Computational Specification of Building Requirements in the Early Stages of Design

Conference topic area: CAD systems for early design phases

Omer Akin, Zeyno Aygen, Michael Cumming, Magd Donia, Rana Sen, and Ye Zhang
School of Architecture, Carnegie Mellon University, Pittsburgh, PA 15213, USA
Phone (412) 268-3594, Fax (412) 268-6129, E-mail: oa04@andrew.cmu.edu
and
James Garrett, Department of Civil and Environmental Engineering
Carnegie Mellon University, Pittsburgh, PA 15213, USA
Phone (412) 268-7813, E-mail: garrett@cmu.edu

ABSTRACT

We have been exploring computational techniques to help building designers to specify design requirements during the early stages of design. In the past, little has been accomplished in this area either in terms of innovative computational technologies or the improvement of design performance. The prospect of improving design productivity and creating a seamless process between requirements specification and formal design are our primary motivations. This research has been conducted as part of a larger project entitled SEED (Software Environment to Support Early Phases in Building Design). SEED features an open-ended modular architecture, where each module provides support for a design activity that takes place in early design stages. Each module is supported by a database to store and retrieve information, as well as a user interface to support the interaction with designers. The module described in this paper, SEED-Pro (the architectural programming module of SEED), is a working prototype for building design requirements specification. It can be used by other modules in SEED or by design systems in other domains, such as mechanical engineering, civil engineering, industrial design and electrical engineering. Our approach to SEED-Pro is divided into two phases: *core, and support* functionalities. The core functionalities operate in an interactive mode relying on a case-based approach to retrieve and adapt complex specification records to the problem at hand. The support functionalities include the case-base, the data-base, and the standards processing environment for building specification tasks. Our findings indicate that SEED-Pro: (1) is a tool that structures the unstructured domain of design requirements; (2) enables the integration of design requirements with the rest of the design process, (3) leads to the creation of complex case-bases and (4) enables the observation of their performance in the context of real world design problems.

1 ISSUES FOR A DESIGN REQUIREMENTS SUPPORT ENVIRONMENT

Computers have entered our lives primarily with the expectation if not the promise that they will make solving difficult problems easier. Whether it is text editing, data processing or simulating complex functions we expect computers to make everyday problems more manageable. This premise has lead to many important and profound developments in almost every facet of our lives. Most of these developments are aimed at producing computer systems that help solve well-defined problems (optimization, etc.) or provide general purpose support for a variety of well-

understood tasks (spreadsheets, etc.). There are very few systems, however, which assist users in structuring unstructured problems. This is a primary motivation for this work.

In the area of CAD, there are a plethora of tools that can assist with various aspects of the design process. The glaring deficiency in this space of tools is those that apply to the *early stages of design,* during which some of the most critical decisions about design are made within a context of incomplete and imprecise data. During this stage, the design problem is usually ill-defined [Reitman, 19641 and designers' ability to evaluate proposals is limited by the incompleteness of data. At the same time the decisions made during this stage are far more important in terms of their lasting impact on the performance characteristics of designs. To add to this difficulty, the amount of time spent on the preliminary design is disproportionately small.

Most design fields, in particular building design, result from a collaboration between multiple sub-areas of expertise. In building design, architects, mechanical engineers, civil engineers, landscape designers, and planners work together undertaking a variety of tasks: such as site design, building requirements specification, financial feasibility analysis, and design requirement analysis. Each of these participants bring to the table a specific skill and knowledge applicable to the building problem. Thus, any credible approach to the problem needs to be *cross-disciplinary,* bringing together a variety of knowledge domains and corresponding expertise.

We have been using some of the most advanced tools in software engineering to formalize and automate significant portions of the difficult and ill-defined task of specifying problem requirements in design. We have developed a computational environment for specifying as completely and accurately as possible the premises, parameters and dimensions of complex problems, thus enabling the computer based generation of a set of problem requirements that can seamlessly lead to computer based solution generation. In particular, strategies such as *case-based* design can be used to generate successful solution examples from pre-stored cases.

Finally, this paper builds on work reported earlier. SEED is a system which has been under development for the last four years at the School of Architecture and ICES (Institute for Complex Engineered Systems) [Akin, *et.al.,* 19951. Our purpose in this paper is to report the design requirements specification aspects of early design, particularly the portion that applies to SEED-Pro.

2. STATE-OF-THE-ART IN DESIGN SYSTEMS

The work described here represents the advancement of the state-of-the-art in computer software development, and the design process. The most significant impact is made in the following three categories: (1) generative support systems for early

stages of building design, (2) enhancement of productivity through computation, and (3) case-based support for design decision making.

The following section will first provide a brief survey of some software systems that define state-of-the-art in each of these categories. The work we are reporting here has a common motivation with some of these systems and attempts to build on the lessons learned from them. Following the survey we provide a discussion of the way this work is related to the current achievements in the field.

