Using Experience To Plan The Synthesis of New Designs

by

M. L Maher, F. Zhao *J*

EDRC-12-03-86

September 1986

Using Experience To Plan The Synthesis Of New Designs

M.L MaherandF. Zhao
Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, PA

ABSTRACT

Engineering design is a creative process in which the experience and knowledge of the designer and the design specifications are combined. Facing a new design task, an experienced designer will easily recall a similar case which he or she met before and make an appropriate adaptation of the previous design solution to fit the new situation. This paper proposes a methodology for automating the use of previous design situations and their solutions to plan the synthesis of new designs.

Using Experience To Plan The Synthesis Of New Designs

M.L. Maher and F. Zhao
Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, PA

Engineering design is a creative process in which the experience and knowledge of the designer and the design specifications are combined. Facing a new design task, an experienced designer will easily recall a similar case which he or she met before and make an appropriate adaptation of the previous design solution to fit the new situation. A less experienced designer may go through much trial and error before a suitable design solution is achieved. The difference between the two is experience. This paper proposes a methodology for using previous design situations and their solutions to plan the synthesis of new designs. The area of application is the preliminary structural design of buildings.

A popular way of using experience to solve new problems is to develop an expert system that contains an appropriate representation of the relevant experience. Though there are several expert systems in practical use, the development of expert systems is still at a primary stage. Most of the successful expert systems, such as MYCIN, DIPMETER, etc., are diagnosis or classification systems. Design problems are more difficult to solve using existing expert system techniques because they require complex representations and strategies. However, knowledge based expert system techniques have been used to develop design assistants for the preliminary structural design process. The techniques employed have relied on the definition of a knowledge base containing objects representing structural subsystems and rules representing the appropriate way to combine the objects and compare the solutions. This paper describes a methodology for expanding the knowledge that is used during the structural design process to include experience in the form j previous building design situations and solutions. Specifically, this knowledge is used to plan :>--} synthesis of alternative structural systems for a given building design problem.

This paper is organized into four sections. The first section introduces structural design j-d highlights the phases relevant to this paper. The second section provides a brief overview f knowledge based approaches to structural design and relevant strategies. The third sec: :n describes a methodology and prototype implementation for using design experience directly. F ' : . Section 4 provides a summary of the work and some preliminary conclusions.

1. Structural Design

The structural design process starts with the definition of a need to transmit loads in space to a support or foundation, subject to constraints on cost, geometry, and other criteria. The final product of the design process is the detailed specification of a structural configuration capable of transmitting these loads with the appropriate levels of safety and serviceability. The design process may be viewed as a sequence of three stages, as illustrated in Figure 1-1 and described below.

(1) PRELIMINARY DESIGN (conceptual design) involves the synthesis of potential structural systems satisfying a few key constraints, and the selection of one, or at most a few, systems to be pursued further. Synthesis requires a knowledge of structural subsystems and their appropriateness for different situations.

(2) ANALYSIS is the process of modeling the selected structural system and determining its response to external effects. This process involves transforming a physical structure to a mathematical model, analyzing the model, and interpreting the results of the analysis in terms of the actual physical structure.

(3) DETAILED DESIGN is the selection and proportioning of the structural components such that all applicable constraints are satisfied.

There may be significant deviations between the properties of components assumed at the analysis stage and those determined at the detailed design stage, which would necessitate a reanalysis. Other major and minor cycles of redesign may also occur. The process continues until a satisfactory (or optimal) design is obtained. The *conceptualize-analyze-detail* cycle is typical of many design paradigms.

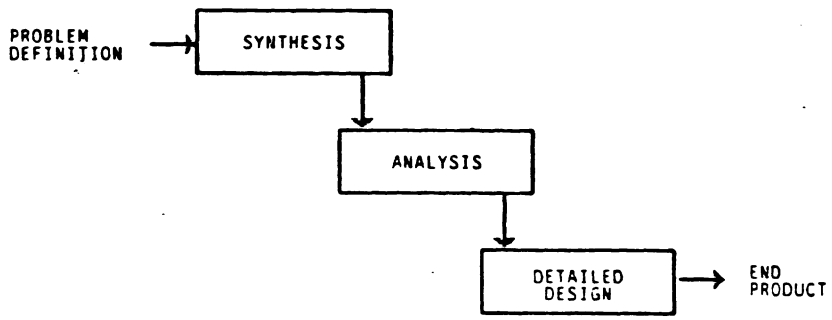


Figure 1-1: Engineering Design Phases

1.1. Synthesis of Alternative Designs

Many of the conceptual aspects of structural design are embodied in the preliminary design phase. At this point, the only information available to the designer are the specifications of the end product. It is during this part of the design process that the creativity and experience of an engineer are needed. The increasing complexity of engineering design problems has made synthesis a very difficult process, even to an experienced designer, if not approached in a structured and organized fashion.

There is no standard approach to the synthesis process suitable for ail design problems. One approach is to decompose the design problem into the design of independent subsystems. The nature of these subsystems will depend on the nature of the problem at hand. In a similar manner, each subsystem is further decomposed into major components. Alternative design solutions can be synthesized by considering all possible combinations of the various subsystems that result from combinations of lower level components. This hierarchical approach to the synthesis of a solution enables the designer to consider an exhaustive set of possibilities based on the manner in which the subsystems and the lower level components are defined. These definitions will depend on the nature of the particular problem as well as the engineer performing the design. A particular problem may justify the decomposition of a problem from an abstract level down to a set of detailed subsystems. Another design may be better approached by considering the details first and building up to more general systems. A correct selection of this sequence can increase the efficiency of the synthesis process.

A key consideration in the synthesis of design alternatives is the identification and satisfaction of constraints at the various levels of abstraction. These constraints control the qualification of vancus components of the design as well as the feasibility of combinations of such components. Typica '/ these constraints are based on the experience of the designer, but may also include dent s specifications and regiortal restrictions. The formal identification of these constraints is a diff CL> t process that is never completed. As more design experience is acquired, the current set of constraints may need to be modified and expanded.

1.2. Design Vocabulary and Experience

The synthesis of alternative designs can be considered as the manipulation of a design vocabo *i-r* using the designer's experience. The design vocabulary for a particular class of problems inciu <-s the linguistic description of the levels of abstraction and the subsystems and components :^e designer knows about. For example, the design vocabulary for a structural system design prct: ···^ may include rigid frames, braced frames, and shear walls. For any given problem, a designer will c^y

consider a subset of his entire design vocabulary; this subset represents the vocabulary that is appropriate for the given circumstances. An experienced engineer not only has a larger design vocabulary, but also is very good at determining the most appropriate subset for a given situation. This kind of experience is difficult to capture in an expert system that contains heuristics in rule form. The next section discusses some relevant attempts at capturing this experience and introduces a strategy for representing and using design experience directly.

