The Conceptimator: An Expert System for Conceptual Cost Estimating of Building Foundations

by

Ellen R. Tauber, Raymond E. Levitt,
Gaye A. Oralkan, Felix C. Reinberg, Timothy J. Walsh

CIFE Working Paper
Number 14

June, 1992

Stanford University

Table Of Contents

The Conceptimator: An Expert System for Conceptual Cost Estimating of Building Foundations

By Ellen R. Tauber, Raymond E. Levitt,
Gaye A. Oralkan, Felix C. Reinberg, and Timothy J. Walsh

1. Abstract

The Conceptimator is an expert system we wrote which generates conceptual cost estimates of building foundations by simulating the design process. It is designed to be used not by civil engineers or building construction experts, but rather by non-technical users, such as land owners who have an idea for a building project. The Conceptimator puts the knowledge of many experts in the building construction field into the hands of the future building owner by producing a detailed conceptual estimate at a stage of the project when some of the data needed to estimate unambiguously are missing. The system outputs: a product model for the designed building, which shows the selected foundation; an Autocad drawing of the foundation; and a report summarizing the inputs to the system, the allowable foundation types and cost for each one, and a detailed description of geometry and cost of the chosen foundation. The Conceptimator is a working system which is a proof of concept prototype; it shows that conceptual estimating can be done by design rather than classification. The Conceptimator's building design and estimate can be used as a new component in project cost control in the role of a *defending champion.* Since it generates a detailed design at an early stage in the project, it could be used by the project owner as a comparator to competing design ideas proposed later in the project by human specialists, such as the structural engineer or contractor. If the estimated costs later in the project are above that of The Conceptimator's cost estimation, the owner can hold The Conceptimator's estimation as the defending champion against new solutions proposed by the contractor. New solutions must defeat The Conceptimator's estimate on a cost versus functionality trade-off before they are accepted.

2. Introduction

In this paper, we describe in detail an expert system which produces conceptual cost estimates and designs for building foundations. We will first discuss, in this section, why The Conceptimator is a new approach to conceptual estimating by differentiating between two types of problem solving. Then we will review the methodology of the project in section 3, followed in section 4 by a detailed description of the expert system shell, Design++, and the design of The Conceptimator's architecture. In section 5, we review the results of the project. The Appendix shows some of the code for the system.

2.1. Conceptual Cost Estimating Defined

Cost estimating involves predicting expected costs for a future project or task. *Conceptual* refers to a stage in a project or task when the final product is not fully defined. Specifically, during the conceptual phase, those involved focus on function more than form and do so in a high-level, abstract manner (Dym and Levitt, p. 248). Thus, *conceptual cost estimating* is the task of predicting expected costs for a future project when the form of the final-product is not completely defined. Conceptual estimates are made in the early phases of a project to "tell an owner whether a contemplated project scope is anywhere near to being economically feasible"(Barrie and Paulson, p. 199).

The name *Conceptimator* is derived from the words *conceptual* and *estimator*. There are other common types of conceptual estimating, such as conceptual time estimating. Therefore, although the conceptual estimating task performed by The Conceptimator is only a cost estimation, we will sometimes omit the term *cost* when referring to the system and will simply use the term *conceptual estimating*, by which we mean *conceptual cost estimating*.

2.2. Types of Problem Solving: Synthesis vs. Classification

Saul Amarel outlines a spectrum of problem-solving tasks which can be used to divide all engineering tasks into two broad categories: *derivation* or *classification tasks* versus *synthesis* or *formation tasks* (Amarel in Dym and Levitt, pp. 2-3). Derivation tasks involve picking a solution from a small number of possible solutions. The problem solver can use associative heuristics to derive a solution from data describing previous problems and historical unit factors. Thus, one can solve a classification problem by simply choosing a position along a one-dimensional scale. In contrast, synthesis tasks involve the formation of a solution from primitive elements. These type of tasks require the formation of solutions in a semi-infinite solution space, because the problem solver's task is to design a solution that meets certain constraints. Typically, derivation problems are the easier of the two because the possible solutions are few in number and known in advance, while synthesis problems necessitate the design of new solutions.

The formalization and automation of engineering tasks parallels their difficulty; although many computer systems exist which can be used by the engineer to aid his or her work for analysis tasks, few computer tools in use today by civil engineers can perform design synthesis tasks. The difficult task of synthesis is left to the person (Dym and Levitt, pp. 2-4).

The Conceptimator is a new approach to conceptual estimating, and is interesting as an expert system, because it uses *model-based reasoning* for synthesis rather than *associative reasoning* for classification. That is, it is a system which treats conceptual cost estimating as a synthesis problem involving the detailed design of a building rather than as a classification problem which estimates the cost of a present building by placing it along a range of costs.

2.3. How Conceptual Cost Estimating is Done Today

As stated earlier, conceptual estimates are made in the early phases of a project in order to tell an owner if a proposed project is economically feasible. Since the engineering design of a project is not complete at the conceptual stage, detailed estimates based on computed quantities are impossible. The owner can only provide the estimator with the general concept, site information, and location. The contractor can make an educated guess by using factored historical unit costs and his or her own experience and construction knowledge to give a rough judgement of the proposed project's cost (Barrie and Paulson, pp. 198-199). Figure 1 shows how conceptual estimating happens today as a classification problem in which the estimator relates proposed project to the cost of a past project using an empirical scaling factor in order to form the estimate.

Specifically, according to Barrie and Paulson, there are four common types of preliminary estimating methods: time-referenced cost indices, cost-capacity factors, component ratios, and parameter costs (pp. 201-208). All use past project data and order of magnitude scaling factors to predict a new project's cost. Cost indices are best for estimating changes in cost over time. They use a base reference period to get an index for current costs. Many indices are published. Cost-capacity factors are best for figuring changes in size, scope, and capacity from previous projects. They most commonly take the form of exponential equations. Component ratios are used to estimate the cost of components. Lastly, parameter costs method of preliminary estimating "relates all costs of a project to just a few physical measures, or 'parameters,' that reflect the size or scope of that project."(p. 208). Parameter costs are often

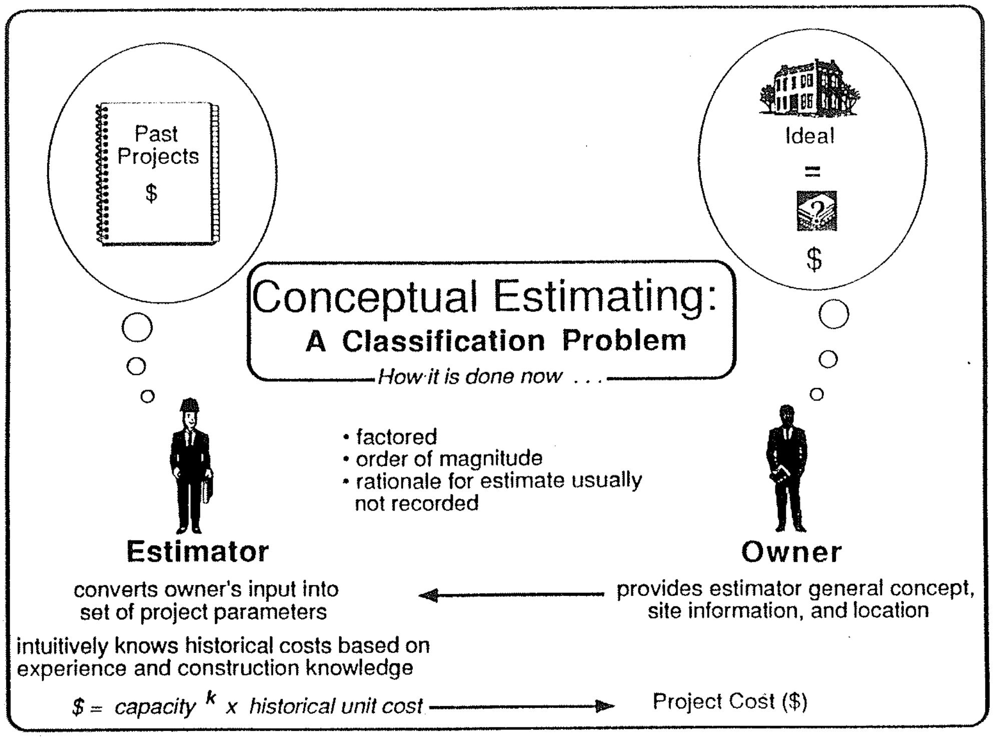

Past Projects $
Ideal = $
Conceptual Estimating:
A Classification Problem
How it is done now . . .
• factored
• order of magnitude
• rationale for estimate usually not recorded
Estimator
converts owner's input into set of project parameters
intuitively knows historical costs based on experience and construction knowledge
$ = capacity ^{k} x historical unit cost
Project Cost ($)
Owner
provides estimator general concept, site information, and location

Figure 1: How Conceptual Estimating is Done Today

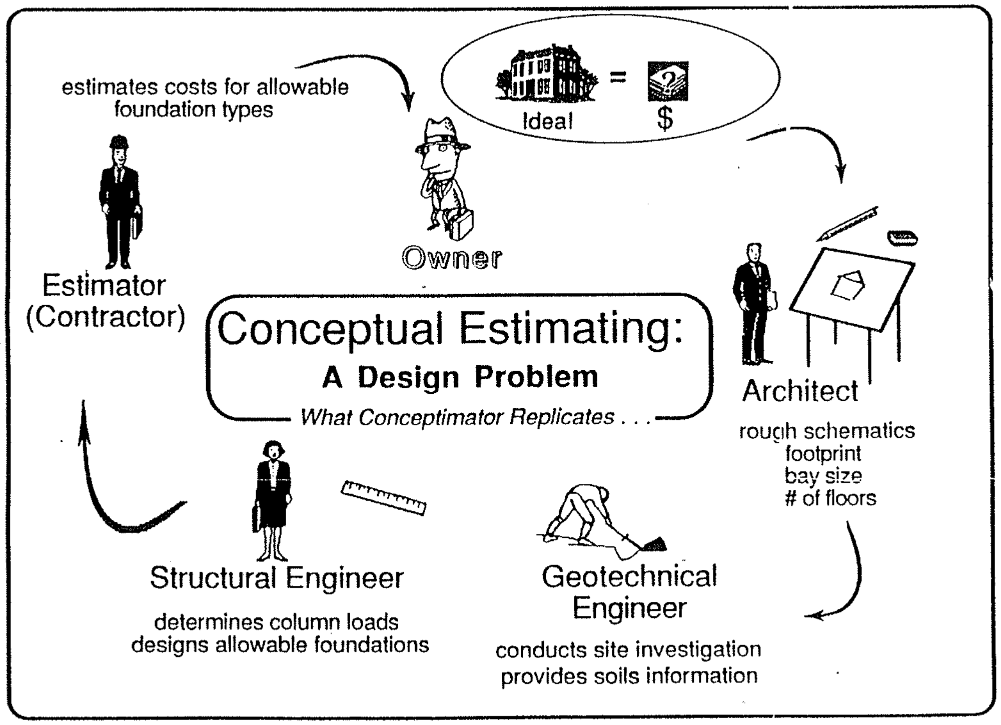


Figure 2: The Process which The Conceptimator Replicates

published. A popular parameter for estimating building costs is cost/square foot based on the type of building.