2.1 **Support systems for early stages of building design**

In the past, there have been many attempts to create design tools for the early stages of design. Unfortunately, most of these tools have made little, if any, impact on the generative aspects of design, even if this was part of the initial motivation for their creation.IBDE (Integrated Building Design Environment) represents an early effort to develop generative design modules representing several domains involved in building design, namely architecture, structural engineering, construction planning and cost estimation [Fenves et al., 1994]. Each design module is seen as an expert system. IBDE provides a test bed for the exploration of generation and communication issues in building design. Its usage is limited by the facts that it doesn't support the iterative design process, negotiation or conflict resolution between different domain designers; and it doesn't have a shared knowledge base which is necessary for having downstream modules dealing successfully with the decision made by upstream modules.

KNODES (Knowledge-based Design System) is a prototype building design framework intended to teach students the "complex relationships between spatial configuration and building performance," In particular it helps the exploration of the relationship between factors like natural lighting, energy consumption, structure analysis and the form and "fabric" of the building [Rutherford, 1994]. KNODES comprehensively incorporates different performance aspects related to building design, and, it tackles the complex relationships between a building geometry and its expected performance. However, it does not provide a generative design capability and it does not provide support for a more in-depth building design process beyond that of designing a building envelope.

ICADS (Intelligent Computer-Aided Design System) is developed to study the decision making process during the conceptual design stage [Pohl, *et.al.,* 1994] "The ICADS architecture consists of several knowledge-based agents that critique and propose changes to the emerging design. An additional agent is provided to propose resolution strategies for any conflicts that may occur between these agents. All agents access and modify the design description stored in a central blackboard system." [Fenves et al., 1994]. ICADS addresses complex problem solving and on-line access

to a large body of shared information; however, it does not support generative design, iterative design, and dialog or negotiation with users.

2.2 **Productivity enhancing design systems**

Productivity enhancement through the use of computer aided tools has been difficult to demonstrate in general, let alone in the area of building design. Here we review a variety of systems that have yielded measurable performance improvements, with the hope that lessons learned in these domains will transfer to our own.

The MULGA symbolic design system assists designers to layout VLSI chips [Boyer *et.al.*, 1989]. After a layout is manually generated, the system simulates the performance of the design and verifies the design. It has helped designers to achieve productivity by allowing a designer to design the whole chip without worrying about process design rules, facilitating cell and module reuse, and, simplifying the programs that implement functions such as compacting and extraction.

Ship Model, a computerized product model of a ship, is employed during the early design stage [Von Haartman, *et.al.*, 1994]. Besides being a tool that helps create and modify a design, this model provides a well defined primary source of design information, and a description of a ship that allows various analysis and calculations to be performed. The tool has been shown to enhance functional design improve basic design efficiency, and speed up detailed design.

The Design Associate (DA) is a tool used for designing racing yachts [Gelsey, 1992]. This system automatically carries out the iterative design process through the use of an expert system. After simulating a design, the expert system reasons and then modifies the design and automatically resubmits its modification for a new simulation. The yacht designed by DA, called "Stars & Stripes", won the 1987 America's Cup.

2.3 **Case-based design systems**

Case based technologies have been receiving growing attention particularly directed to the solution of routine design problems. There is a rapidly growing list of applications that catalog designs, and provide matching algorithms to find appropriate cases. In this picture, we also find, albeit with less frequency, systems that adapt cases to new situations. Below we provide a small sample of such systems.

CASECAD is a domain-independent design assistance system which integrates case-based reasoning and CAD techniques. The information held in cases has multiple representations. These include attribute-value pairs used as important design parameters, textual explanations used to provide clarification of certain design problems, and drawings used to illustrate the physical appearance of the design. Each of these features assist in the retrieval and adaptation of the case for a new problem.

Revision and reformulation of the design problem requirements can also be enabled in order to make the overall design process more flexible [Maher, 1997]. Similarly, in CADSYN each case is decomposed into subparts. By representing knowledge hierarchically, only relevant parts are retrieved. However the emphasis here is not on case retrieval but on case adaptation [Maher, 1997]

GENCAD is a case-based reasoning tool for structural design of tall buildings [Maher, 1997]. The unique contribution of GENCAD is that it uses a "general-purpose, knowledge-lean method based on genetic algorithms" for the task of case adaptation in order to avoid using a great deal of domain-specific knowledge and domain- and task-specific heuristics normally needed for such knowledge-based case adaptation.

2.4 **Expected act of this work on the state-of-the-art**

Our approach is intended make progress in all three categories of the literature surveyed above. First, it will advance the state-of-the-art in generative design systems by supporting (1) the iterative design process, (2) design negotiation and shared knowledge, and (3) a greater range of user-specified functional requirements, including, usage requirements, environmental requirements, and parametric design requirements.

Second, our approach is directed at enhancing productivity in building design. We expect that SEED-Pro will be instrumental in improving design productivity through the seamless integration of the design specification step with the later steps of the building design process. Automatic and interactive translation of specification data into design proposals will make the design process more efficient and eliminate upstream/downstream incompatibilities.