2. Expert Systems

Expert systems, as an application of artificial intelligence, have been progressing rapidly in the last five years. The knowledge representation techniques and problem solving strategies used in expert systems are quite different from conventional programming languages. Rather than requiring well-defined algorithms and dealing with primarily numerical computations, expert systems are based on heuristic rules and deal primarily with symbolic processing. Expert systems tend to behave as human experts and are highly interactive. These characteristics make expert systems an appropriate tool for assisting humans in solving the problems which can not be defined mathematically but are solved daily by human experts using both domain knowledge learned from books and experiences gained from practice. One thing that should be emphasized here is that experiences play an very important role in an expert system. The heuristic rules used in expert systems are often drawn from the experiences of human experts.

In this section, several prototype structural design expert systems are briefly presented, followed by a discussion on the use of knowledge and a potentially powerful problem solving approach.

2.1. Existing Structural Design Expert Systems

In the field of structural engineering, expert systems for preliminary design problems have l™n attempted. Among the prototypes are HI-RISE [4], LOW-RISE [6], DESTINY [5] and ALL-RISE [5) They all explored, to different extents, the issues in developing design expert systems, such as des ^n knowledge representation, constraint formulation and handling, and strategies for solution generation.

2.1.1.HI-RISE

HI-RISE, implemented in PSRL[1O], is a knowledge-based expert system for the prelim r;-y structural design of commercial or residential high-rise buildings which are rectangular in s^a^e Given the space planning of a building, which is described by a three dimensional grid, HI-RISE can produce feasible structural systems.

Structures in HI-RISE are represented hierarchically using schemas. The structural systems are decomposed into two functional systems, the lateral system and gravity system, which are further decomposed into subsystems of several levels. There are a number of subsystems at each hierarchical level in the knowledge base of HI-RISE. The lateral system is decomposed into 3D, 2D and material alternatives, as shown in Figure 2-1. The gravity system is decomposed into 2D horizontal, support, and subdivide alternatives, as shown in Figure 2-2. During synthesis, HI-RISE generates feasible combinations of all subsystems and forms a solution tree. Infeasible alternatives are pruned during the synthesis as soon as possible using synthesis constraints in the form of heuristic elimination rules. After generating the feasible solution tree, HI-RISE does an approximate analysis of each functional system and chooses parameters for the components which make up the functional system so that key design constraints are satisfied. An evaluation is performed to rank the alternative designs and the best system is recommended to the **user.**

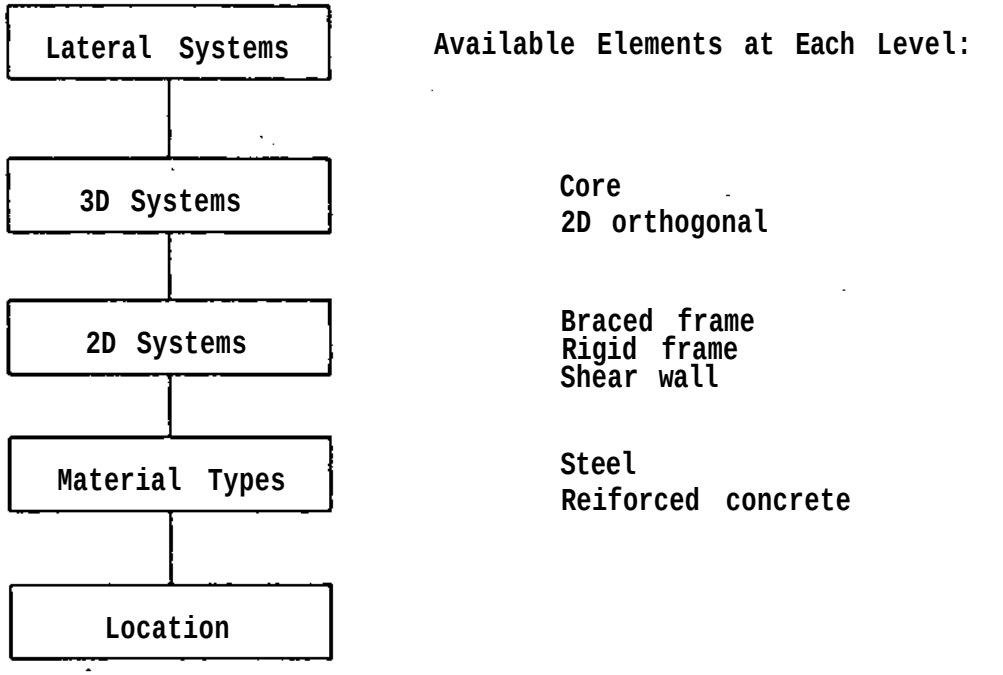


Figure 2-1: Levels **of** synthesis for lateral system[1]

HI-RISE has a group of heuristic design constraints aiding in defining the feasible combinations of the elements at different levels. These constraints are represented by elimination production rules and are stored in the knowledge base of HI-RISE. When the number of elements at each levef s lar^e. the number of feasible combinations may become very large. Evaluating these combinations will

[1] **Figure 2-1 is adapted from [4]**

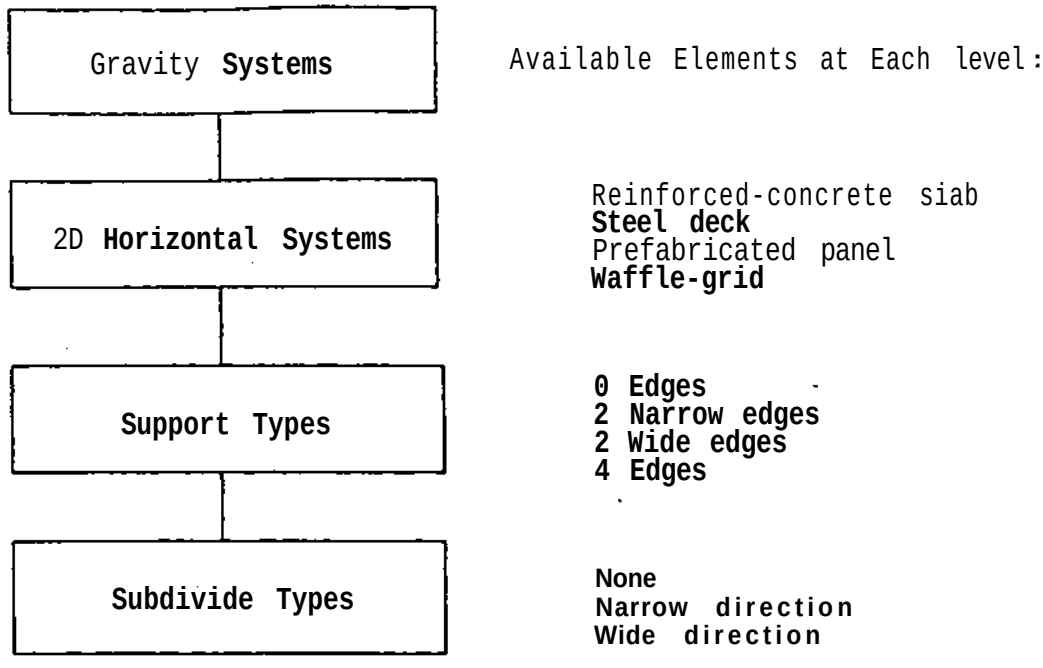


Figure 2-2: Levels of synthesis for gravity system[2]

involve a large amount of computation.

