyle, Jon.
On rationality and learning.
Technical Report CMU-CS-88-122, Department of Computer Science, Carnegie-Mellon University, 1988.

[Eastman 69] Eastman, C.
Cognitive processes and ill-defined problems: a case study from design.
In *Proceedings International Joint Conference on Artificial Intelligence 1969.* UCAI, August, 1969.

[Flemming 88] Hemming, Ulrich.
Rule-based systems in computer-aided architectural design.
In Rychener, Michael D. (editor), *Expert Systems for Engineering Design,* chapter 4, pages 93-112. ACADEMIC PRESS, INC., San Diego, CA 92101,1988.

[Hemming, et al 89]
Flemming, U., Coyne, R.F., Glavin, T., Hsi H., and Rychener, M.
A generative expert system for the design of building layouts (final report).
Technical Report EDRC 48-15-89, Engineering Design Research Center, Carnegie-Mellon University, 1989.

[Galle86] Galle,P.
Abstraction as a tool of automated floor-plan design.
Environment and Planning B: Planning and Design 13:21-46,1986.

[Genesereth 83] Michael R. Genesereth.
An overview of meta-level architecture.
In *Proceedings of the National Conference on Artificial Intelligence,* pages 119-124. AAAI, 1983.

[Gero et al. 88] Gero, John, Maher, Mary Lou and Zhang, W.
Chunking structural design knowledge as prototypes.
In Gero, John S. (editor), *Artificial Intelligence in Engineering Design,* pages 3-21. Computational Mechanics Publications, Southampton - Boston, 1988.

[God and Pirolli 89]
Goel, Vinod and Pirolli, Peter.
Motivating the notion of generic design within information processing theory: the design problem space.
Al Magazine 10(1): 18 - 36, Spring, 1989.

[Kalagnanam and Subrahmanian 89]
Kalagnanam, J., Subrahmanian, E.
Learning to diagnose by doing.
In *Proceedings of the International Joint Conference on Artificial Intelligence,IJCAI-89,* pages 540-545. Morgan Kaufmann, Los Altos, CA, 1989.

[Levesque 86] Levesque, Hector.
Making beleivers out of computers.
Journal of Artificial Inteligence 30(2):81 -108,1986.

[Liggett and Mitchell 81]
Liggett, Robin S. and Mitchell, William J.,
Optimal space planning in practice.
Computer Aided Design 13(5):277-288, September, 1981.

[Mittaletal.86] Mittal, San jay; Dym, Clive L.; and Morjaria, Mahesh.
PRIDE: An expert system for the design of paper handling systems.
Computer: 102-114, July, 1986.

[Mostow 85] Mostow, Jack.
Toward better models of the design process.
*Al Magazine 6(1):44-51,*1985.

[Newell and Simon 72]
Newell, A., Simon, H. A.
Human Problem Solving.
Prentice Hall, Englewood Cliffs, NJ, 1972.

[Ohlsson83] Ohlsson,S.
Mathematics, behaviour, and creativity: a reply to bundy.
Artificial Intelligence and Simulation ofBehaviour Quarterly (summer):25, 1983.

[Orelup 87] Mark F. Orelup; John R. Dixon; and Melvin K. Simmons.
Dominic II: more progress towards domain independent design by iterative redesign.
In *WAM,* pages 1-14. ASME, December, 1987.

[Rich 84] Rich, Elaine.
The gradual expansion of artificial intelligence.
IEEE Computer :4-12, May, 1984.

[Rychener 88] Rychener, Michael D.
Research in expert systems for engineering design.
Expert Systemsfor Engineering Design.
Academic Press, Inc., 1250 Sixth Avenue, San Diego, CA 92101,1988, pages 1-33, Chapter 1.

[Sharkey 85] Sharkey, N. and Brown, G.
Why artificial intelligence needs an empirical foundation.
Artificial Intelligence: Principles and Applications.
Chapman Hall, NewYork, 1985, pages 260-291.

[Simon 73] Simon, H. A.
The structure of ill structured problems.
Artificial Intelligence 4:181-201,1973.

[Simon 81] Simon, H. A.
Sciences of the Artificial(2nd edition).
MIT Press, Cambridge, MA, 1981.

[Smithers, et al. 89]
Smithers, T., Conkie A., Doheny J., Logan B., and Milligan K.
Design as intelligent behaviour: anAIin design research programme.
Technical Report DAI 426, Dept. of Artificial Intelligence, Univ. of Edinburgh, May, 1989.

[Stefik 81] Stefik, *Mark.*
Planning and meta-planning (Molgen: part 2).
Artificial Intelligence 16(2): 141-170,1981.

[Stefik etal. 83] Mark Stefik, Janice Aikins, Robert Balzer, John Benoit, Lawrence Birnbaum, Frederick Hayes-Roth, and Earl Sacerdoti.
Basic concepts for building expert systems.
In Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat (editors), *Building Expert Systems,* chapter 3, pages 59-86. Addison-Wesley Publishing Company, Inc., London, 1983.

[Steinberg 87] Steinberg, L.
Design as refinement plus constraint propagation: the VEXED experience.
In *Proceedings ofthe Sixth Annual Conference on Artificial Intelligence, AAAI-87,* pages 830 - 835. Morgan Kaufinann, Los Altos, CA, 1987.

[Sussman 75] Sussman, Gerald J.
A computer model of skill acquistion.
American Elsevier, New York, 1975.

[Takewalri 85] Toshiaki Takewaki; Taizo Miyachi; Susumu Kunifuji; and Koichi Furukawa.
An algebraic manipulation system using meta-level inference based on humanheuristics.
ICOT Technical Report TO-140, ICOT Research Center, Institute for New Generation Computer Technology, October, 1985.

[Ullman, et al. 88] UUman, D. G., Dietterich, T. G., Stauffer, L., A.
A model of the mechanical design process based on empirical data.
AIEDAM 2(1):33 - 52, 1988.

[Zhao and Maher 87]
Zhao,F.,Maher,M.
Using analogical reasoning to design buildings.
Technical Report EDRC 12-22-88, Engineering Design Research Center, Carnegie-Mellon University, 1987.