We also expect that case-based retrieval and adaptation of existing designs will also play a role in enhancing design productivity. In routine design, a case based approach is expected to improve efficiency of creating new designs. Through the reuse of data stored in cases, designers will be able to start their work at a more advanced stage of the design process and save valuable design time.

Finally, our approach expands the boundaries of collaborative computer system design in the building design domain. This incorporates various disciplines, such as information modeling, functional reasoning, optimization, process management, and interface design.

3. SOFTWARE ENVIRONMENT TO SUPPORT EARLY PHASES IN BUILDING DESIGN (SEED)

3.1 **The Overall Environment**

In 1993 researchers at the Engineering Research Design Center (EDRC) and the School of Architecture at Carnegie Mellon University started the SEED project, which provides a software environment to support the early phases of building design. The overall goal of SEED is to provide support for the preliminary design of buildings in all aspects that can gain from computational support. This includes using computers for the "rapid generation of computable design representations describing conceptual design alternatives and variants of such alternatives with a sufficient level of detail that enables sophisticated evaluation tools to receive all of the needed input data from the representation" [Flemming *et.al.*, 1995b]. SEED encourages the exploratory mode of designing by making it easy for designers to iterate through design versions and pursue conceptual alternatives in parallel.

SEED features an open-ended modular architecture, where each module provides support for a design activity that takes place in the early design stages. Each module consists of five main components: input, specification, generation, evaluation and output. These are supported by a database to store and retrieve information, as well as a user interface to support the interaction with designers [Figure 1].

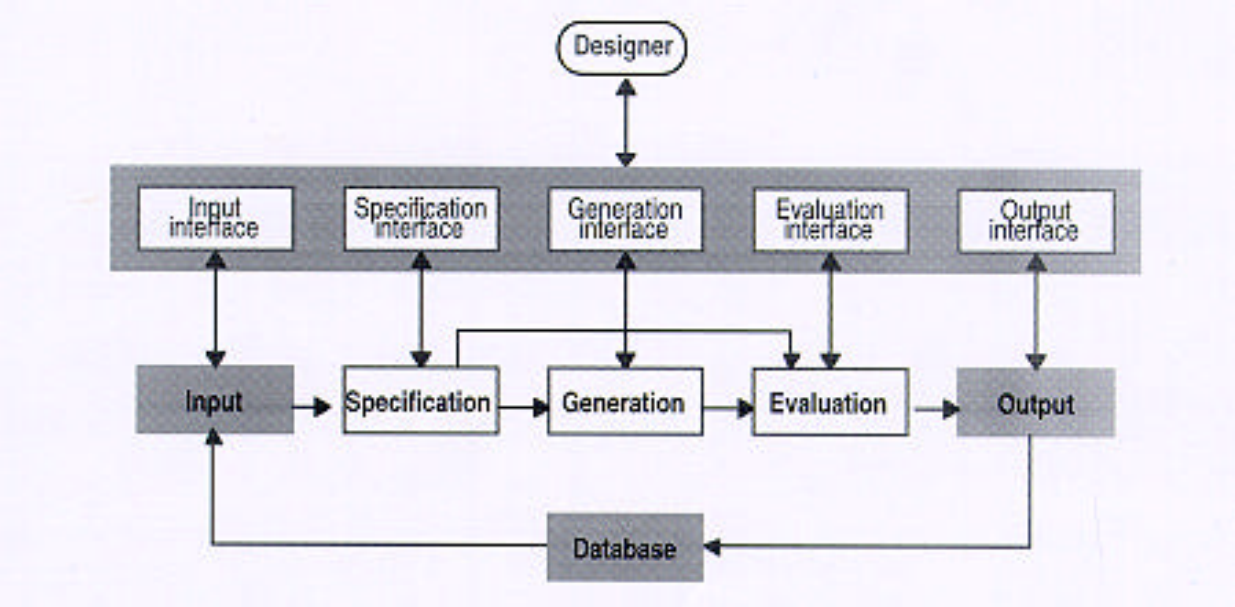

Figure 1: **Generic Architecture of a SEED module** [Flemming, *et.al.*, 1995b].

3.2 **Architectural Programming in SEED**

To support design generation, a well-defined set of explicit requirements is needed. The prototypical version of SEED-Pro, the building requirements specification module of SEED, is designed with the intention to support the modeling and generation of design requirements in a form that is usable by other modules of SEED. It has the following objectives [Akin *et.al*, 1995]:

- Provide means of storing and handling all aspects of the requirements specification information including site characteristics, codes, client preferences, and different performance criteria and requirements.

- Ensure the use of criteria established during the building requirements specification phase as a basis of design.

- Enable the integration of building requirements specification and architectural design as a seamless process.

- Maintain a database of reasons employed in making requirement specification decisions, and improving the computability of requirement specification information by allowing non-numerical types of reasoning.

- Achieve a flexible way of interaction that does not tie the user to a specific requirements specification model.

- Enable the use of past programs and the requirements specifications in future projects.