2.1.2. LOW-RISE

LOW-RISE, implemented in OPS5 [9], is a knowledge-based expert system for the structural planning and preliminary design of industrial-type buildings, i.e. one-story, steel buildings which enclose large open areas. LOW-RISE has three main capabilities: structural grid planning, preliminary design and evaluation of alternatives. Given the loads, soil conditions and spatial constraints, LOW-RISE defines appropriate grids for the given constraints and selects feas be structural systems to provide alternative framing schemes. The alternatives are then evaiu :••-< ranked and presented to the user.

LOW-RISE has an archival knowledge base which contains various framing systems with :• ·M properties. Different from HI-RISE, which only has generic structural elements of different leve's [n] 's knowledge base, the framing systems stored in LOW-RISE's archival knowledge base are a[r] :. assembled; examples are trusses and wf purlins, and rigid frame and wf purlins. In order to select :• •* feasible framing schemes, LOW-RISE matches the grid patterns that it generates with all the fra- ; systems in the archival knowledge base. Since the buildings considered by LOW-RISE have open areas, the spans of the framing systems are the governing factor. The framing systems :~ .t

[2] Figure 2-2 is adapted from [4]

conflict with the spatial constraints are eliminated using heuristic rules. The proportioning and evaluation processes in LOW-RISE are similar to those of HI-RISE.

2.1.3. DESTINY

DESTINY, implemented in SRL[11], is an integrated structural engineering design system developed for the purpose of demonstrating the feasibility of providing a domain-independent uniform framev/ork for integrating preliminary design, analysis and detail design. DESTINY is composed of a knowledge-base, a global blackboard, and an inference mechanism. The knowledge-base consists of several knowledge-modules(KMs) and has three levels. The top level is the strategy level where the KM contains the control knowledge which sets the execution sequence of lower level specialist KMs. A specialist KM performs an individual design task and has it own inference mechanism which supervises the execution of all its subtasks. At the bottom level are the resource KMs which provide different facilities such as a database management system. The global blackboard is used for the communication between different, relatively independent knowledge sources or KMs. Any information produced by a KM and needed by other KMs is posted on the blackboard and can be accessed by other KMs. The information maintained on the blackboard can be classified as two types: control information and the solutions, which are produced by the various KMs. Correspondingly, the blackboard is divided into two parts. The first part consists of the execution information of various KMs. The second part consists of different levels representing the abstraction hierarchy of buildings used to store the solution generated by various KMs. The inference mechanism is responsible for monitoring and controlling the blackboard. A more detailed description can be found in [5].

2.1.4. ALL-RISE

ALL-RISE is one of the knowledge-modules used in DESTINY which performs the syntho[r] s *J* feasible structural systems for an input building. The motivation for developing ALL-RISE-[:] » •: provide a framework for building structural synthesis systems. ALL-RISE is an extension to HI [-] it uses hierarchical representation, top-down synthesis and constraint handling. ALL-RISE deals [-.] *» the same types of buildings as HI-RISE does, but is extended to include low-rise and medium [r] -? buildings.

ALL-RISE has a context with an organization similar to the DESTINY'S blackboard. The cont-[:-] two parts. The first part contains the general information which coordinates the subtasks; *v' :* "-? second part contains the solutions generated by ALL-RISE for the input building. Differer.t [''] - HI-RISE, ALL-RISE provides a *flexible sequencing* for generating alternative subsystems, in H! •• [-] the design sequence is fixed: the lateral system is considered first and the gravity system sv–. [-]

Since ALL-RISE deals with buildings from low to high rise, the design sequence is determined according to the input building. For example, when the input building io a high rise building, the design sequence will be the lateral system first and the gravity system second. If the input building is a low rise building, then the gravity system will be designed first. When a subsystem needs the information of another subsystem which has not been generated, interaction constraints are posted. Similar to HI-RISE, the alternative with the best evaluation will be selected and posted on DESTINY'S blackboard to serve **as** the model for the analysis KM and detail-design **KM.**

2.2. Classification of Knowledge

A designer uses many different types of knowledge during the design process. The development of **an expert system to aid or automate the design process involves the formalization and** representation **of this knowledge. According to Paul Harmon and David King, knowledge can be** classified as two types, *surface knowledge* and *deep knowledge,* **as** illustrated **in Figure 2-3.**

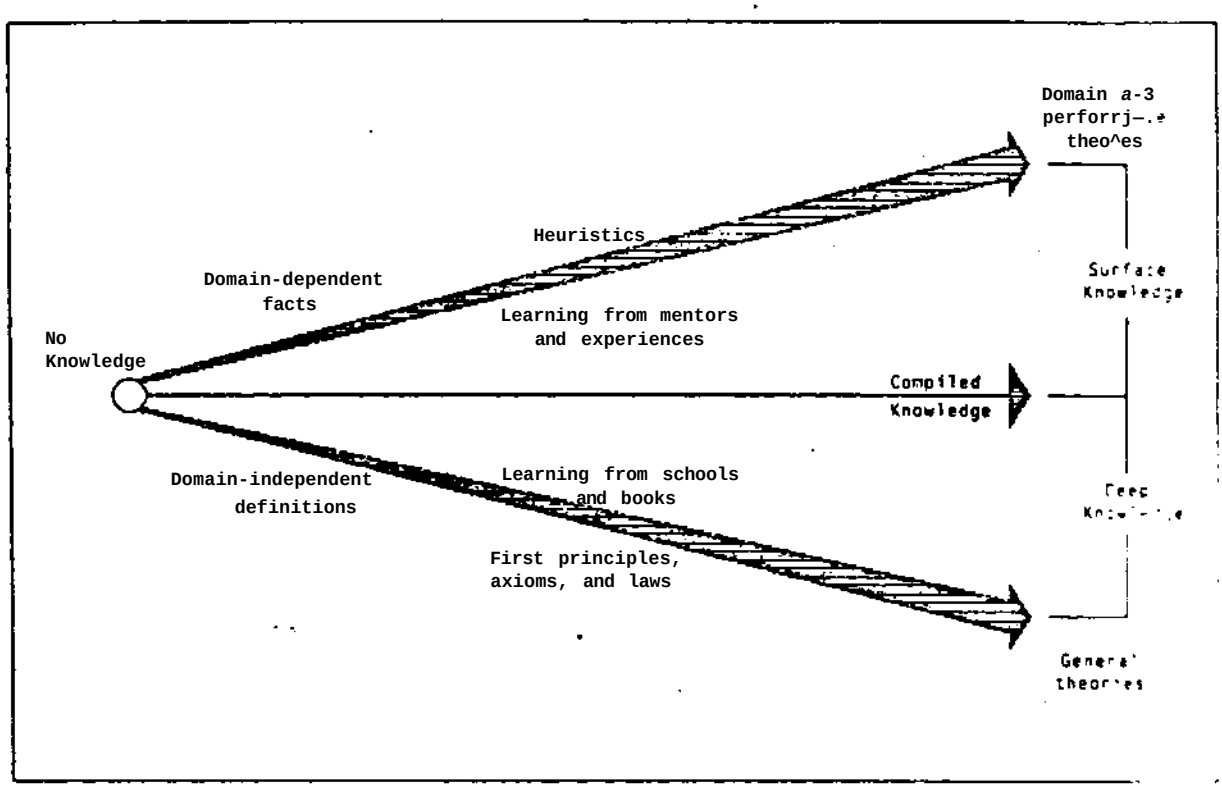


Figure 2-3: Varieties of knowledge[3]