2.4. How The Conceptimator Does Conceptual Estimating

The Conceptimator generates conceptual estimates by simulating the design process. It allows the owner to utilize knowledge from many experts in the field by replicating the process of detailed design, but at the conceptual stage of the project. Figure 2 shows the process The Conceptimator replicates. The Conceptimator contains the knowledge of the process pictured and places it within the computer.

An approach to cost estimating by modeling the design process was outlined in 1978 by Robert D. Logcher, et al. Their project was called "COSTMOD." The system was "an attempt to model the role of the estimator"(p. 16) for "accurate cost estimation both in the initial phases of design and continuously thereafter"(p. 1). The system was developed without the benefit of AI technologies, and has not been extended beyond a prototype stage.

The owner using The Conceptimator on the computer interacts as shown in Figure 3, but has access to the process pictured in Figure 2. Specifically, the user can input a set of project parameters to generate a product model, Autocad drawing, and a summary report.

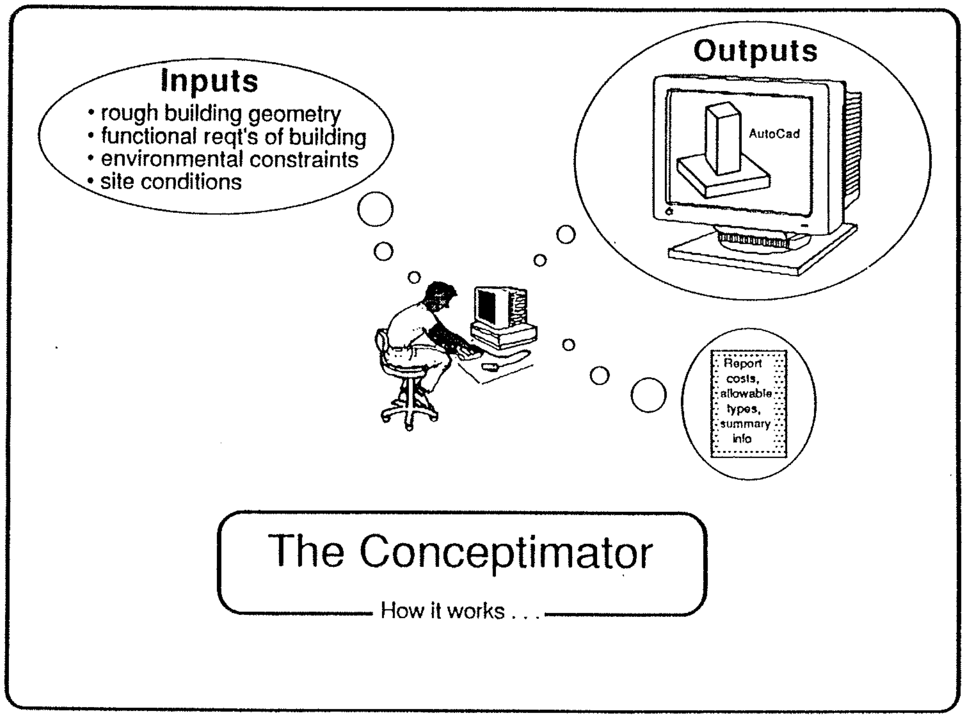

Inputs
• rough building geometry
• functional reqt's of building
• environmental constraints
• site conditions
Outputs
AutoCad
Report costs, allowable types, summary info
The Conceptimator
How it works . . .

Figure 3: How a User Interacts with The Conceptimator

2.5. What The Conceptimator Does

The Conceptimator gathers information from the user about a proposed building by asking questions about the building itself (What is the length of the building?) and about the surrounding area (How far are the adjacent structures?). It selects a set of feasible foundation systems based on approximate engineering calculations and reasoning about environmental considerations. It then designs the components of the feasible systems, accesses an Oracle database for cost information which was created for use with The Conceptimator, computes the costs for each feasible foundation, and selects the least expensive foundation. It then generates the 3-D geometry of the selected foundation and displays it using Autocad, along with a detailed written report (which is incomplete at this time). Also, through the use of *dialog boxes,* the user can look into the system and change values in order to perform a *what if* analysis. For example, the user could change the value of the top soil's material, which would propagate through the system, possibly resulting in new values of other attributes, and maybe even changing the chosen foundation.

3. Methodology

3.1. Compilation of Knowledge

Felix Reinberg, a Construction Engineering masters student with estimating experience, gathered the original conceptual estimating knowledge for the system. Felix also interviewed local companies to gather detailed design estimating knowledge.

3.2. Implementation

Ellen Tauber, a masters student in Computer Science, designed and implemented the majority of the system during the summer of 1991. Ellen worked with Felix to organize the raw knowledge into heuristics and rules which were appropriately formed for the computer system. Much of the design of the system evolved at that time.

Once we had a body of properly outlined knowledge, Ellen implemented the system using Design++, a design automation expert system shell which is described in detail in section 5.1. Gaye Oralkan, a PhD student in Civil Engineering, implemented the geometric rules in the system. As mentioned above, the implementation of the system's report output, described in section 4.8, is not yet complete. Therefore, there is no example system report included in this working paper.

3.3. Validation

The goal of validation is to verify the accuracy of the foundation design rules and the cost data. Tim Walsh, a masters student in Construction Engineering and Management, has studied the present system and is in the process of testing it using four recently constructed Bay Area buildings with four foundations: spread footing, mat, belled caisson, and pre-cast concrete piles. Through comparison of The Conceptimator's results with actual design and costs of the already constructed buildings, we can determine the source of discrepancies and make appropriate changes to rules and knowledge in the system. Target accuracy is choice of the correct foundation and estimates which are within 10% of actual cost. Also, we are documenting sources of rules and data. First indications based on cursory tests of the present application indicate that the system is behaving appropriately.

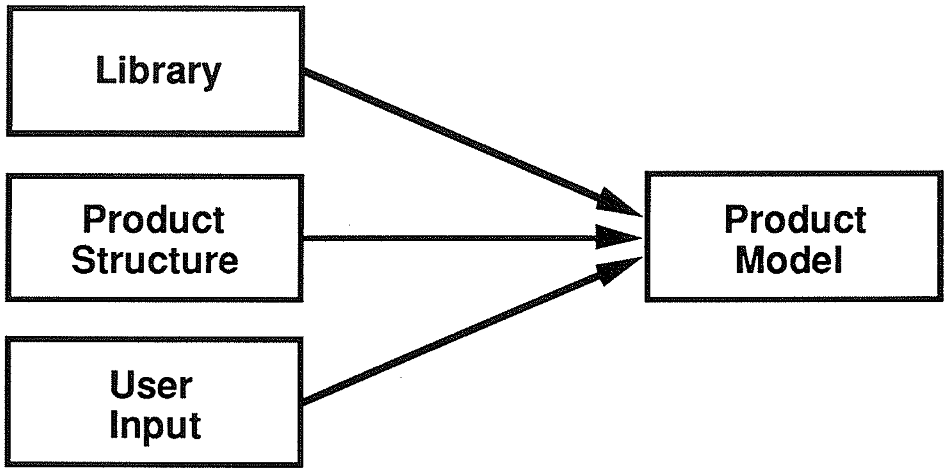
4. The Conceptimator System Architecture

The Conceptimator was implemented using Design++, Version 2.1, running on a Sun Sparc Station. It uses Autocad, Version 11, and an Oracle relational database management system.

4.1. Design++

Design++ is a design configuration system built on top of KEE^{TM}, an expert system shell based on object-oriented programming concepts, which is, in turn, built on Lisp. An *expert system shell* is like a high level programming language for writing expert systems. Expert-system-building tools, such as Design++, "differ from conventional programming languages in that they provide convenient ways to represent complex, high-level concepts"(Waterman, p. 9). Design++ is a one level higher tool than a general purpose expert system shell because it provides the designer with mechanisms to do the following: create libraries of components which store general rules and use inheritance, create specific product models which represent one possibility for the combination of those library components, and interface with Autocad and Oracle. It is the product of a local computer software company, Design Power, Inc., in Cupertino, California.

When the user sits down at The Conceptimator, as with any system built using Design++, the object is to produce one *product model.* The product model is a tree-structure. In this case, it is a building structure which has one selected foundation within it. As shown by the figure below, there are three things used by Design++ to produce a product model: *component libraries, product structure,* and *user input.*



we will go through the library, product structure, and user input, and we will then describe the resulting product model in depth.

4.2. Component Libraries

The system designer defines all of the components of the system in libraries. Thus, the library defines all the types of components which can be used to create a product model. The libraries are tree structures which are *object-oriented* and exploit *inheritance.* Thus, the most general objects are closer to the root of the tree, and the more specific objects of each type are further from the root. General rules which define how the components' values are determined are stored in the library in association with the components. Also stored in the library is other information which defines the component, such as its possible values (*valueclass*) and a default value. The Conceptimator system has one library which is shown in the Appendix, Figure A1.

An example library component which has been "opened" from the library tree is pictured in the Appendix, Figure A2. The component pictured is *top_soil* and its attribute *material* is shown. As you can see, the rule lives in the library and defines how *top_soil material* will be determined. The rule language is a combination of lisp and Design++'s *design rule language.* The *valueclass* limits the possible values of *material* to one of: mud, organic, rock, sand, soft_clay, hard_clay, or silt.

4.3. Product Structure

The product structure is a text file which specifies a generic structure for combining library components. It is somewhat like a recipe which gives an outline to guide the system in creating a product model from the library components. Product structures list, in order, the components to put in the product model. For example, The Conceptimator's product structure, shown in the Appendix, section 10.2, lists *building* first so that every product model created by The Conceptimator will have *building* at the root of its product model tree. If you look at the product structure, you will notice that some of the components are listed giving only their name and some have *:n* before them. The first component listed in that fashion in The Conceptimator's product structure is *:n mat*. All components listed by name must appear exactly one time in the product model at the exact place specified by the product structure. Clearly, if all components were listed only by name, the system would be less flexible; the only variation would be in the attributes -- geometric and functional -- of components. Design++ allows components to be included based on rules, so the number of instances of the component which will appear in the product model at that place is undefined in general. For example, *:n mat* means that some number of mat foundations -- based on the evaluation of a rule in the mat library element -- can appear in the product model under *foundation*. That number may be 0. Thus, Design++ uses user input to flesh out an instance of the product structure with library elements, by evaluating design rules contained in the elements.

4.4. User Input

The user must provide three types of information. First is information about the building itself:
•size and configuration of the building footprint:
The building is assumed to be rectangular at this time. If the system is expanded, it is conceivable that the user could enter any footprint shape using Autocad.
•type of facility to be designed:
The user may choose between hospital, industrial, residential, office, manufacturing, or other. The type of building determines the live and dead load requirements
•area of the building:
The area is figured from the number of floors and the building footprint. In order to do so, the system assumes the building is prismatic. That is, the

configuration of the superstructure must follow the same shape as the building footprint.
•bay size
The user must specify both the bay length and width.
•basement depth

The second type of user input involves site and subsurface inquiries. Specifically, the system must have values for the following attributes:
•Top Soil: bearing capacity, material
•Sub Soil: bearing condition, condition, material
•Bottom Soil: location, bearing support for piles
However, since the inputs are expected to be furnished by a future building owner, not a construction expert, the system does not ask the values directly from the user. Instead, the user may provide one of three types of analyses: boring sample, user inspection of the site, or a United States Geological Survey (USGS) Map. The first level of analyses, and the most desirable one, is the boring sample. If the user has information from a boring sample, s/he can enter that information into the system directly. If not, the system asks the user if s/he has visited the site. If so, the system asks simple questions such as what it looked like, and what the soil felt like. The system can then use the answers to those simple questions to determine the desired information for the top soil. If there was an existing cut that the user observed on the site, s/he can provide similar information for the system to determine the sub soil characteristics. The third and final level of analysis is the USGS map. The USGS makes easily obtainable, detailed maps for most areas of the United States. We provide a map with the system. If the user does not have a boring sample or personal information about the site, s/he can locate the site on the USGS map, enter the USGS code, and the system will locate the desired values in a database attached to the system.