Through the sharing of domain object classes, SEED-Pro aims to provide a seamless interaction with all of the other modules of SEED and share data across these modules. SEED-Pro positions itself as a good candidate for maintaining a robust record of design requirements, criteria, and constraints to be used persistently in SEED.

Currently, SEED-Pro is implemented in C++ using the ET++ application framework [Weinand, *etal.*, 1994] and includes the core functionalities of building requirements specification. This prototype is primarily an interactive system relying on user input of massive amounts of data.

4. CORE FUNCTIONALITIES OF SEED-PRO

SEED-Pro aims to provide a seamless interaction with all of the other modules of SEED [Flemming, *et.al.*, 1995a]. It shares data as well as methods of data manipulation with other modules. By providing the outputs that the other SEED modules require as input and through the shared domain object classes and libraries in SEED, SEED-Pro complements the basic steps of early design: architectural problem specification, two dimensional design and three dimensional configuration design.

Through the sharing of domain object classes which represent entities like *functional unit (FU)*, *design unit* (DU) and *specification unit* (SU) SEED-Pro enables translations between organizational, functional and spatial concepts.

- A DU is a spatial or physical part of a building with an identifiable spatial boundary (e.g. a living room). A DU can contain other DUs such as other rooms or furniture.

- A FU represents a combination of functions to be satisfied by a single DU and also serves as the repository of requirements for that DU. The requirements could be in the form of constraints and criteria regarding the shape, size, etc. of the DU. A FU can contain other FUs.

- A SU collects the design intentions and criteria to be satisfied by one or more FUs. A SU can contain other SUs.

SEED-Pro converts the SUs or SU-hierarchies given to it by the user into FU-hierarchies. These are sets of FUs organized hierarchically through constituent relationships. There are also other relationships specified within FUs that indicate design constraints, technologies and various inheritance relationships [Akin, *et.al.*, 1995]. The FU-hierarchy serves as the output of SEED-Pro and the input to the other two modules of SEED. There are several different methods of FU-hierarchy generation used in SEED-Pro: (1) building from scratch (2) adapting an AP from the case library, and (3) automatically generating from specifications of clients.

The problem decomposition functionality is another tool that has been developed in SEED. This is intended to make the coupling of the output of SEED-Pro and the input of SEED-Layout more successful. The spatial system synthesis is not the only potential application in this category. Other design issues, such as thermal, acoustical and structural systems, also provide appropriate categories for problem decomposition.

4.1 **Specification of the architectural problem in SEED-Pro**

Design specifications are defined in SEED-Pro as a collection of design intentions and criteria to be maintained by the architectural design. This is the front end to SEED-Pro and provides it with a rich record of specifications categories. To date, we have implemented only the building specification category. This specification component allows for multiple AP alternatives that satisfy the same set of specifications, as well as facilitating case matching and retrieval. It also captures the rationale and the intentions of the design criteria, thus providing a basis for functional reasoning by being able to trace a design decision to the specifications that initially triggered it. This feature is being developed further to enable the automatic translation between SUs and FUs (Section 5.3).

A SU is the basic building block for representing specifications in SEED-Pro. It embraces an object-oriented representation that consists of the SU and its component *objects.* A SU object represents an organizational entity in the building which can correspond either to a physical space, a room, or to an abstract organizational concept. SUs can be recursively aggregated to form the organizational and functional hierarchies of the building. A SU can be associated with a number of component specifications, each of which describes desired performance requirements and design criteria. The list of components is open ended and allows the system developer to add additional component types as needed.

4.2 **Generation of the solution in SEED-Pro**

The purpose of the generation component of SEED-Pro is to develop the FU-hierarchy corresponding to the SU-hierarchy and in conformance with related codes and standards. Three different generation methods are considered: developing a FU-hierarchy from scratch, case matching and retrieval, and automatic translation from a SU-hierarchy to a FU-hierarchy.

Once the SU-hierarchy is built the designer can invoke the FU-Iibrary and start the generation process. The designer browses the library and selects the FU corresponding to the SUs. The connections between each SU and FU must be explicitly made by the designer. The system assists the designer in building the FU-hierarchy by providing sophisticated editing capabilities.

4.2.1 Case-based generation of an architectural program

The SU-hierarchy is the primary criteria for matching and retrieving cases from the case library. A general description of the indexing, retrieval and adaptation of cases in SEED modules is presented elsewhere [Flemming *et. al.,* 1994]. Once a case has been retrieved, the designer will be able to make modifications to the AP and adapt it to the current problem. If, however, multiple cases are retrieved then the designer has to make an appropriate selection first. In this mode, the AP is generated interactively

with the designer; the system plays a more active role in the process of retrieving matched cases.

4.2.2 Generation of an architectural program by automatic translation
SEED-Pro has the capability to translate some values from the SU-hierarchy into one or more FU-hierarchies according one of many mapping techniques being developed for this purpose. The output of some mappings take forms other than just FU-hierarchies that can be fed into other CAD systems or simulation tools attached to SEED as external modules. Such direct mappings are nontrivial in nature and further research is being conducted in this area. In the simplest instance, the mapping technique retrieves relevant values from the SU space and computes values in corresponding FU space(s).