In the figure, the horizontal arrow indicates how compiled knowledge is acquired. Co

[3] **Figure 2-3 is taken from [1]**

knowledge is "information that is organized, indexed and stored in such a way that it is easily accessed." [1] One can acquire compiled knowledge by either learning from text books and in schools or learning from mentors and experiences. General theories learned from schools are powerful for verification or explanation of phenomena or solutions of problems, but rarely provide insight into the solutions of practical problems. Domain theories may guide people more specifically to find the solution paths but are still unable to identify the exact paths. For practical problems, the paths that lead to the solution are usually found by applying domain theories, sometimes general theories, and by trial-and-error methods.

In the preliminary stages of engineering design, the engineer primarily uses surface knowledge. And even though general theories and domain theories such as static equilibrium and material strength are applied as the basic principles that the design must follow, a lot of design constraints come from non-technical sources: designer's taste, client's special requirement, public policy and economy all play important roles in the design decision making and implementation. Theories, or compiled knowledge, are not sufficient for engineers to handle these constraints which have complex interactions. An expert designer, who understands the theories well and has rich experience gamed from practice, can solve a design problem under such complex constraints efficiently and effectively, i.e. experience is as equally important as the theories in engineering design.

Most expert systems which have been implemented use compiled knowledge. The same is true with HI-RISE and ALL-RISE. In these systems, knowledge about structural configurations is generalized. The expert systems only know what subsystems are available to compose a structural system but do not know any particular structural scheme. Structural subsystems are combined to form schemes and infeasible schemes are eliminated by heuristic elimination rules represented as *,i^* gn constraints. This solution generation method can produce a lot of solutions all of wh::* -.•* applicable to a particular problem, but it may not be efficient.

In contrast, successful experience gained from one situation is usually not ready to be *c* <•--"/ apllied to other problems. However, it could be very helpful for solving another problem when •- s problem has very strong similarity to the old problem in some important aspects. In fact, pec;: •• '•? always using their accumulated experiences to solve the problems similar to those they solved c:' -* This means that the methodologies of using successful experience in problem solving should ^ : -? exploited for expert systems. Thus an interesting issue arises: how experience can be direct!/ L , : * 3 solve problems. This issue is discussed in the next section.

2.3. Problem Solving by Analogy

There are many problem solving approaches which have been studied and developed for AI research and building expert systems. Examples are forward chaining, backward chaining, heuristic generate-and-test, backtracking, hierarchical planning and least commitment principles, means-ends analysis, constraint handling, and analogical reasoning. An overview of these problem solving strategies can be found in [4, 5, 3]. The analogical reasoning approach is relevant to the work described in this paper.

The analogical reasoning approach has not been as well developed as the others mentioned before, but it is certainly an important approach which has great potential and power as a problem solving mechanism. Carbonell suggests that analogical reasoning is "a central inference method in human cognition". Though there has been debate on this assertion, it can be observed that it is very common and also successful for humans to solve various problems, daily, technical or scientific, by analogical reasoning. The following definition of analogical reasoning is given by Carbonell. [2]

> **Definition:** *Analogical problem solving consists of transferring knowledge from past problem solving episodes to new problems that share significant aspects with corresponding past experience -- and using the transferred knowledge to construct solutions to the new problems.*

Two types of analogy have been studied by Carbonell.

1. **Transformational analogy.** For old and new problems, if the problem statements, problem solving process and solutions have strong similarity, the past solutions can be transferred, i.e. retrieved, modified and augmented, to satisfy the criteria of the new problems. [8, 2]

2. **Derivational analogy.** Sometimes, even though the old and new problems have similar problem statements and problem solving process, the resultant solutions may bear little, if any direct, similarity. In such cases, the reasoning steps in the construction of the past solutions may be retrieved and modified in order to construct derivational paths toward the new solutions. [2]

For now, our interest is the transformational analogy because it is more natural for structural design

The analogical reasoning approach can be decomposed into four subproblems [2].

1. Identify the "significant aspects" shared by old problems and new problems.

2. Select the relevant successful experiences from a vast long term memory, for instance, a database.

3. Determine what knowledge is to be transferred from the past experience to the new solutions.

4. Construct the transformation engine to embody the transformation process.

Generally, the transformation engine should be able to select the best and most closely-related solution for a new problem from a set of the solutions to past problems with the help of a similarity metric, and then transfer the solution to fit the new problem. In addition, the transformation engine should be able to learn and remember the experience of a transformation so that when a similar new problem is encountered, the transformation engine does not have to reconstruct the transformation all over again and can apply the transformation process knowledge directly to the new problem. This last feature is the subject of machine learning in artificial intelligence and is not to be considered for now.

In [7], Carbonell gives an approach for solving scientific problems by transformational analogy using means-and-ends strategy. Typically, scientific problems can be described by mathematical formulas and thus have initial states, final states, and a set of operators which are applied to the initial state and bring the initial state to the final state through a series of media states. However, engineering design problems are very different from mathematical or physical problems. They are not well defined. Though usually the initial state, i.e. the design problem statement and some of the design specifications are given, little is known about the final state, i.e. the design solution, except a few major properties and specifications. Also, there are no predefined operators which can guarantee that a satisfactory design solution be achieved. Therefore, the transformation process for design experiences should be studied and an analogical transformation approach for engineering design problem solving has to be developed.

3. STRUPLE: Structural Planning from Experience

STRUPLE is an expert system which is being developed in Civil Engineering Department at Carnegie Mellon University. STRUPLE uses a database to store structural design solutions collected from several structural consulting firms or construction companies. The major capacity of STRUPLE is: given the description of a building, mainly the architectural information and some structural information such as the loads and the construction material that the designer prefers to use, STRUPLE will find relevant past structural solutions in the database and use the information existing in the old solutions to plan the structural configuration of the new building.