The third type of user information required is information about local environmental conditions. These are often important in deciding if certain pile systems are feasible for a given site. For example, the system investigates concerns regarding allowable noise levels. If the community regulations regarding noisy driven pile systems are strict, those systems will be eliminated from consideration. The system also considers how existing structures will react to new construction by asking questions about the distance of adjacent buildings, the type of area in which the building will built, such as residential or commercial, and the strength of the foundations of adjacent buildings.

To get an idea of what interacting with the system is like, see the Appendix, Figure A3, which shows the screen of the Design++ system as it is asking the user to enter the length of the building which s/he is creating.

4.5. Product Model

As shown in the diagram above, the product model is created from a combination of libraries, product structure, and user input. To be more specific about the interaction of the four parts, the product model is a unique instance of the product structure, built of library elements, and based on rules in the library elements which reference user input. By looking at the sample product model shown in the Appendix, Figure A4, you can see that the product model, like the library, is a tree-structure. The product model, whose root is the product (in this case, a building), is the result of one run of the system. The product model for the designed building shows the selected foundation. In the example product model, we see that the chosen foundation is *spread footing*.

4.6. The Process: How Design++ Creates a Product Model

Now that you are familiar with the parts and output of the system, we will describe the process Design++ goes through when the user asks it to create a product model. To run the system, the user tells it to create a new model. The system asks which product structure to use, which in this case is the *building_struct* shown in the Appendix, section 10.2. The system goes through the product structure line-by-line, adding instances of the listed components to the product model it is creating. The components in the product structure have no values at this time. For example, *building* is in the product model and the attribute *length* is associated with the *building* instance, but *length* does not have a numerical value yet. The system continues to add single instances of each component until it reaches the first ambiguous line in the product structure, which is *:n mat*.

The system must find out how many instances of the component *mat* to add to the product model before it can proceed to the next line of the product structure. Therefore, it looks to see if there is a rule which will tell it what to do. In fact, there is a rule *foundation nr_mat* which says if the *foundation f_type* (the foundation type) equals *mat*, the number of *mat* is 1, otherwise the number is 0. Thus, in order to get the value for this *nr_mats*, the system must have the value for *foundation f_type*. The system thus looks for a rule to give the value of *foundation f_type*. There is a rule which says that *foundation f_type* is the minimum cost value from the list of *allowable_types_and_cost* under *foundation_evaluation*. Now, the system goes to find a rule for *foundation_evaluation allowable_types_and_cost*.

The backwards chaining process continues until the small details, such as the *water_table depth* and *building bay_length*, needed to determine the upperlevel choices are reached and instantiated. Then, the rules chain back up to the top level of the system, and the original value which prompted the work of the system, which was in this example *nr_mat*, is determined. The system continues through the product structure, forcing the rules in the components to fire, thereby causing calculation of their values, until the product structure is completely instantiated.

It is interesting to note that the system only fires rules which are necessary for the completion of the product model. If the system does not need to have a value for some component, it is left valueless. When the system is done running through the product structure, a product model tree like the one in the Appendix, Figure A4, will be displayed on the screen showing the completely constructed building. The Autocad 3-D model and report are easily produced at that time (see section 4.8). Not all applications using Design++ run in this exact manner. Some require the user to ask for one value within the product model in order to start the required calculations, rather than the method used here which is to force all needed calculations when the product model is being created. However, both methods use the backward chaining method described.

4.7. Dialog Boxes

Design++ can be used to provide a *what if* scenario. Once the system is run, the user can open a *dialog box* for any component in the product model. See the Appendix, Figure A5, for a sample dialog box. The user can alter any value manually, and that new information will propagate through the system, changing any values which were calculated using the adjusted attribute. The user can see the results of the change within minutes. For example, in one run of the system, we switched the *feel* of the *top_soil* from *granular* to *crumbly*. That change caused the *top_soil material* to change from *sand* to *hard_clay*, thereby decreasing the *bearing_capacity* of the *top_soil*. Since the bearing capacity was less, the spread footings had to be bigger in order to support the weight of the building. The greater spread footings were more expensive. Thus, a different foundation, steel piles, replaced spread footing as the chosen foundation because they were now less expensive than the bigger spread footings.

4.8. Other Outputs from Design++

Other than the product model itself, Design++ provides a report facility and a connection to Autocad and Oracle, all of which can be used for output. For example, by producing a report, producing an Autocad drawing, and by storing calculated values in a database, respectively. The Conceptimator utilizes three of these four facilities for output. First, the system outputs a product model for the designed building, which shows the selected foundation. Second, the system generates an Autocad 3-D model of the foundation, which can be viewed from any angle. See the Appendix, Figure A6, for an example Autocad drawing. Third, the report summarizes the inputs to the system, the allowable foundation types and cost for each one, and gives a detailed description of geometry and cost of the chosen foundation. Although The Conceptimator uses the database to get inputs to the system, it does not output any values to the database.

5. Project Results

The Conceptimator shows that we can gather conceptual design knowledge about building foundations and implement the knowledge in a computer system in order to expedite the conceptual design and estimating process. The Conceptimator is a working prototype system; it produces conceptual cost estimates by simulating the design process at a stage in the project when the data to do so is ambiguous or missing. It successfully takes a task which has historically been solved as a classification problem and shows that it can be solved as a synthesis problem. The synthesis solution is more complete and accurate, the rationale for the estimate is explicit and traceable, and it provides an easy means for a *what if* analysis.

An unexpected outcome of the project is a serendipitous new idea for project control: *the defending champion.* The Conceptimator's estimate can be upheld against later contractor statements of design and cost. Any more expensive design must be explained in relation to the defending champion, which is The Conceptimator's design. The Conceptimator can be overruled but serves as a defending champion in the sense that it must be successfully challenged by a new idea, on a cost versus functionality trade-off, in order to be defeated.

This project could be extended to other parts of the building. However, the next planned step is an extension of the concept to a more difficult domain: hazardous waste remediation. That project will use the form and concepts of this building project in the new domain.

References

Barrie, D. S., and Paulson, B. C. (1992). *Professional Construction Management, Third Edition.* McGraw-Hill, Inc., New York, 198-251.

Dym, C. L., and Levitt, R. E. (1991). *Knowledge-Based Systems in Engineering.* McGraw-Hill, Inc., New York.

Levitt, R. E. (1990). "CIFE Seed Research Proposal: Knowledge-Based Time and Cost Estimating of Facility Projects." Unpublished.

Logcher, R. D., et al (1978). "Costmod: An Approach to Cost Estimating for Planning and Design." *Proceeding of the Second International Symposium on Organization and Management of Construction of CIB-W65,* Israel Institute of Technology, Haifa, Israel.

Retik, A., and Warszawski, A. (1991). "Knowledge Based System for Design of Prefabricated Buildings." *Artificial Intelligence and Civil Engineering,* Civil-Comp Publications, Edinburgh, Scotland, 187-195.

"Streamlining Facility Development: Developing an Entire Facility Plan in less than a Day." (1991) SARA Systems, Inc.

Waterman, D.A. (1986). *A Guild to Expert Systems.* Addison-Wesley Publishing Company, Inc., Menlo Park, California.