4.3 **Evaluation of the solution in SEED-Pro**

SEED-Pro cycles through generation and evaluation steps until the designer is satisfied with the output. Although a generated AP is intended to be an accurate representation of the requirements derived from the SUs, there are several aspects of the AP the accuracy of which cannot be guaranteed a priori. Consequently, evaluation, such as checking for conformance to minimum requirements of codes and standards, is needed after the generation cycle is completed. None of the evaluation features are fully implemented. Currently work is being done in the following areas:

- *Compliance checking* After a FU is generated or the FU-hierarchy is built, the designer will be able to request the Standards Support Environment (SSE) to check the FU or the FU-hierarchy for conformance against all applicable standards and codes.

- *Compliant generation* This functionality will help the designer to retrieve applicable provisions or constraints or both and incorporate them into the FU. This will be helpful when an AP for a new building type is being generated and the designer is not familiar with the codes or standards.

4.4 **Output of SEED-Pro**

The output of SEED-Pro includes an FU-hierarchy that forms the architectural program along with other alternative hierarchies. The FU-hierarchy is the input to the functional decomposition component (FDC). FDC generates alternative decompositions of the FU-hierarchy as the input to SEED-Layout. Some of the other outputs of SEED-Pro include:

- *Output to case base* From among the several APs that the designer develops, those that are considered to be good prototypical examples of a given building type can be saved for future reference and use towards generation of new APs.

- *Output to data base* APs developed for the generation of designs by the SEED-Layout and SEED-Config modules are saved in the data base. Other versions of AP expected to be used in interim client reviews and other project related work are also saved in the data base as phases of work or as alternatives [Akin, *et.al.*, 1995].

- *Pre-formatted reports* Printed reports describing the AP to accompany inventory of spaces, feasibility studies and other facilities management related tasks.

4.5 **Problem Decomposition**

The FDC helps translate the output of SEED-Pro into the input for SEED-Layout. More specifically, it generates alternative decompositions of the FU-hierarchy developed by the generation component of SEED-Pro. This clusters FUs according to different types of relationships like area, adjacency, thermal, daylight and acoustic requirements.

FDC is designed to generate alternative groupings of FUs based on a variety of relationships. Under the proximity relationship, a certain distance is required between any two FUs. The FUs with closer proximity requirements are grouped together while the FUs with remote proximity requirements are separated into different groupings. In the thermal category rooms with similar heating/cooling requirements are kept together. Acoustical criteria are applied to minimize noise/quiet requirements of functions. FDC takes as a constraint a single requirement category at a time in order to generate alternative decompositions. In section 5.4 this core functionality of SEED-Pro is further elaborated.

5. ONGOING WORK

The work currently in progress constitutes the kernel of SEED-Pro's automated and semi-automated processing of requirements data towards becoming a server for a number of design applications, such as layout generation, cost estimation, feasibility analysis, and so on. This work is based on the core functionalities of SEED-Pro described above: specification, generation, evaluation, output and functional decomposition. The categories included in this section are based on the elaboration of these functionalities to achieve an enhanced level of functionality, such as modeling *specification primitives, design requirements, constraint generation, stacking and zoning, and process management.*

5.1 **Specification Primitives**

The complexity of modeling design requirements for buildings arises mainly from the absence of a formal way to define such requirements. Firms specializing in building

requirements specification, have adopted or created formal models for defining design requirements. In general, the ways in which these models define design requirements are often grouped into two main styles [Kumlin, 1995]. The first is the *prescriptive style* through which design requirements are specified in terms of properties of materials and building systems based on solution standards or on the designer's experience. The second is the *performance* style by which design requirements are specified as performance criteria, such as required air temperature, illuminance, activities to be performed. Adopting one style over the other is most likely to result in an inadequate model to define design requirements. Different aspects of design requirements are better specified in given styles, because it is often hard to determine whether certain design requirements are prescriptive or performance based. For example, defining the R-value of a wall can be considered as a performance criterion or alternatively as a prescriptive material property.

5.2 **Modeling Design Requirements**

Modeling design requirements could be achieved using three different approaches. The first approach is to model design requirements after an industry standard, such as those supported by the American Institute of Architects (AIA) or building materials suppliers' specifications (i.e., the Sweet's catalog). Such an approach is relatively easy to accomplish and is likely to be used by people and firms that adopt the standard on which the model is based. However, this results in a model that is restricted to a single standard and inflexible, thus failing to accommodate the evolving nature of information categories in the field.

The second approach is to provide a model that encompasses all possible industry standards-a union of all the existing standards and classifications. In this case, the model is not restricted to a single standard and would cater to a wider group of practitioners; however, this approach would prove to be impractical due to contradictions that exist between standards, and the very large number of conventions used in the field.