3.1. Finding Similar Buildings

Recall the four subproblems of analogical transformation presented in Section 2.3. The first two are to identify the strong similarity shared by the old and the new problems and to select the most successful experiences from a database. Here the *problems* are design tasks of buildings and the experiences are the design solutions for buildings. These two subproblems are completed by «L -li'

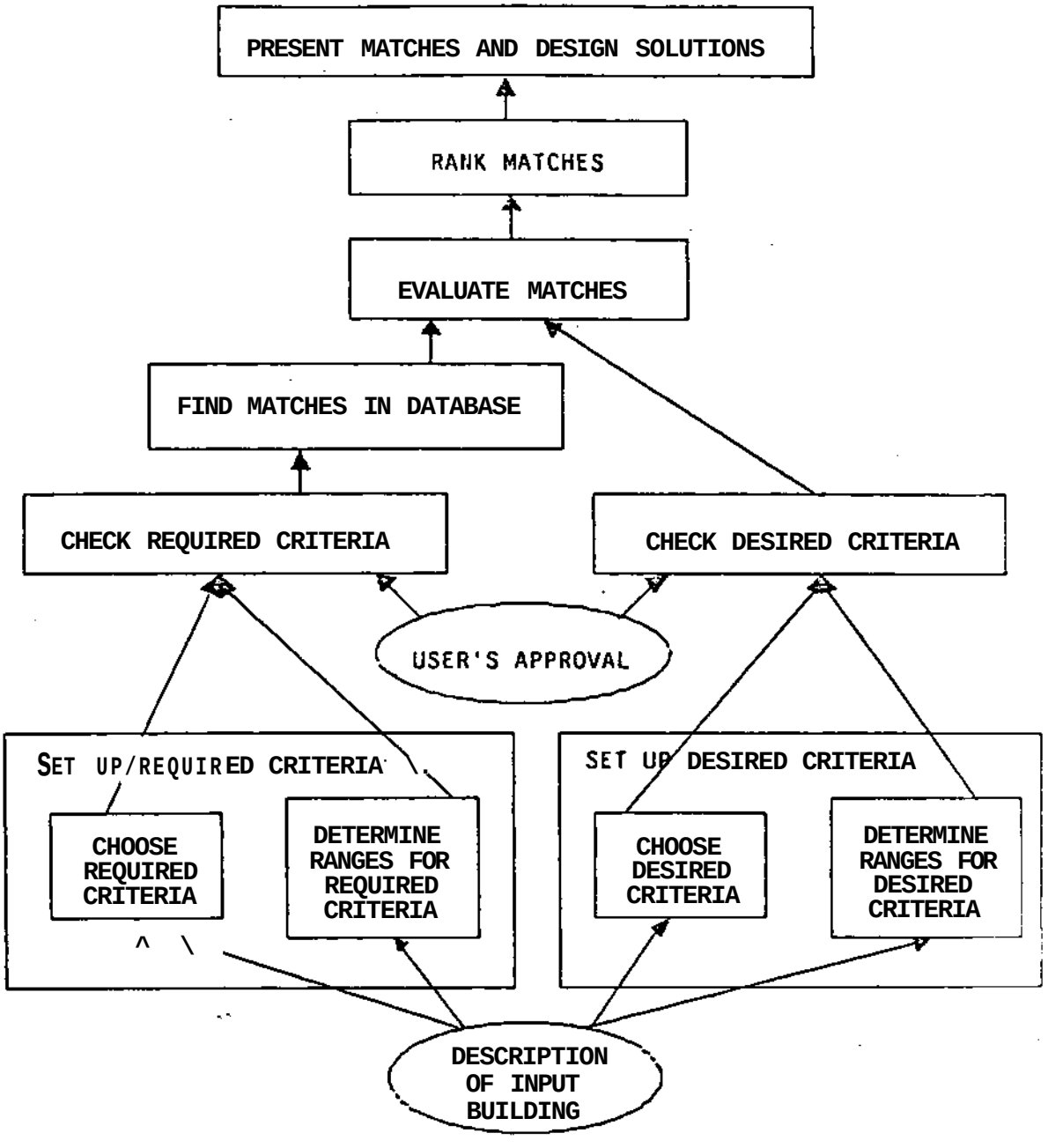
Intelligent DataBase Interface, which is a part of STRUPLE. ID8I is a prototype expert system, currently functioning as an interface between a human designer and a database of building cases. The building cases include architectural and structural Information about buildings whose designs and constructions have been completed. In order to determine whether these design solutions are successful, they should be evaluated by experts from not only the viewpoint of structural engineers, but also the viewpoints of other engineers participating in the project. The evaluation should be made on a holistic basis including the aesthetic, function and economy of the design solution. Here, we assume that all the buildings which are stored in the database are successful designs in order to simplify the problem.

The input to IDBI is a partial description of the architectural plan and some structural information of a building to be designed. The output of IDBI is a list of similar buildings retrieved from the database. The evaluations of the buildings and their rankings based on their similarities to the input building are also provided. The detailed architectural and structural information about the similar buildings can also be retrieved from the database and used to plan the structural configurations of the building to be designed.

The buildings which are considered by IDBI to be similar to the given building are called *matching buildings,* or sometimes *matches.* To find matches requires a similarity metric to determine which buildings in the database are similar to the input building. In IDBI, the similarity metric is described by a set of *criteria,* also called *matching criteria,* which define what significant common aspects the matching buildings and the input building should share. A matching criterion is a requirement of similarity imposed on a *feature* of a matching building. For example, the number of stories, the intended use, the design wind load, the construction material, etc. all are features of a building and are potential criteria. Obviously, not all features of a building are equally important in terms of :N;-.r impacts on the structural system. Thus, criteria are divided into three classes: required catena. desired criteria and no-match criteria, where required criteria must be satisfied by a match ng building, the satisfaction of desired criteria is desirable but not necessary and no-match criteria are not considered. *Required, desired* and *no-match* are qualifiers that represent the *status* of a enter.on

When IDBI searches the database for similar buildings, required criteria are used as qualifiers n order to locate a matching building explicitly, we need to know the limits within which the building cin be considered to be a match. Thus a *range* must be set for each criterion. For a required enters. the range is defined by the limits of allowable difference of the feature values of the two build - ;s compared. To measure how well the matching buildings resemble the given building, each ma?c*i s evaluated and ranked. A *single evaluation* is a measure of the difference between the values of ce

feature **of** a match and **the** given building. The *final evaluation* is a comprehensive measure combining **all individual evaluations for a match.**



Figu re 3-1: General process of IDBI

Figure 3-1 illustrates the general process that IDBI goes through. First, IDBI asks the user to prov *-te* **the description of the building to be designed. Then according to the input information,** appropriate **criteria as well as their ranges are set up by IDBI for matching buildings. At this point, the** user can **view and change the status of the criteria. After the user approves all the criteria, IDBI** searches t[r]**e**

database to find all the matches which satisfy the required criteria. If any matches are found, IDBI evaluates and ranks them, and presents them with detailed information **to** Vie user.