7. Appendix

7.1. Figures

Figure A1: The Conceptimator Library

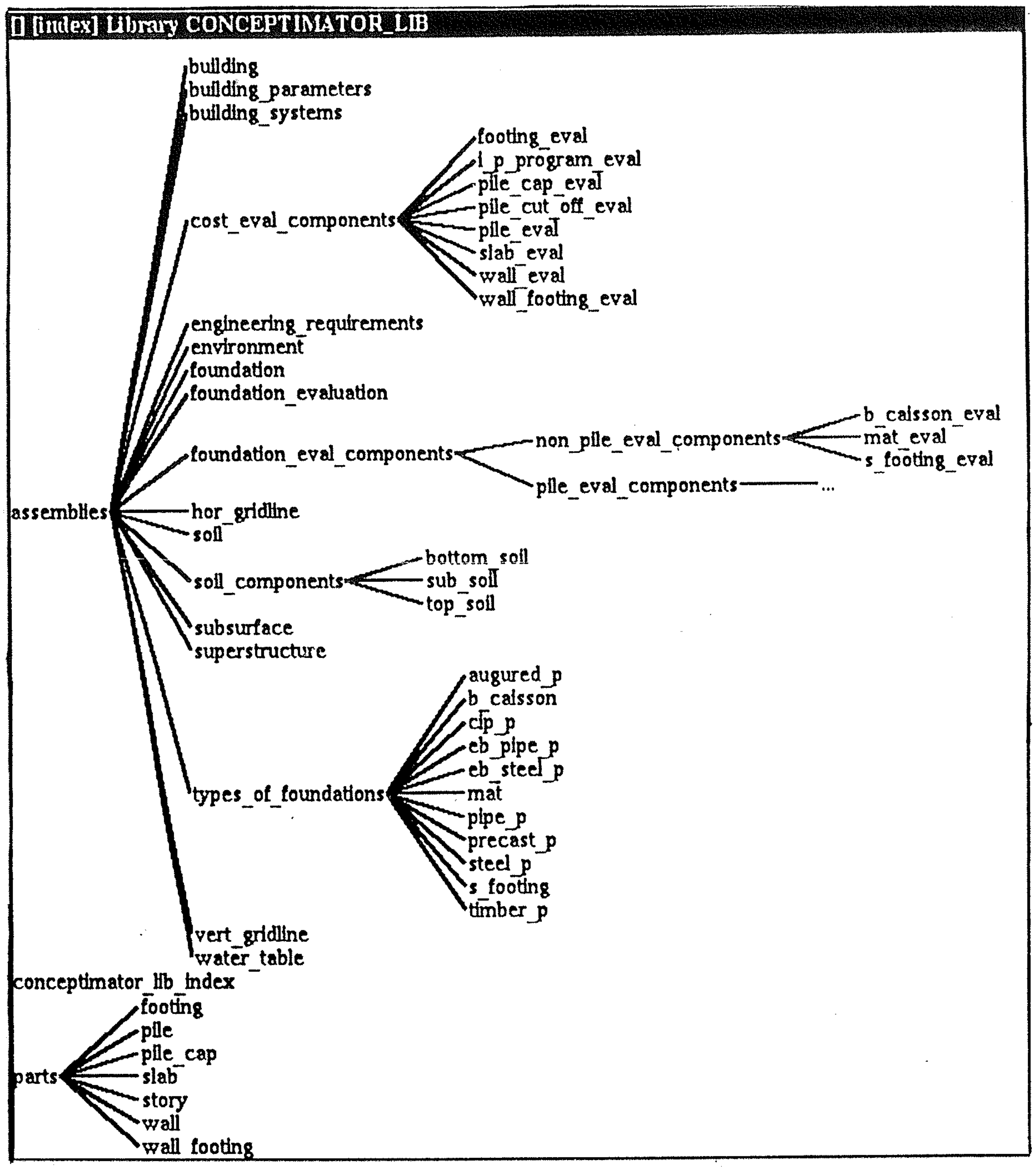
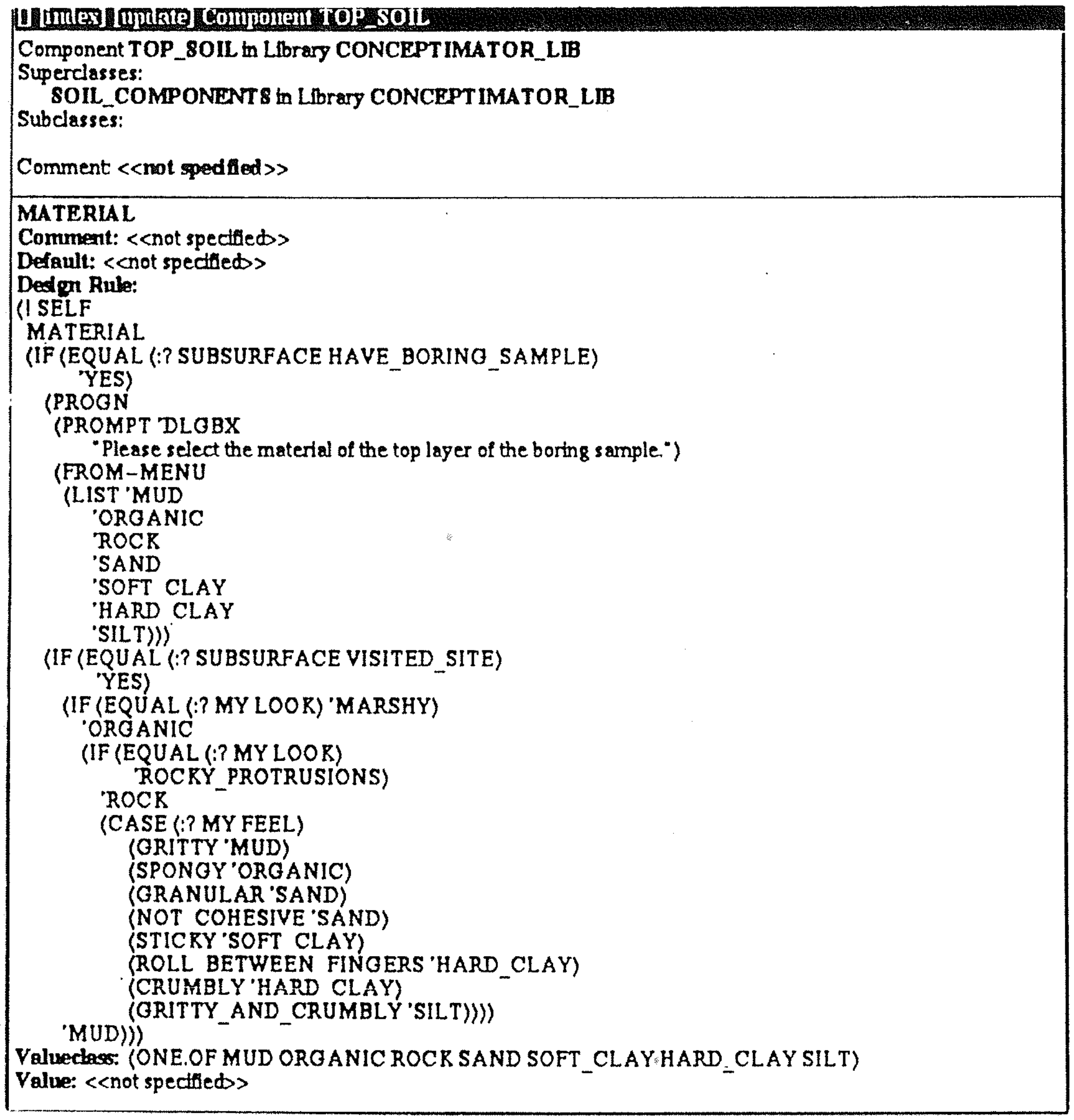


Figure A2: Example Library Component



Component **TOP_SOIL** in Library **CONCEPTIMATOR_LIB**
Superclasses:
  SOIL_COMPONENTS in Library **CONCEPTIMATOR_LIB**
Subclasses:

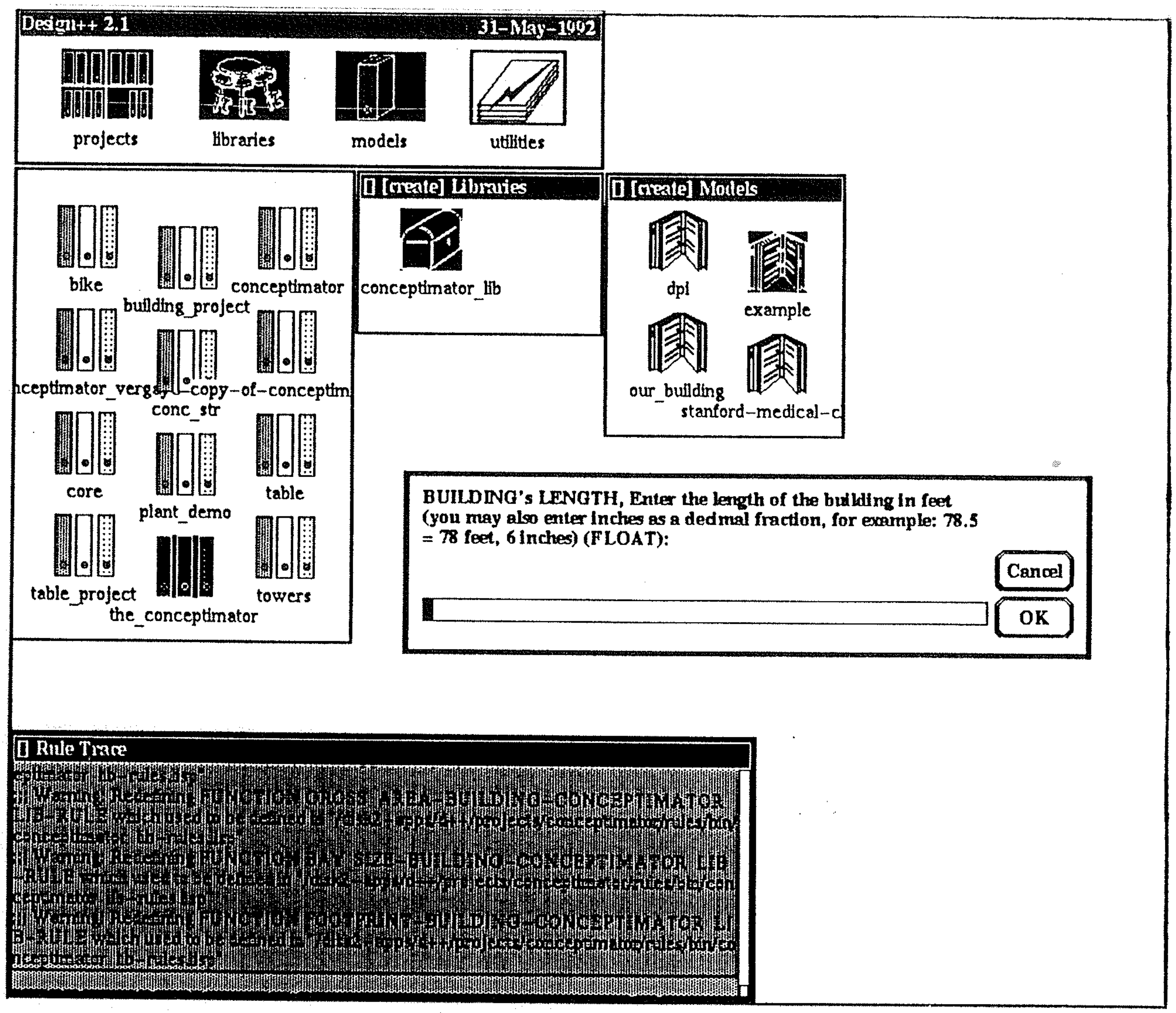
Comment: <<**not specified**>>

MATERIAL
Comment: <<not specified>>
Default: <<not specified>>
Design Rule:

```
(! SELF
 MATERIAL
 (IF (EQUAL (:? SUBSURFACE HAVE_BORING_SAMPLE)
        'YES)
   (PROGN
    (PROMPT 'DLGBX
        "Please select the material of the top layer of the boring sample.")
    (FROM-MENU
     (LIST 'MUD
        'ORGANIC
        'ROCK
        'SAND
        'SOFT_CLAY
        'HARD_CLAY
        'SILT)))
   (IF (EQUAL (:? SUBSURFACE VISITED_SITE)
        'YES)
     (IF (EQUAL (:? MY LOOK) 'MARSHY)
       'ORGANIC
       (IF (EQUAL (:? MY LOOK)
             'ROCKY_PROTRUSIONS)
         'ROCK
         (CASE (:? MY FEEL)
            (GRITTY 'MUD)
            (SPONGY 'ORGANIC)
            (GRANULAR 'SAND)
            (NOT_COHESIVE 'SAND)
            (STICKY 'SOFT_CLAY)
            (ROLL_BETWEEN_FINGERS 'HARD_CLAY)
            (CRUMBLY 'HARD_CLAY)
            (GRITTY_AND_CRUMBLY 'SILT))))
     'MUD)))
```

Valueclass: (ONE.OF MUD ORGANIC ROCK SAND SOFT_CLAY HARD_CLAY SILT)
Value: <<not specified>>

Figure A3: A View of The Conceptimator Asking for User Input



BUILDING's LENGTH, Enter the length of the building in feet (you may also enter inches as a decimal fraction, for example: 78.5 = 78 feet, 6 inches) (FLOAT):

```
[illegible] lib-rules.lisp"
;;; Warning: Redefining FUNCTION CROSS-AREA-BUILDING-CONCEPTIMATOR-LI
B-RULE which used to be defined in [illegible]
[illegible] lib-rules.lisp"
;;; Warning: Redefining FUNCTION BAY-SIZE-BUILDING-CONCEPTIMATOR-LIB
-RULE which used to be defined in [illegible]
[illegible] lib-rules.lisp"
;;; Warning: Redefining FUNCTION FOOTPRINT-BUILDING-CONCEPTIMATOR-LI
B-RULE which used to be defined in [illegible]
[illegible] lib-rules.lisp"
```