A third approach is to provide a flexible framework to model design requirements according to a given standard or classification. This framework is intended to be adaptable in a way similar to application frameworks, such as ET++ [Weinand, *et.al*., 1994]. It provides an overall organizing framework to model design requirements using adaptable components to accommodate different ways of describing design requirements. This approach is more flexible than the previous two approaches and facilitates the addition of new specifications categories.

Our approach in SEED-Pro is represented by the third approach. This poses certain computational problems that have to be addressed at the functionality and interface levels. One of these problems is the creation of new specification categories at run-time. It is well known that strictly linked languages, such as C++ do not allow the

introduction of new types or classes at run-time. Our solution to this problem consists of the development of a modeling language that permits a very large set of specification types generated through the compilation of primitive modeling elements. In addition to these, the problem of translating specifications created according to a certain set of categories to another need to be addressed. These issues include:

- *An overall organizing concept, such as the functional specification (FS) hierarchy, which is for structuring the design description according to certain criteria:* function, spatial composition, organizational hierarchy. This organizing concept represents the notion of composition in an object-oriented model. In that sense, a hospital design can be functionally structured as a composition of several floors, each containing zones and rooms, while an organizational description may lead to an alternative based on departments: pediatrics, cardiology, etc.

- *Relationships that can be expressed between components of the organizing concept that are independent of the way the components are structured:* examples of such relations are adjacency, minimum distance, and accessibility requirements for building designs.

- *A set of specification primitives, composed mainly of attribute types, used to define specification categories:* these are grouped to create sets of specifications, such as for building design specifications or circuit board designs.

- *One or more generation mechanisms that 'know' how to extract information from the model that uses such a set in order to create reports and outputs for other design systems can be available:* some of these generation mechanisms create a spatial description of the building that feeds into a two-dimensional layout design system such as SEED-Layout [Flemming *et.al.,* 1995b]. Other mechanisms can create input for mechanical or structural design or energy simulation systems, while others can generate reports.

There are issues to be addressed in order for such a framework to be fully implemented. Among these issues are the form of the overall organizational concept, the ways relationships are modeled, and creating the set of specifications primitives needed to model the different categories of specifications, in addition to user interface issues.

5.3 **Constraint generation**

The current version of SEED-Pro has two main schema: functional specifications (FSs) and requirements specifications (RSs) both of which can be represented as individual objects or as hierarchic structures. FSs structure the preliminary design description according to certain criteria, such as operational, functional or

organizational structures relevant to a facility. Other relationships could also be expressed in such a structure, such as adjacencies. The next step is the representation of spatial relationships between building components based on the FS-hierarchy.

The RSs encode such spatial constraints as maximum and minimum areas and dimensions besides other building performance requirements, such as thermal, acoustic, and lighting. The focus of our research has been specifically on the generation of spatial RSs from representations of FSs.

Design descriptions include: functional specification primitives which have spatial attributes and user activity models. We are building a functional reasoning engine which will take as input a particular activity model and develop spatially based relationships such as adjacency or distances at different levels of abstraction. The knowledge base for different building types represented in these activity models will grow as a consequence of using SEED-Pro. This knowledge base then will be used to derive more sophisticated heuristics to supplement the reasoning engine.

The other generation mechanism that is being developed is more straightforward: it retrieves the spatial constraints stored in FSs, like equipment and furniture dimensions or adjacency relationships and derives the corresponding spatial constraints. The process thus involves the development of a representation for different kinds of generation mechanisms called "technologies" [Fenves *et al,* 1995] and the generation techniques themselves.

The output is in the form of RS-hierarchies, each representing a particular spatial solution. The constraints have their own domain model and communicate with other modules, like SEED-Layout, through a shared representation [Snyder, *et.al.,* 1995].

5.4 **Stacking and zoning**

The processes of stacking and zoning are important decision-making steps in the early stages of building design. Using these processes, designers group spaces by their functional requirements and fit them into a predefined building structure, e.g., floors. This task is both time-consuming and complex.

The stacking and zoning module in SEED-Pro (called the FDC) automates these processes. It takes as input the functional requirements specified and generated in SEED-Pro and outputs groupings of spaces by *floor* and *zone.* It works by partitioning the spaces within a building into different floors and zones according to the strength of their relations with each other.

State-of-the-art stacking and blocking programs have some conimon deficiencies which are resolved in FDC. These include:

- *Limited user interaction.* FDC allows for user interactivity and for users to evaluate solutions generated by the system and to modify those solutions. Users are also able to modify the initial problem statement, and pre-assign spaces to floors, or zones.

- *Lack of automatic, support.* Those that do provide an inadequate solution and have a time-consuming algorithm. FDC uses an adapted state-of-the-art graph-partitioning algorithm to provide a solution in linear time.

- *Lack of input for different types of functional requirements.* FDC supports affinity, thermal, and acoustical requirements and is set up to handle additional functional requirements in the future.

- *Weak user interfaces.* In FDC the interface uses direct manipulation, multiple representations, and transparent-to-user system operations.