Criteria	Status
unit cost	required: if user specifies
intended use	requi red
-stories above grade	requi-red
shape	requi red
typical bay size	required: if the intended use is office desired: if the intended use is residential
floor-to-floor height	required
floor-to-ceiling height	required
seismic zone	required
wind load	required: if stories >= 30 desired: if 5 < stories < 30 no-match: if stories <= 5
live load	desired
construction materials	required: if user specifies particular materials
3D system	required: if user specifies particular 3D systems
2D system	required: if user specifies particular 2D systems
floor system	required: if user specifies particular floor systems
foundation systems	required: if user specifies particular foundation systems

Figure 3-2: Criteria status

The status **of a** criterion is assigned by IDBI according to the description of the input building Whether a criterion is defined as required or desired depends on the extent to which it would aid :n determining the structural system for the input building. For instance, when a building has over thirty stories, the lateral system design will probably govern the design or at least be as important as **the** gravity system. In this case, wind load is chosen to be a required criterion. If the input building has

less than thirty stories, wind load is not so important and is defined as a desired criterion. When the building has less than five stories, wind load will not be considered in the design, so it is defined as a no-match criterion. Figure 3-2 lists all the criteria used in IDBI as well as their status assumed in different cases.

A required criterion is satisfied by the matching building if the value of its corresponding feature falls within the range of the criterion. The ranges of criteria are set heuristically according to the values of the input building features. For most building features which have numerical values, e.g. floor-to-floor-height, live load, etc., ranges are set quite arbitrarily to be from 0.85 to 1.15 times the corresponding input values. There are three exceptions: the range of wind load is from 0.8 to 1.4 times the input wind load, that of seismic zone from input zone to input zone + 1, and the range of stories is determined according to the range limits shown in Figure 3-3.

Stories of input building	Range limits
1 - 5	0
6 - 10	t 1
11 - 20	t 3
21 - 30	£ 4
31 - 40	£ 5
41 - 50	t 8
51 - 60	± 10
61 - 150	± 20

Figu re 3-3: Ranges For Stories Criterion

There are a few features which do not have numerical values but can be discretized; examples are intended use and construction material. For matching these features, check-lists are used. Check lists contain all the allowable alternatives that are considered to be matching values of a feature. For instance, the input building's intended use is always an alternative and so is in the check-list. T/:o particular extensions of use are made when the input building's stories are more than 40 and the ^v? is residential or commercial. In such cases, both commercial and residential are considered sTP-!ar and appear in the check-list for intended use. IDBI allows the user to change the status and the ranges of the criteria and checks whether the changes made by the user are reasonable.

Evaluation of a matching building is completed by *single evaluations* and a *final evaluation.* First, a single evaluation is made for all the matching buildings according to each criterion, and then the final evaluation is made based on the results of the single evaluations. The value of each feature of a match is used for a single evaluation. The value of a single evaluation is real number between 0.0 and 1.0. A positive value is considered a penalty in the sense that 0.0 means an exact match while 1.0 means no match or that the difference is very big.

A single evaluation is done in different ways according to whether the corresponding criterion has a numerical range or check-list associated with it. The following formula is used for a single evaluation of a criterion with a numerical range.

$$\text{evaluation} = \frac{|\langle v\rangle - \langle \text{input-v}\rangle|}{\langle u\rangle - \langle l\rangle}$$

where <v> = the value of the feature of a match
<input-v> = the value of the same feature of the input building
<u> = the upper limit of the range for this criterion
<l> = the lower limit of the range for this criterion

If the result is greater than 1.0, then it is set to 1.0. A single evaluation of a criterion with a check list is based on the the following rules.

```
IF    <v>  is  in  the  check-list
AND   <v>  is  the  same  as  <input-v>
THEN  the  single  evaluation  is  0.0

IF    <v>  is  in  the  check-list
AND   <v>  is  not  the  same  as  <input-v>
THEN  the  single  evaluation  is  0.4

IF    <v>  is  not  in  the  check-list
THEN  the  sing/ls  evaluation  is  1.0
```

Here <v> and <input-v> have the same meanings as before, but they are strings of characters n:*ejd of numerical values.

Cost is an important factor in the selection of structural systems and if the user desires, it is me:•.4. --1 as another single evaluation. In the database, the unit cost of each building case is stor»?<i ? S/sqft). The evaluation is made differently in two cases. In the first case, the estimated ?.r ,--r\ budget is given and IDBI calculates the estimated unit cost by dividing the estimated cost by the y s area of the input building. In the second case, the project budget is tight but unknown and ;u? t evaluation is made on a relative basis. The maximum unit cost and the minimum unit cost cf : I

matching buildings are found and IDBI uses these values to evaluate each match with the following formula.

$$\text{evaluation} = \frac{\langle\text{cost}\rangle - \langle\text{min-cost}\rangle}{\langle\text{max-cost}\rangle - \langle\text{min-cost}\rangle}$$

where <cost> = the unit cost of the match being evaluated
<min-cost> = the minimum unit cost among all the matches
<max-cost> = the maximum unit cost among all the matches

The result of the evaluation is always between 0.0 and 1.0.

Criteria	Weights	
	status = required	status = desired
unit cost	10.0	6.0
intended use	9.0	4.0
stories above grade	8.0	4.0
shape	9.0	4.0
typical bay size	7.0	5.0
floor-to-floor height	8.0	3.5
floor-to-ceiling height	8.0	4.0
seismic zone	8.0	4.0
wind load	7.0	3.0
live load	7.0	3.0
construction materials	8.0	--
3D system	8.0	--
2D system	8.0	--
floor system	8.0	--
foundation systems	8.0	--

Figure 3-4: Weights Used For Evaluation

The final evaluation of a match is the summation of all the evaluations of a single feature multiplied by the corresponding weights in the weight table. The weights reflect the degree of importance of the criteria. The weights currently used by 1DBI are shown in Figure 3-4.

Each criterion has two weights; one to be used if the criterion is required and one if it is desired. This is due to the fact that when a required criterion is considered, the weight should be greater than that corresponding to a desired criterion. The weights used in IDBI are assigned based on the heuristics gained from the author's experience and not from any publications or designers.

Ranking of the matches is done according to the final evaluation values of the matching buildings. A match which has a smaller evaluation is ranked as a better match since a smaller evaluation value indicates less difference between the match and the input building. The rank of a match is indicated by an integer. The purpose is to provide a clearer view to the user of how one match compares with the others.

3.2. Transforming Knowledge from Past Experiences

Once matching buildings have been found which represent the experience that may be relevant to the new design problem, we have to determine what knowledge is to be transformed and used to solve the new problem, which is the third subproblem of analogical transformation. It is a fact that even though two buildings could have many common aspects, because of their different special design requirements, the design solutions may not be the same. However, it is also a fact that a certain group of design vocabulary frequently appears in certain types of buildings. Although we can not directly use the design solution of an existing building for a new building, we can use the design vocabulary of the old design to construct the new solution. Thus we choose the design vocabulary as the knowledge to be extracted from the old design solution and be transformed.