Figure A4: Example Product Model

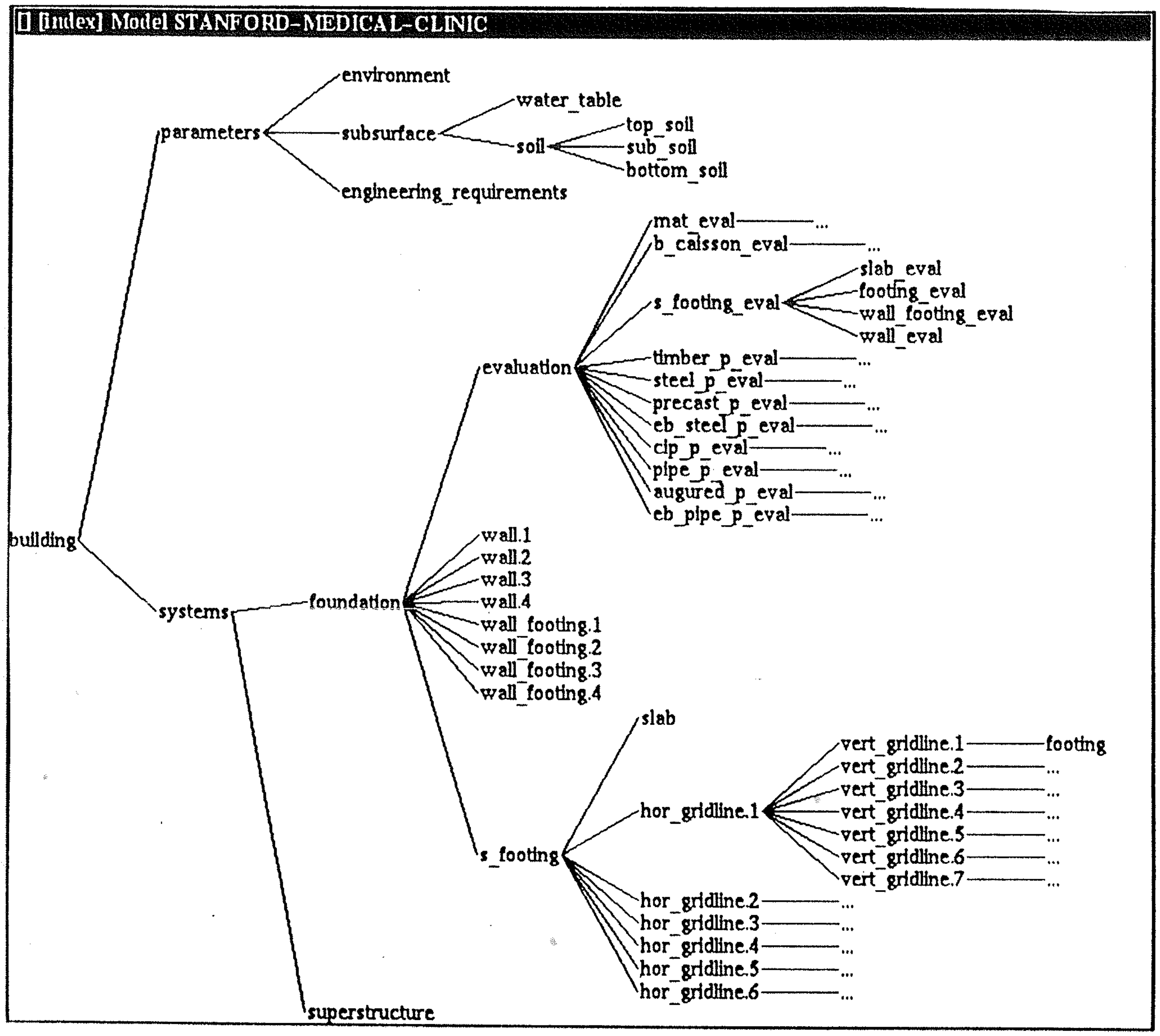
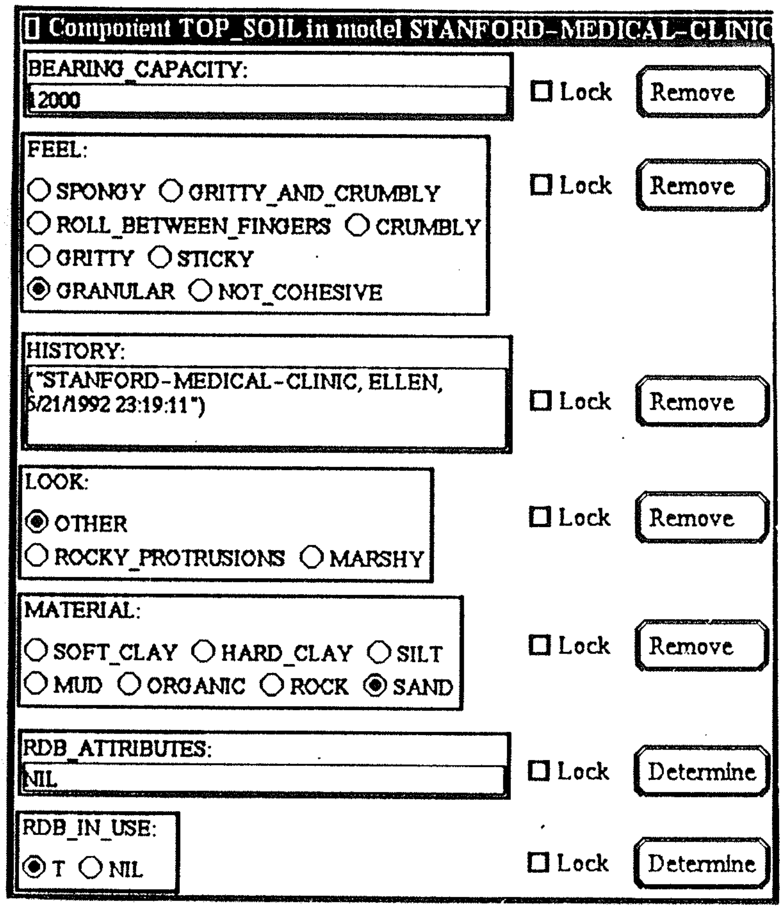


Figure A5: Example Dialog Box



Component TOP_SOIL in model STANFORD-MEDICAL-CLINIC

BEARING_CAPACITY: 12000

FEEL: SPONGY GRITTY_AND_CRUMBLY ROLL_BETWEEN_FINGERS CRUMBLY GRITTY STICKY [✓ GRANULAR] NOT_COHESIVE

HISTORY: ("STANFORD-MEDICAL-CLINIC, ELLEN, 5/21/1992 23:19:11")

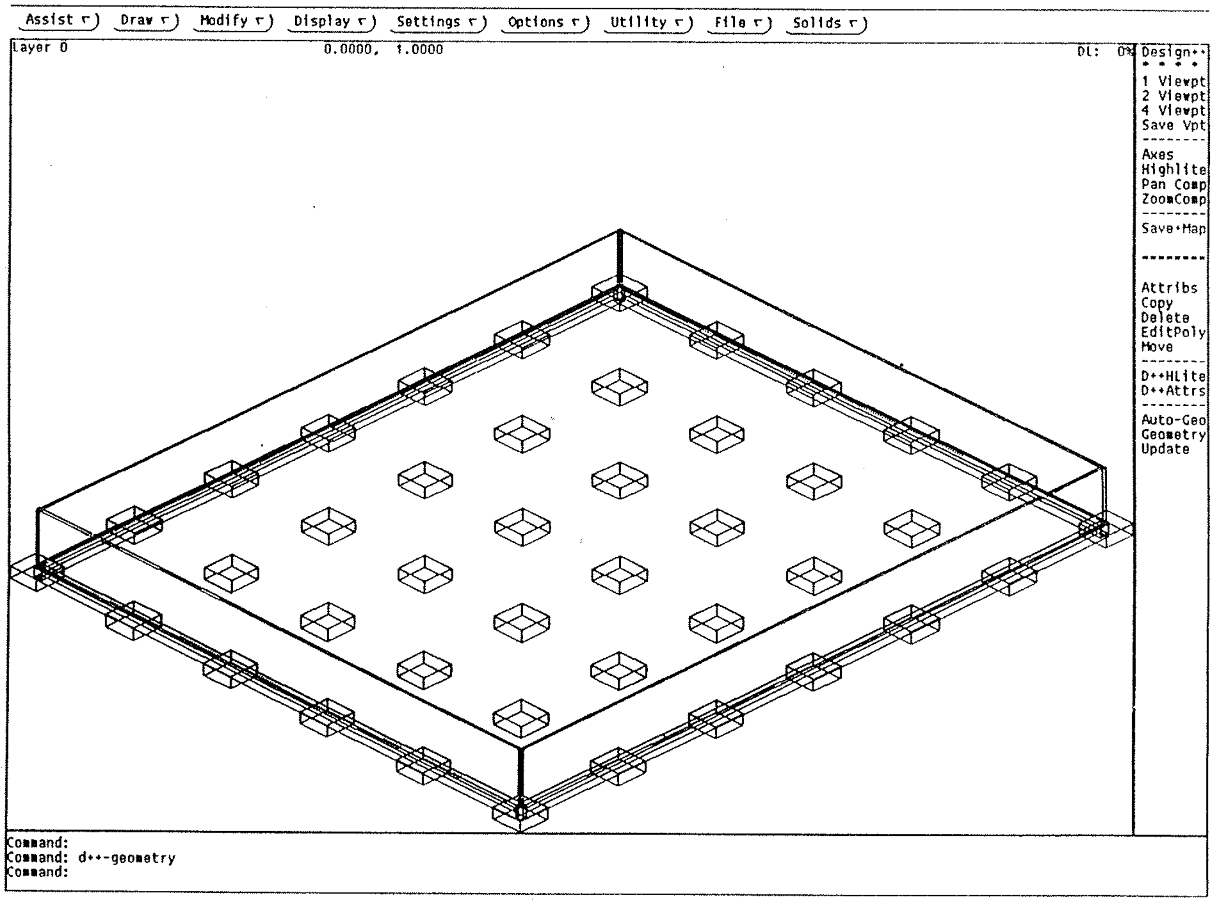
LOOK: [✓ OTHER] ROCKY_PROTRUSIONS MARSHY

MATERIAL: SOFT_CLAY HARD_CLAY SILT MUD ORGANIC ROCK [✓ SAND]