6. FUTURE WORK

While work on the current design features are nearing completion we are starting work on several other important functionalities of SEED-Pro. These include process management tools, automated case base support and standards processing.

6.1 **Process management tool**

This is a tool to document, design and analyze process information during the requirements specification phase of design. Process is defined as any time or other resource consuming activity which follows some sequence of definable steps. The motivation for such a tool is the fact that design specifications often ignore the processes involved. These processes should comprise a major portion of design specifications since in practice they can be seen to have a major effect on the final form and use of a product.

The processes which will be addressed in the context of SEED-Pro are: activities involved in the design process itself including layout and 3D configuration design; the construction activities required to construct an building design, and the activities of end-users housed within a completed design.

The task of completing such a process modeling tool includes: (1) a thorough literature search of available computable models of process involving both the design and manufacturing communities; (2) a survey of available applications which are currently used in the design of design processes, including project, construction and facilities management software; (3) a set of requirements which no available applications properly address in the context of design requirements specification; and (4) an adaptation of this process model to the design processes.

This design process model will be informed by current design process research which may or may not target computable forms of design. The chosen representation will be aligned to available standards such as the NIST (National Institute of Standards and Testing) Unified Specification Language proposal [Schlenoff, *et.al.*, 1996]. The display and manipulation of design processes should be allowable in several views, one being graphical. It will be possible to store this process information in SEED data representations, with particular emphasis on its use in SEED-Pro. It will be integrated with product and functional requirements already stored in the SEED-Pro data model.

6.2 **Case-base support**

Case-base capabilities aim at aiding the system user by providing access to a large memory of past design specifications. In this way, the user is provided with an initial design specification that is immediately available for modifications. The suggested case-base engine will provide systematic support for: the storage and indexing of past design specifications generated by the system, their retrieval at a similar problem context, and their reuse through adaptation to generate new functional specifications.

The indexing of specification cases will employ a stand-alone classification engine which will allow the user to attach multiple classifications to cases. The flexibility in classification is assured by allowing the users to create and modify their own classification vocabulary without having to alter the data model used for case representation. The AJPI for case-base and classification knowledge-base transactions is part of a software system currently available to us -- namely SPROUT, an Object Modeling Language based schema representation system (Snyder, *et.al.*, 1995).

6.3 **Standards processor**

When the building requirement specifications is being developed, the code provisions that apply to this building, such as provisions from the local building code, plumbing code or electrical code, must be determined and incorporated into the requirements specification. This problem is compounded by the fact that there is a multiplicity of model codes that exist and that each jurisdiction may adopts and modifies these model codes as their local codes. Dealing with codes is thus an important and difficult aspect of building requirements specification.

As part of the previously described SEED project, [Kiliççöte *et.al.*, 1995] developed a .formal language for modeling, in computer-usable form, the content of building codes, and a processor for applying this formal description of the code to the description of a building. This language is able to capture the complex, higher-order logic that is often found in these codes, such as provisions that define exceptions to other provisions or the applicability of other provisions. The code processor is able to identify, for a given building description, the applicable provisions, identify if enough

infon-nation is available to evaluate these provisions, develop a plan to evaluate them, evaluate them, and report on the conformance of the building to these code provisions. To date, this processor has been used by the SEED system as a post processor, where the building description is checked for conformance to an identified code.

We intend to expand the standard processor in functionality so as to assist SEED-Pro in identifying the applicable portions of a building code and incorporating those code provisions into the requirements specification. The existing standards processor first identifies the applicable provisions and then proceeds to evaluate the provisions, if possible. Early in the design process, the provisions will not be able to be evaluated because design has not yet occurred. However, if the standards processor is able to identify provisions that apply, these provisions can be transformed into code-based constraints that can be included in the requirements specification. Transforming these applicable provisions from their modeled form into a constraint that is usable by SEED-Pro will require that a code application knowledge-base be added to the standards processor. We will have to make this approach practical for the current code context, where each jurisdiction has its own set of local building codes that must be incorporated into a requirements specification. Hence, the code application knowledge-base will have to be applicable for all codes modeled using the code modeling language-

7. CONCLUSIONS

SEED-Pro is, to the best of our knowledge, the first generative tool to assist in architectural programming and link it seamlessly to tools that help in early design generation, like building layout design or standards compliance checking. This in it self is significant. We fully expect it to become both a seminal piece of software leading to even more substantial activity in software development and a benchmark for other approaches aimed at the same problem.

Currently, we have not tested SEED-Pro in the field. This phase of our work is schedule for the 1998-9 academic year. It is our expectation that it will improve user productivity and program quality for architectural programmers. The case based approach we have adopted and the standards processing features extensions of SEED-Pro warrant recognition and further attention, in an area of computer aided design that has been neglected by researchers for a long time.

8. REFERENCES

[Akin, *et.al.*, 1995]
Akin, Ö., R. Sen, M. Donia, & Y. Zhang. (1995). SEED-Pro: Computer Assisted Architectural Programming in SEED. *Journal of Architectural Engineering, Vol* I No, 4, December 1995.