The database may have a vast amount of design solutions and many of them may be found *o *T,e* similar to the new building. In order to use the information efficiently, the frequency of each e'v-^nt of design vocabulary used in the old designs is calculated. According to their frequenc -s of appearance and evaluations based on their similarity to the new building, each type of e«err--! s assigned a priority of consideration. A very high priority indicates that this type of e!e^~'-t s frequently used in the buildings of the type of the new one and should be considered first.

To embody the transformation of knowledge existing in the past experiences is the ' -'h subproblem. The transformed experiences are considered to be the different types of s:r^\.- ii elements used in similar buildings. STRUPLE has a set of structural elements that are stored n :s

knowledge base representing the complete design vocabulary that the synthesis process knows about. The structural elements are organized into different abstraction levels, as shown in Figure 3-5. The design vocabulary of lateral and gravity systems is shown in Figure 3-6 and Figure 3-7. For each level of abstraction, STRUPLE determines and orders the design vocabulary to be considered for the new building, thus identifying a subset of the complete vocabulary of structural elements which is most promising. This subset of design vocabulary will be used during the synthesis process, in which each structural element will be examined to determine whether it is an eligible or efficient alternative.

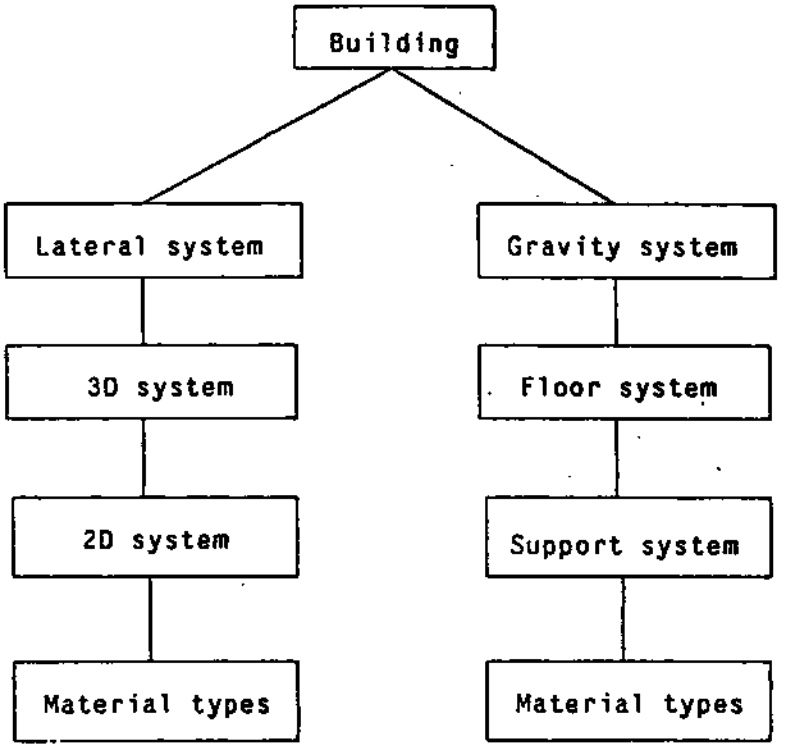


Figure 3-5: Abstraction levels of a building

There **are** several possible situations. First, if all types of elements at one level have close pncr • -s. all of them should be considered one by one during the synthesis. Second, if a few types of e!en:;r#s have much higher priorities than the others, then those with very low priorities may not be cons•-!•:•'-3 if the elements with high priorities are considered optimal solutions. Third, if no similar buiicJ-^-j s found, or there are only a few types of elements at each level with very low priorities, thon •. \ structural elements in the complete vocabulary should be considered. In the third case. i.• • ' .'• experience is not available and STRUPLE has to use the general design knowledge stored n •-, knowledge base. Finally, it is possible that there is a building in the database whose most **irrp;*** . 't characteristics and special design constraints are very similar to those of the new one, in which c^ •>

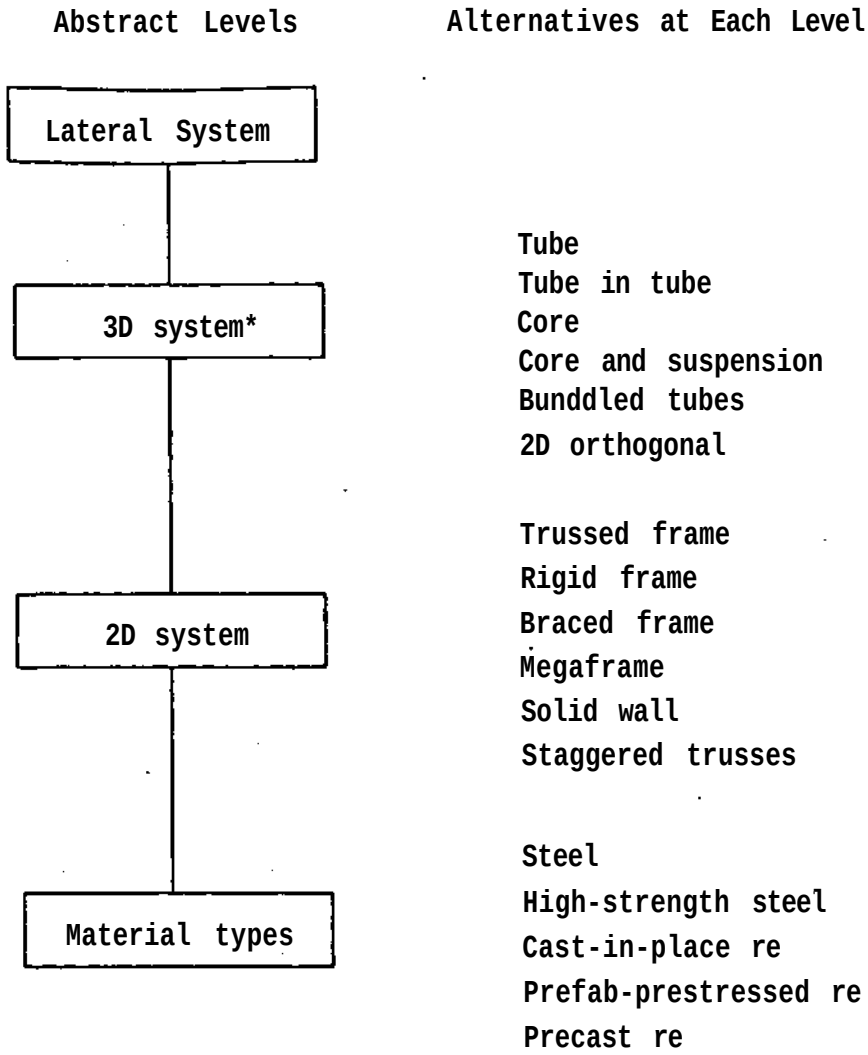


Figure 3-6: Design vocabulary of lateral systems

the old structural system may be directly transformed by reproportioning or making minor changes in the structural system for the new building.