RDB_ATTRIBUTES: NIL

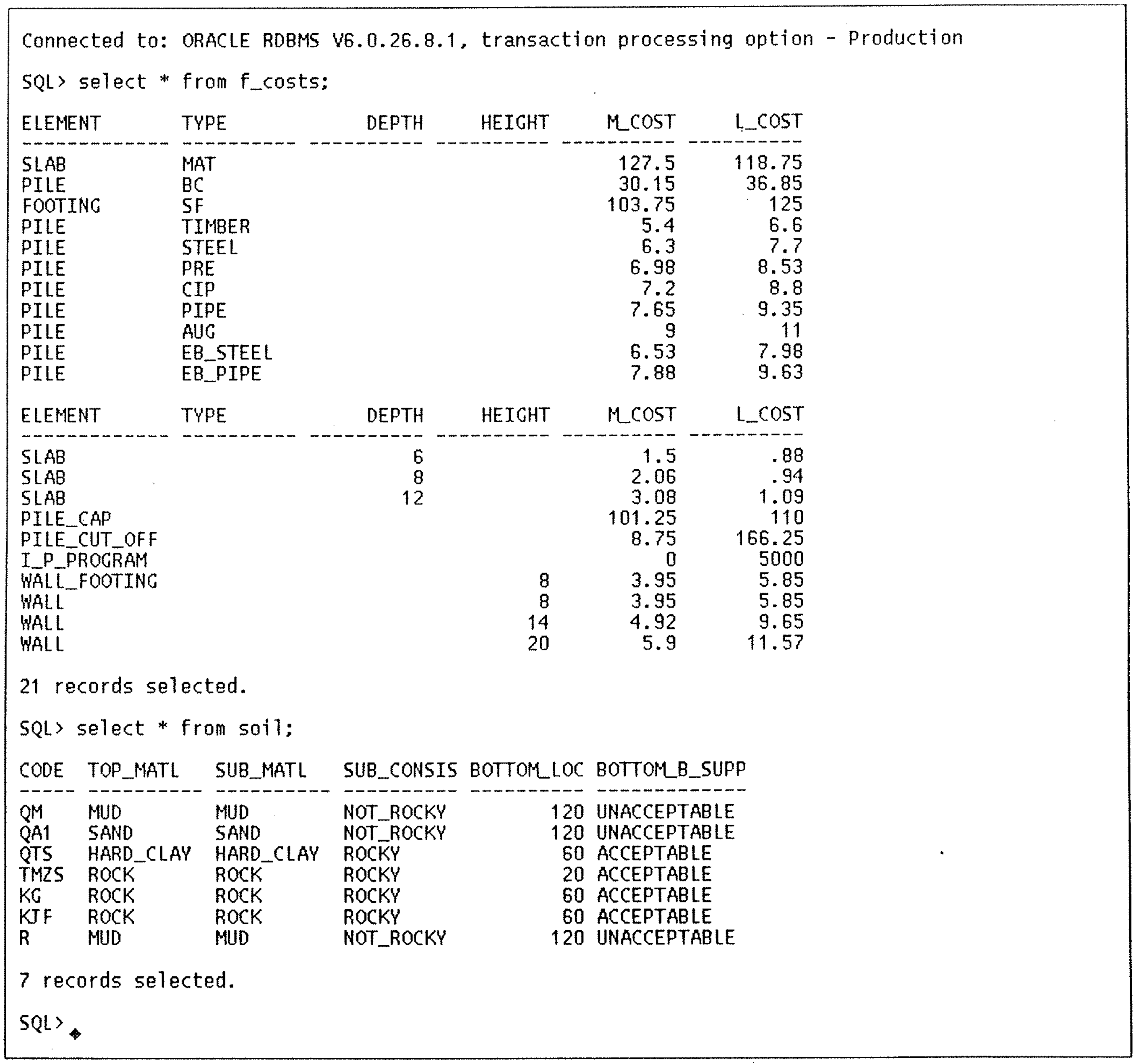
RDB_IN_USE: [✓ T] NIL

Figure A6: Example Autocad Drawing



```
Command:
Command: d++-geometry
Command:
```

Figure A7: Oracle Database Values



```
Connected to: ORACLE RDBMS V6.0.26.8.1, transaction processing option - Production

SQL> select * from f_costs;

ELEMENT       TYPE            DEPTH     HEIGHT     M_COST     L_COST
------------- ---------- ---------- ---------- ---------- ----------
SLAB          MAT                                   127.5     118.75
PILE          BC                                    30.15      36.85
FOOTING       SF                                   103.75        125
PILE          TIMBER                                  5.4        6.6
PILE          STEEL                                   6.3        7.7
PILE          PRE                                    6.98       8.53
PILE          CIP                                     7.2        8.8
PILE          PIPE                                   7.65       9.35
PILE          AUG                                       9         11
PILE          EB_STEEL                               6.53       7.98
PILE          EB_PIPE                                7.88       9.63

ELEMENT       TYPE            DEPTH     HEIGHT     M_COST     L_COST
------------- ---------- ---------- ---------- ---------- ----------
SLAB                              6                   1.5        .88
SLAB                              8                  2.06        .94
SLAB                             12                  3.08       1.09
PILE_CAP                                           101.25        110
PILE_CUT_OFF                                         8.75     166.25
I_P_PROGRAM                                             0       5000
WALL_FOOTING                                8        3.95       5.85
WALL                                        8        3.95       5.85
WALL                                       14        4.92       9.65
WALL                                       20         5.9      11.57

21 records selected.

SQL> select * from soil;

CODE  TOP_MATL   SUB_MATL   SUB_CONSIS BOTTOM_LOC BOTTOM_B_SUPP
----- ---------- ---------- ---------- ---------- -------------
QM    MUD        MUD        NOT_ROCKY         120 UNACCEPTABLE
QA1   SAND       SAND       NOT_ROCKY         120 UNACCEPTABLE
QTS   HARD_CLAY  HARD_CLAY  ROCKY              60 ACCEPTABLE
TMZS  ROCK       ROCK       ROCKY              20 ACCEPTABLE
KG    ROCK       ROCK       ROCKY              60 ACCEPTABLE
KJF   ROCK       ROCK       ROCKY              60 ACCEPTABLE
R     MUD        MUD        NOT_ROCKY         120 UNACCEPTABLE

7 records selected.

SQL>
```

7.2. Conceptimator Product Structure File

```
;;;; created by Ellen Tauber
;;;; August, 1991
;;;; modified by Gaye Oralkan
;;;; May, 1992

(building
 (building_parameters
  (environment)
  (subsurface
   (water_table)
   (soil
    (top_soil)
    (sub_soil)
    (bottom_soil)))
  (engineering_requirements))
 (building_systems
  (foundation
   (foundation_evaluation
    (mat_eval
     (slab_eval)
     (wall_footing_eval)
     (wall_eval))
    (b_caisson_eval
     (slab_eval)
     (pile_eval)
     (wall_footing_eval)
     (wall_eval))
    (s_footing_eval
     (slab_eval)
     (footing_eval)
     (wall_footing_eval)
     (wall_eval))
    (timber_p_eval
     (slab_eval)
     (pile_eval)
     (pile_cap_eval)
     (pile_cut_off_eval)
     (i_p_program_eval)
     (wall_footing_eval)
     (wall_eval))
    (steel_p_eval
     (slab_eval)
     (pile_eval)
     (pile_cap_eval)
     (pile_cut_off_eval)
     (i_p_program_eval)
     (wall_footing_eval)
     (wall_eval))
    (precast_p_eval
     (slab_eval)
     (pile_eval)
     (pile_cap_eval)
     (pile_cut_off_eval)
     (i_p_program_eval)
     (wall_footing_eval)
     (wall_eval))
    (eb_steel_p_eval
     (slab_eval)
     (pile_eval)
     (pile_cap_eval)
     (pile_cut_off_eval)
     (i_p_program_eval)
     (wall_footing_eval)
     (wall_eval))
     (cip_p_eval
      (slab_eval)
      (pile_eval)
      (pile_cap_eval)
      (pile_cut_off_eval)
      (i_p_program_eval)
      (wall_footing_eval)
      (wall_eval))
     (pipe_p_eval
      (slab_eval)
      (pile_eval)
      (pile_cap_eval)
      (pile_cut_off_eval)
      (i_p_program_eval)
      (wall_footing_eval)
      (wall_eval))
     (augured_p_eval
      (slab_eval)
      (pile_eval)
      (pile_cap_eval)
      (pile_cut_off_eval)
      (i_p_program_eval)
      (wall_footing_eval)
      (wall_eval))
     (eb_pipe_p_eval
      (slab_eval)
      (pile_eval)
      (pile_cap_eval)
      (pile_cut_off_eval)
      (i_p_program_eval)
      (wall_footing_eval)
      (wall_eval)))
   (:n wall)
   (:n wall_footing)
   (:n mat
       (slab))
   (:n b_caisson
       (slab)
       (:n hor_gridline
           (:n vert_gridline
               (pile))))
   (:n s_footing
       (slab)
       (:n hor_gridline
           (:n vert_gridline
               (footing))))
   (:n timber_p
       (slab)
       (:n hor_gridline
           (:n vert_gridline
               (pile_cap)
               (:n pile))))
   (:n steel_p
       (slab)
       (:n hor_gridline
           (:n vert_gridline
               (pile_cap)
               (:n pile))))
   (:n precast_p
       (slab)
       (:n hor_gridline
           (:n vert_gridline
               (pile_cap)
               (:n pile))))
```

```
    (:n cip_p
        (slab)
        (:n hor_gridline
            (:n vert_gridline
                (pile_cap)
                (:n pile))))
    (:n pipe_p
        (slab)
        (:n hor_gridline
            (:n vert_gridline
                (pile_cap)
                (:n pile))))
    (:n augured_p
        (slab)
        (:n hor_gridline
            (:n vert_gridline
                (pile_cap)
                (:n pile))))
    (:n eb_steel_p
        (slab)
        (:n hor_gridline
            (:n vert_gridline
                (pile_cap)
                (:n pile))))
    (:n eb_pipe_p
        (slab)
        (:n hor_gridline
            (:n vert_gridline
                (pile_cap)
                (:n pile)))))
  (superstructure)))
```