[Boyer, *et.al.,* 1989]
Boyer, David G., and Robert R. Cordell (1989). Symbolic Layout for Rapid Full-Custom Prototyping of High-Speed Telecommunications Chips, *Proceedings of the Twenty-Second Annual Hawaii International Conference on Systems Sciences: Architecture Track, Vol* 1, p92-101.

[Fenves, *et.al*., 1994]
Fenves, Steven, Ulrich Flemming, Chris Hendrickson, Mary Lou Maher, Richard Quadrel, Michael Terk, and Rob Woodbury (1994). *Concurrent Computer-Integrated building Design.,* PTR Prentice Hall, Englewood Cliffs. NJ.

[Fenves, *et.al.,* 1995]
Fenves, S. J., Rivard, H., Gomez, N., and Chiou, S. C. (1995). "Conceptual Structural Design in SEED." *Journal of Architectural Engineering, Vol.* 1, No. 4, December 1995. American Society of Civil Engineers.

[Flemming, 1994]
Flemming, U. (1994). "Case-Based Design in the SEED System." in *Knowledge-Based Computer-Aided Architectural Design,* Carrara, G., and Kalay, Y. E., eds. Dordrecht, the Netherlands: Elsevier Science B.V.

[Flemming, *et.al*., 1995a]
Flemming, U., and Chien, S. F. (1995) "Schematic Layout Design in the SEED Environment." *Journal of architectural Engineering, Vol.* 1, No. 4, December 1995, American Society of Civil Engineers.

[Flemming, *et.al*., 1995b]
Flemming, U. and Woodbury, R. (1995). "Software Environment to Support the Early Phases of Building Design (SEED): Overview" *Journal of Architectural Engineering,* 1(4), New York: American Society of Civil Engineers. pp. 162-169.

[Gelsey, 1992]
Gelsey, Andrew (1992). "Modeling and Simulation for Automated Yacht Design," Rutgers University Technical Report, CAP-TR- 16.

[Kiliççöte, *et.al., 1995*]
Kiliççöte, H., B. Choi and J. H. Garrett, Jr. (1995) "Providing formal support for standards usage in SEED," *ASCE Journal of architectural Engineering,* Special Edition on the Carnegie Mellon University SEED Project, Vol. 1, No. 4, pp. 187-194.

[Kumlin, 1995]
Kumlin, R., (1995). *Architectural Programming: Creative Techniques for Design Professionals.* McGraw Hill.

[Maher, *et.al.*, 1997]
Maher, Mary Lou, Andres Gomez de Silva Garza, Dongmei Zhang, Bala Balachandran, and Kate Bridge (1997). "Case-Based Reasoning in Design," retrieved November 15, 1997 from the World Wide Web: http://www.arch.su.edu.au/-andes/cbd.html.

[Pohl, *et.al.*, 1994]
Pohl, Jens, and Leonard Myers (1994). A Distributed Cooperative Model for Architectural Design, in Gianfranco Carrara and Yehuda E. Kalay (ed.), *Knowledge based Computer-Aided Architectural Design,* Elsevier Science B.V., Amsterdam, The Netherlands, p205-242.

[Reitman, 1964]
Reitman, W. R. (1964) "Heuristic decision procedures, open constraints and structure of ill-defined problems," in *Human Judgements and Optimality*, eds. M. W. Shelly and G.L. Bryan, John Wiley, New York, pp. 282-315.

[Rutherford, *et.al*., 1994]
Rutherford, James H., and Thomas W. Maver (1994). Knowledge-based Design Support, in Gianfranco Carrara and Yehuda E. Kalay (ed.), *Knowledge-based Computer-Aided Architectural Design,* Elsevier Science B.V., Amsterdam, The Netherlands, p243-267.

[Schlenoff *etal.*, 1996]
Schlenoff, C.,. Knutilla, A., Ray, S. 1996. *Unified Process Specification Language: Requirements for Modeling Process.* NISTIR 59 10. Washington DC: National Institute of Standards and Technology.

[Snyder *et.al.*, 1995]
Snyder, J., Aygen, Z., Flemming, U., and Tsai, J. 1995. SPROUT: "A Modeling Language for SEED." In *Journal of Architectural Engineering, Vol.* 1, No. 4, December 1995. American Society of Civil Engineers.

[Von Haartman, *et.al*., 1994]
Von Haartman, J., I. Kuutti, and C. Schaumen (1994). Improved Design Productivity with a Product Model for Initial Ship Design, *Proceedings of the Eighth International Conference on Computer in Automation of Shipyard Operations and Ship Design,* vol 2, 5-9 Sept., Bremen, Germany.

[Weinand, *et.al*., 1994]
Weinand, A. and E. Gamma, (1994). "ET++ - a Portable, Homogenous Class Library and Application Framework", in *Proceedings of the UBILAB '94 Conference, Zurich.*