As mentioned **above,** in most cases, STRUPLE can select and order the most promising subset of design vocabulary stored in the knowledge base. During synthesis, the structural elements with higher priorities will be considered first. When one or a few complete alternatives which have an element with high priority at each abstract level are generated and are evaluated to be efficient, they are presented to the user before other alternatives are generated which will be composed of elements with lower priorities. This is more efficient than generating alternatives by combining all known structural elements, especially when the number of elements is large.

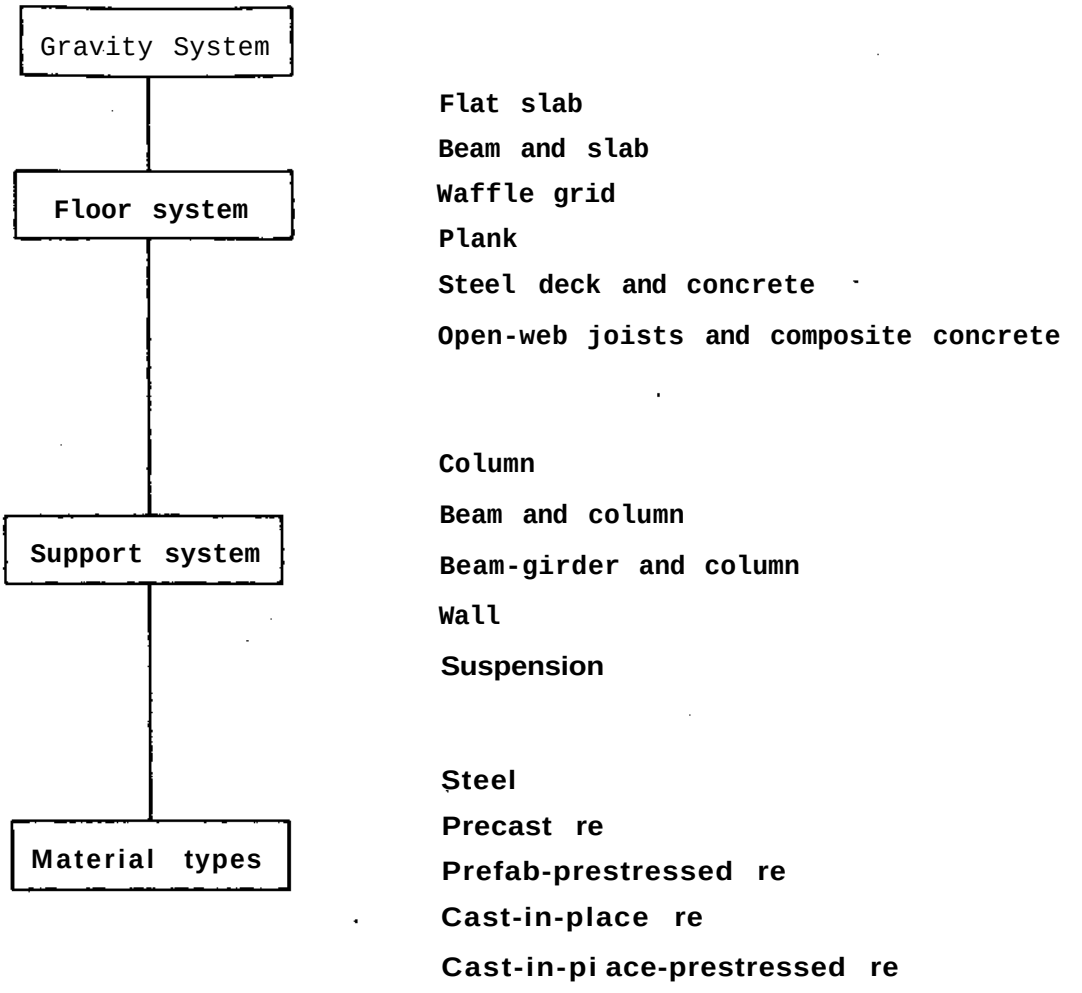


Figure 3-7: Design vocabulary of gravity systems

4. Conclusions

A methodology for using design experience directly has been described. The motivation for ^ s work is to provide a means for capturing and using design experience directly in an expert sys*-?^ "o that the performance of the expert system is not limited by the surface knowledge that c.^ r e formalized. The concepts of design vocabulary and levels of abstraction are used to trans':--^ "e design experience into a form suitable for planning the synthesis of alternative solutions.

The methodology presented, although it illustrates the concept of using design experience J -•.•.• • has some limitations that need to be addressed. The method for determining the similar s t~ -* • i.e. finding the matches in the database, is based on a fixed set of criteria and cannot consider : : : ·

unusual circumstances. A second limitation is the inability of the existing method to benefit from the spatial decisions made in previous situations; currently, the methodc'ogy provides a means for reasoning about appropriate structural systems and subsystems, not about the number or location of these systems.

The potential for using design experience can be considered in two areas. One area is the advantage of the computer's unfailing memory to present relevant experience a human designer may have forgotten. The other area is the potential for machine learning by recording and using the design experience of the expert system.

References

[1] Harmon, P. and King, D.
Expert Systems: Artificial Intelligence in Business.
John Wiley & Sons, Inc.,, 1985.

[2] Carbonell, J.G.
Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition.
Technical Report CMU-CS-85-115, Department of Computer Science, Carnegie Mellon University, March, 1985.

[3] Maher, M.L., Sriram, D., and SJ. Fenves.
Tools and Techniques For Knowledge-Based Expert Systems For Engineering Design.
Advances in Engineering Software , 1984.

[4] Maher, M.L. and Fenves, S.J.
HI-RISE: A Knowledge-Based Expert System for the Preliminary Structural Design of High Rise Buildings.
PhD thesis, Depertment of Civil Engineering, Carnegie Mellon University, January, 1985.

[5] Sriram, D.
Knowledge-based Approaches for Structural Design.
PhD thesis, Department of Civil Engineering, Carnegie Mellon University, February, 1986.

[6] Camacho, G.T.
LOW-RISE: An Expert Stystem for the Structural Planning and Preliminary Design of Industrial-type Buildings.
Master's thesis, Department of Civil Engineering, Carnegie Mellon University, July, 1985.

[7] Carbonell, J.G.
Learning by Analogy: Formulating and Generalizing Plans from Past Experience.
Technical Report CMU-CS-82-126, Depertment of Computer Science, Carnegie Mellon University, June, 1982.

[8] Carbonell, J.G. and Larkin, J.H.
Towards a General Scientific Reasoning Engine.
Technical Report CUM-CS-83-120, Department of Computer Science, Carnegie Mellon University, April, 1983.

[9] Forgy, C. L.
OPS5 User's Manual.
Technical Report CMU-CS-8M35, Carnegie-Mellon University, July, 1981.

[10] Rychener, M. D.
PSRL: An SRL-Based Production-Rule System.
Reference Manual.
1984

[11] Wright, J. M. and Fox, M. S.
SRL/1.5 User Manual.
Technical Report, CMU Robotics Institute, June, 1983.