7.3. Conceptimator Rule Files

```
(in-package 'kee)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   BUILDING   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 12-Aug-91 10:16:50 by Ellen Tauber
;;;  # in square feet, currently = building length * building width but can
;;;  be entered by user directly if building is not assumed to be rectangular
(:! BUILDING FOOTPRINT
    (* (:? my length) (:? my width)))


;;;; Mon 12-Aug-91 10:30:43 by Ellen Tauber
;;;  # in square feet = bay_length * bay_width
(:! BUILDING BAY_SIZE
    (* (:? my bay_length) (:? my bay_width)))


;;;; Mon 12-Aug-91 10:34:54 by Ellen Tauber
;;;  # in feet, gross_area = #_of_stories * footprint
; gives total floor space
(:! BUILDING GROSS_AREA
    (* (:? my number_of_stories) (:? my footprint)))


;;;; Mon 12-Aug-91 10:38:58 by Ellen Tauber
;;;  # in pounds/ square feet, weight = gross_area * (dead_load + live_load)
(:! BUILDING WEIGHT
    (* (:? my gross_area)
       (+ (:? my dead_load) (:? my live_load))))


;;;; Mon 12-Aug-91 11:11:14 by Ellen Tauber
;;;  # in pounds/square feet, depends upon building_type
(:! BUILDING DEAD_LOAD
    (CASE (:? my building_type)
        (hospital 150)
        (industrial 125)
        (residential 150)
        (office 150)
        (manufacturing 135)
        (other 150)))


;;;; Mon 12-Aug-91 11:11:14 by Ellen Tauber
;;;  # in pounds/square feet, depends upon building_type
(:! BUILDING LIVE_LOAD
    (CASE (:? my building_type)
        (hospital 100)
        (industrial 125)
        (residential 50)
        (office 50)
        (manufacturing 100)
        (other 50)))


;;;; Thu 15-Aug-91 17:16:33 by Ellen Tauber
;;;  # in feet, perimeter = squareroot of footprint
(:! BUILDING PERIMETER
    (+
     (* 2 (:? building width))
     (* 2 (:? building length))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;   ENVIRONMENT RULES   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 12-Aug-91 11:55:34 by Ellen Tauber
(:! ENVIRONMENT NOISE_LEVEL
    (if
     (or
      (equal (:? my community_scrutiny) 'strict)
      (and
       (equal (:? my type_of_area) 'commercial)
       (equal (:? my adjacent_structures_distance) 'close))
      (and
       (equal (:? my community_scrutiny) 'average)
       (equal (:? my type_of_area) 'residential)
       (equal (:? my adjacent_structures_distance) 'close)))
     'unacceptable
     'acceptable))


;;;; Mon 12-Aug-91 12:36:13 by Ellen Tauber
(:! ENVIRONMENT VIBRATION_LEVEL
    (if
     (or
      (and
       (equal (:? my adjacent_structures_distance) 'close)
       (or
        (or_equal (:? my type_of_area) '(commercial residential))
        (equal (:? my adjacent_structures_strength) 'weak)))
      (and
       (equal (:? my adjacent_structures_strength) 'weak)
       (equal (:? my adjacent_structures_distance) 'moderate)))
     'unacceptable
     'acceptable))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;   ENGINEERING_REQUIREMENTS   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 12-Aug-91 14:45:05 by Ellen Tauber
(:! ENGINEERING_REQUIREMENTS BENDING_RESISTANCE_REQUIREMENT
    (if
     (> (:? building number_of_stories) 5)
     'high
     'low))


;;;; Mon 12-Aug-91 02:44:00 by Ellen Tauber
(:! ENGINEERING_REQUIREMENTS DIFFERENTIAL_SETTLEMENT
    (if
     (or
      (equal (:? my settlement_in_30_feet) 'strict)
      (equal (:? my column_settlement) 'strict))
     'strict
     'standard))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  SOIL  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 12-Aug-91 15:32:44 by Ellen Tauber
;;;  # in lbs/square feet; depends upon soil's material
(:! TOP_SOIL BEARING_CAPACITY
    (case (:? my material)
      (mud 100)
      (organic 2000)
      (rock 50000)
      (sand 12000)
      (soft_clay 1000)
      (hard_clay 8000)
      (silt 6000)))


;;;; Thu 15-Aug-91 10:58:50 by Ellen Tauber
(:! TOP_SOIL MATERIAL
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (progn
       (prompt 'dlgbx "Please select the material of the top layer of the
                boring sample.")
       (from-menu (list 'mud 'organic 'rock 'sand 'soft_clay 'hard_clay 'silt)))
     (if
      (equal (:? subsurface visited_site) 'yes)
      (if
       (equal (:? my look) 'marshy)
       'organic
       (if
        (equal (:? my look) 'rocky_protrusions)
        'rock
        (case (:? my feel)
          (gritty 'mud)
          (spongy 'organic)
          (granular 'sand)
          (not_cohesive 'sand)
          (sticky 'soft_clay)
          (roll_between_fingers 'hard_clay)
          (crumbly 'hard_clay)
          (gritty_and_crumbly 'silt))))
      (intern
       (car
        (comr::flatten
         (rdb-sql
          (format nil
                  "SELECT TOP_MATL FROM SOIL WHERE CODE = '~A';"
                  (:? subsurface map_code_value)) 'ELLEN))) 'kee))))


;;;; Thu 15-Aug-91 10:58:50 by Ellen Tauber
(:! SUB_SOIL MATERIAL
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (progn
       (prompt 'dlgbx "Please select the material of the majority of the middle
                       layers of the boring sample.")
       (from-menu (list 'mud 'organic 'rock 'sand 'soft_clay 'hard_clay 'silt)))
     (if
      (and
       (equal (:? subsurface visited_site) 'yes)
```

```
         (equal (:? subsurface existing_cut) 'yes))
        (if
         (equal (:? my look) 'marshy)
         'organic
         (if
          (equal (:? my look) 'rocky_protrusions)
          'rock
          (case (:? my feel)
            (gritty 'mud)
            (spongy 'organic)
            (granular 'sand)
            (not_cohesive 'sand)
            (sticky 'soft_clay)
            (roll_between_fingers 'hard_clay)
            (crumbly 'hard_clay)
            (gritty_and_crumbly 'silt))))
        (intern
         (car
          (comr::flatten
           (rdb-sql
            (format nil
                    "SELECT SUB_MATL FROM SOIL WHERE CODE = '~A';"
                    (:? subsurface map_code_value)) 'ELLEN))) 'kee))))


;;;; Mon 12-Aug-91 15:59:01 by Ellen Tauber
;;;  depends upon soil's material
(:! SUB_SOIL BEARING_CONDITION
    (case (:? my material)
      (mud 'compressible)
      (organic 'compressible)
      (rock 'non_compressible)
      (sand 'non_compressible)
      (soft_clay 'compressible)
      (hard_clay 'non_compressible)
      (silt 'non_compressible)))


;;;; Mon 12-Aug-91 16:01:46 by Ellen Tauber
;;;  measure of amount of shear friction available to support piles
(:! SUB_SOIL CONDITION
    (case (:? my material)
      (mud 'unacceptable)
      (organic 'unacceptable)
      (rock 'unacceptable)
      (sand 'acceptable)
      (soft_clay 'unacceptable)
      (hard_clay 'acceptable)
      (silt 'acceptable)))


;;;; Mon 12-Aug-91 16:05:31 by Ellen Tauber
(:! SUB_SOIL CAVE_IN
    (if
     (or
      (equal (:? water_table depth) 'high)
      (equal (:? water_table depth) 'very_high)
      (equal (:? my material) 'mud)
      (equal (:? my material) 'organic))
     'true
     'false))


;;;; Thu 15-Aug-91 11:36:27 by Ellen Tauber
```

```
;;;  % of rocks/boulders/etc. that may damage a pile
(:! SUB_SOIL CONSISTENCY
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (progn
       (prompt 'dlgbx  "What is the overall consistency of the boring sample
               (were rocks and boulders found in the excavation)? YOU MAY CHOOSE
                FROM THE FOLLOWING: rocky, not_rocky")
       (from-menu (list 'rocky 'not_rocky)))
     (if
      (and
       (equal (:? subsurface visited_site) 'yes)
       (equal (:? subsurface existing_cut) 'yes))
      (progn
        (prompt 'dlgbx "From the existing cut you saw during the site visit,
                judge the consistency of the majority of the middle layers? YOU MAY
                 CHOOSE FROM THE FOLLOWING: rocky, not_rocky")
        (from-menu (list 'rocky 'not_rocky)))
      (intern
       (car
        (comr::flatten
         (rdb-sql
          (format nil
                  "SELECT SUB_CONSIST FROM SOIL WHERE CODE = '~A';"
                  (:? subsurface map_code_value)) 'ELLEN))) 'kee))))



;;;; Thu 15-Aug-91 11:53:08 by Ellen Tauber
;;;  indicates either the location of bedrock or the stopping point of the boring
;;;   sample if bedrock was not reached
(:! BOTTOM_SOIL LOCATION
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (prompt-and-read 'dlgbx "Enter the depth of the boring (point at which refusal
                      was met or excavation was stopped. ENTER A NUMBER IN FEET AND
                      DECIMAL INCHES. (78.5 = 78feet 6inches)")
     (if
      (equal (:? subsurface visited_site) 'yes)
      (case (:? subsurface rocky_protrusions_or_hills)
        (rocky_protrusions 20)     ; location_range = high
        (hills 50)                 ; location_range = medium
        (neither 100))             ; later go to table for neither
      (car
       (comr::flatten
        (rdb-sql
         (format nil
                 "SELECT BOTTOM_LOC FROM SOIL WHERE CODE = '~A';"
                 (:? subsurface map_code_value)) 'ELLEN))))))


;;;; Thu 15-Aug-91 12:07:30 by Ellen Tauber
(:! BOTTOM_SOIL BEARING_SUPPORT_FOR_PILES
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (if
      (and
       (equal
        (progn
          (prompt 'dlgbx "What is the bottom layer of the boring sample? YOU MAY
                  CHOOSE FROM THE FOLLOWING: rock, hard_clay, other")
          (from-menu (list 'rock 'hard_clay 'other)))
        (or
         'rock
         'hard_clay))
        (not
         (< (:? my location) 100)))
       'acceptable
       'unacceptable)
      (if
       (equal (:? subsurface visited_site) 'yes)
       (if
        (or
         (equal (:? subsurface rocky_protrusions_or_hills) 'hills)
         (equal (:? subsurface rocky_protrusions_or_hills) 'rocky_protrusions))
        'acceptable
        'unacceptable)
       (intern
        (car
         (comr::flatten
          (rdb-sql
           (format nil
                   "SELECT BOTTOM_B_SUPPORT FROM SOIL WHERE CODE = '~A';"
                   (:? subsurface map_code_value)) 'ELLEN))) 'kee))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   WATER_TABLE   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Thu 15-Aug-91 16:21:16 by Ellen Tauber
(:! WATER_TABLE DEPTH
    (if
     (equal (:? subsurface have_boring_sample) 'yes)
     (progn
       (prompt 'dlgbx "At what depth was ground water encountered in the boring
               hole? YOU MAY CHOOSE FROM THE FOLLOWING: very_high (<=0ft), high
               (0-3ft), medium (3-22ft), low (>22ft)")
       (from-menu (list 'very_high 'high 'medium 'low)))
     (if
      (equal (:? subsurface visited_site) 'yes)
      (case
          (progn
            (prompt 'dlgbx "What does the surface look like regarding wetness? YOU
                    MAY CHOOSE FROM THE FOLLOWING: underwater, marshy, other")
            (from-menu (list 'underwater 'marshy 'other)))
          (underwater 'very_high)
          (marshy 'very_high)
          (other 'low))
      (case (:? subsurface map_color)
        (white 'very_high)
        (blue 'high)
        (green 'medium)
        (red 'low)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   END OF FILE   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(in-package 'kee)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;   FOUNDATION   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 9-Sept-91 23:32:09 by Ellen Tauber
;;; chooses the foundation with the lowest cost from allowable_types_and_cost
(:! FOUNDATION F_TYPE
    (let* ((ty (min-type-from-list-of-lists
                (:? foundation_evaluation allowable_types_and_cost))))
      (:? ty type)))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;   FOUNDATION_EVALUATION   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Tue 13-Aug-91 10:57:47 by Ellen Tauber
;;;  true if pile systems are acceptable based upon environmental and
;;;  soil conditions
(:! FOUNDATION_EVALUATION PILE_SYSTEM_ACCEPTABLE
    (if
     (and
      (equal (:? environment noise_level) 'acceptable)
      (equal (:? environment vibration_level) 'acceptable)
      (equal (:? sub_soil consistency) 'not_rocky)
      (not
       (equal (:? top_soil material) 'rock))
      (not
       (equal (:? sub_soil material) 'rock)))
     'true
     'false))


;;;; Tue 13-Aug-91 11:08:11 by Ellen Tauber
;;;  # in lbs; gives weight distributed to one foundation bay
(:! FOUNDATION_EVALUATION WEIGHT_TO_ONE_BAY
    (/
     (:? building weight)
     (/
      (:? building footprint)
      (:? building bay_size))))


;;;; Tue 13-Aug-91 11:11:36 by Ellen Tauber
;;;  #'s for comparison in lbs/square feet
(:! FOUNDATION_EVALUATION SPREAD_FOOTING_POSSIBLE
    (if
     (<
      (/
       (:? building weight)
       (*
        (:? building footprint)
        0.5))
      (:? top_soil bearing_capacity))
     'true
     'false))


;;;; Fri 23-Aug-91 by Ellen Tauber
```

```
(:! FOUNDATION_EVALUATION ALLOWABLE_TYPES
    (loop for f in (:parts my)
          when
          (equal (:? f allowable) 'yes)
          collect f))


;;;; Mon 9-Sept-91 18:56:03 by Ellen Tauber
;;; list of pairs of allowable types and foundation costs ((type cost)
;;; (type cost)...)
(:! FOUNDATION_EVALUATION ALLOWABLE_TYPES_AND_COST
    (loop for type in (:? my allowable_types)
          collect
          (list type (:? type foundation_total_cost))))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;    FOUNDATION_EVAL_COMPONENTS    ;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon  9-Sep-91 17:52:25 by Ellen Tauber
;;;  total cost of all material for foundation system
(:! FOUNDATION_EVAL_COMPONENTS FOUNDATION_MATERIAL_COST
    (let* ((cost_list (loop for p in (:parts my)
                            collect (:? p material_cost)))
           (result (apply '+ cost_list)))
      (/
       (round result 0.01)
       100.0)))


;;;; Mon  9-Sep-91 18:40:27 by Ellen Tauber
;;;  total cost of all labor for foundation system
(:! FOUNDATION_EVAL_COMPONENTS FOUNDATION_LABOR_COST
    (let* ((cost_list (loop for p in (:parts my)
                            collect (:? p labor_cost)))
           (result (apply '+ cost_list)))
      (/
       (round result 0.01)
       100.0)))


;;;; Wed 21-Aug-91 10:29:02 by Ellen Tauber
;;;  total cost for foundation system
(:! FOUNDATION_EVAL_COMPONENTS FOUNDATION_TOTAL_COST
    (/
     (round
      (+
       (:? my foundation_material_cost)
       (:? my foundation_labor_cost))
      0.01)
     100.0))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;    PILE_EVAL_COMPONENTS    ;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Thu 15-Aug-91 15:25:54 by Ellen Tauber
(:! PILE_EVAL_COMPONENTS PILES_ABLE_TO_SUPPORT_LOAD
    (if (or (<= (:? bottom_soil location) 20)
            (<= (* .5 (:? building bay_size))
                (:? pile_cap_eval area))
            (> (:? pile_eval num_piles_at_each_bay) 20))
        'false
        'true))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;   COST_EVAL_COMPONENTS  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 13:46:47 by Ellen Tauber
(:! COST_EVAL_COMPONENTS TOTAL_COST
    (+
     (:? my material_cost)
     (:? my labor_cost)))


;;;; Mon 19-Aug-91 13:47:56 by Ellen Tauber
;;;  total cost for one unit
(:! COST_EVAL_COMPONENTS UNIT_TOTAL_COST
    (+
     (:? my unit_material_cost)
     (:? my unit_labor_cost)))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   SLAB_EVAL   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 13:54:00 by Ellen Tauber
(:! SLAB_EVAL QUANTITY 1)


;;;; Mon 19-Aug-91 13:57:22 by Ellen Tauber
(:! SLAB_EVAL COST_UNIT_OF_MEASURE
    (if
     (equal (:? (:parent my) type) 'mat)
     'CY
     'SF))


;;;; Mon 19-Aug-91 13:58:54 by Ellen Tauber
;;;; # in ft2
(:! SLAB_EVAL AREA
    (:? building footprint))


;;;; Mon 19-Aug-91 14:00:56 by Ellen Tauber
;;;; returns #'s in feet
(:! SLAB_EVAL DEPTH
    (if
     (equal (:? (:parent my) type) 'mat)
     (if
      (>
       (/ (:? building bay_length) 16)
       (/ (:? building bay_width) 16))
      (/ (:? building bay_length) 16)
      (/ (:? building bay_width) 16))  ;want greater of length or width
     (case (:? building building_type)
       (manufacturing (/ 8 12))  ; 8 inches
       (industrial 1)            ; 12 inches
       (t .5))))                 ; 6 inches


;;;; Mon 19-Aug-91 14:17:41 by Ellen Tauber
;;;; # in ft3
(:! SLAB_EVAL VOLUME
    (*
     (:? my area)
     (:? my depth)))


;;;; Mon 19-Aug-91 14:19:58 by Ellen Tauber
;;;; returns # in CY or SF
(:! SLAB_EVAL TOTAL_MASS_FOR_COST
    (if
      (equal (:? (:parent my) type) 'mat)
      (/ (:? my volume) 27)       ; returns # in CY
      (:? my area)))              ; returns # in SF


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! SLAB_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! SLAB_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! SLAB_EVAL UNIT_MATERIAL_COST
    (if
     (equal (:? (:parent my) type) 'mat)
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT M_COST FROM F_COSTS WHERE ELEMENT = 'SLAB' AND
                TYPE = 'MAT';") 'ELLEN)))
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT M_COST FROM F_COSTS WHERE ELEMENT = 'SLAB' AND
                DEPTH = '~A';"
                (* 12 (:? my depth))) 'ELLEN)))))
```

```
;;;; Wed 11-Sep-91 by Ellen Tauber
(:! SLAB_EVAL UNIT_LABOR_COST
    (if
     (equal (:? (:parent my) type) 'mat)
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT L_COST FROM F_COSTS WHERE ELEMENT = 'SLAB' AND
                TYPE = 'MAT';") 'ELLEN)))
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT L_COST FROM F_COSTS WHERE ELEMENT = 'SLAB' AND
                DEPTH = '~A';"
                (* 12 (:? my depth))) 'ELLEN)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;    FOOTING_EVAL   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 14:40:07 by Ellen Tauber
(:! FOOTING_EVAL QUANTITY
    (greater
     (round
     (*
      (+ (round
       (/
        (:? building length)
        (:? building bay_length))) 1)
      (+ (round
       (/
        (:? building width)
        (:? building bay_width))) 1)))
     1))


;;;; Mon 19-Aug-91 14:43:46 by Ellen Tauber
;;;; # in ft2
(:! FOOTING_EVAL AREA
    (/
     (/
      (:? building weight)
      (/
       (:? building footprint)
       (:? building bay_size)))
     (*
      .8
      (:? top_soil bearing_capacity))))


;;;; Mon 19-Aug-91 14:44:13 by Ellen Tauber
;;;; #s in feet
(:! FOOTING_EVAL DEPTH
    (case (:? top_soil material)
      (rock 3)
      (hard_clay 4)
      (sand 2.5)
      (silt 3)
      (soft_clay 2.5)
```

```
        (mud 2)
        (organic 2)))


;;;; Mon 19-Aug-91 14:44:43 by Ellen Tauber
;;;; # in ft3
(:! FOOTING_EVAL VOLUME
     (*
      (:? my area)
      (:? my depth)))


;;;; Mon 19-Aug-91 14:45:16 by Ellen Tauber
;;;; takes #s in ft, returns CY
(:! FOOTING_EVAL TOTAL_MASS_FOR_COST
     (/
      (*
       (:? my quantity)
       (:? my volume))
      27))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! FOOTING_EVAL MATERIAL_COST
     (/
      (round
       (*
        (:? my unit_material_cost)
        (:? my total_mass_for_cost))
       0.01)
      100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! FOOTING_EVAL LABOR_COST
     (/
      (round
       (*
        (:? my unit_labor_cost)
        (:? my total_mass_for_cost))
       0.01)
      100.0))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;     PILE_EVAL    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 14:56:38 by Ellen Tauber
;;;; ONLY APPLIES TO: timber, steel, pre, cip, pipe
;;;; AND for sub_soil material of soft_clay, mud, organic, or hard_clay
(:! PILE_EVAL TOTAL_DEPTH
    (if
     (and
      (or_equal (:? (:parent my) type) '(timber pre cip pipe))
      (or_equal (:? sub_soil material) '(soft_clay mud organic hard_clay)))
     (if
      (equal (:? sub_soil material) 'hard_clay)
      (* 0.00047 (:? foundation_evaluation weight_to_one_bay))
      (* 0.00624 (:? foundation_evaluation weight_to_one_bay)))
     (if
      (and
       (equal (:? (:parent my) type) 'steel)
       (or_equal (:? sub_soil material) '(soft_clay mud organic hard_clay)))
      (if
       (equal (:? sub_soil material) 'hard_clay)
       (* 0.0000832 (:? foundation_evaluation weight_to_one_bay))
       (* 0.000346 (:? foundation_evaluation weight_to_one_bay))))))


;;;; Mon 19-Aug-91 14:57:59 by Ellen Tauber
;;;  divisor used in rule for num_piles_at_each_bay
(:! PILE_EVAL TD_DIV
    (lesser
     (:? bottom_soil location)
     (case (:? (:parent my) type)
                (timber 45)
                (steel 60)
                (pre 40)
                (cip 65)
                (pipe 70))))


;;;; Mon 19-Aug-91 14:58:20 by Ellen Tauber
;;;  divisor used in rule for num_piles_at_each_bay
(:! PILE_EVAL WT1B_DIV
    (case (:? (:parent my) type)
      (timber 135800)
      (steel 264000)
      (pre 186500)
      (cip 135800)
      (pipe 565000)
      (aug 112000)
      (eb_steel 264000)
      (eb_pipe 565000)))


;;;; Mon 19-Aug-91 14:58:53 by Ellen Tauber
(:! PILE_EVAL NUM_PILES_AT_EACH_BAY
    (greater
     (if
     (equal (:? (:parent my) type) 'bc)
     (*
      (/ (:? building length) (:? building bay_length))
      (/ (:? building width) (:? building bay_width)))
     (if
      (or_equal (:? (:parent my) type) '(aug eb_steel eb_pipe))
      (/ (:? foundation_evaluation weight_to_one_bay) (:? my WT1B_div))
      (if
       (or_equal (:? sub_soil material) '(soft_clay mud organic hard_clay))
       (lesser
        (/ (:? my total_depth) (:? bottom_soil location))
        (/ (:? my total_depth) (:? my td_div)))
       (/ (:? foundation_evaluation weight_to_one_bay) (:? my WT1B_div)))))
     1))


;;;; Mon 19-Aug-91 14:55:30 by Ellen Tauber
(:! PILE_EVAL NUM_BAYS
    (greater
     (round
     (*
      (+ (round
          (/
           (:? building length)
           (:? building bay_length))) 1)
       (+ (round
           (/
            (:? building width)
            (:? building bay_width))) 1)))
      1))


;;;; Mon 19-Aug-91 14:55:01 by Ellen Tauber
(:! PILE_EVAL QUANTITY
     (round
      (*
       (:? my num_piles_at_each_bay)
       (:? my num_bays))))


;;;; Mon 19-Aug-91 14:56:14 by Ellen Tauber
(:! PILE_EVAL DEPTH
     (if
      (or_equal (:? (:parent my) type) '(bc aug eb_steel eb_pipe))
      (:? bottom_soil location)
      (case (:? (:parent my) type)
        (timber (lesser (:? bottom_soil location) 45))
        (pre (lesser (:? bottom_soil location) 40))
        (pipe (lesser (:? bottom_soil location) 70))
        (t
         (if
          (and
           (equal (:? water_table depth) 'low)
           (or_equal (:? sub_soil material) '(rock sand silt))
           (or_equal (:? (:parent my) type) '(steel cip)))
          (if
           (equal (:? (:parent my) type) 'steel)
           55
           50)       ; value for cip only
          (:? my td_div))))))


;;;; Mon 19-Aug-91 14:57:28 by Ellen Tauber
(:! PILE_EVAL TOTAL_MASS_FOR_COST
     (*
      (:? my quantity)
      (:? my depth)))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_EVAL UNIT_LABOR_COST
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT L_COST FROM F_COSTS WHERE ELEMENT = 'PILE'
                AND TYPE = '~A';"
                (:? (:parent my) type)) 'STD))))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_EVAL UNIT_MATERIAL_COST
     (car
      (comr::flatten
       (rdb-sql
        (format nil
                "SELECT M_COST FROM F_COSTS WHERE ELEMENT = 'PILE'
                AND TYPE = '~A';"
                (:? (:parent my) type)) 'STD))))
```

```
;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;    PILE_CAP_EVAL    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 by Ellen Tauber
(:! PILE_CAP_EVAL QUANTITY
    (greater
     (round
      (*
       (+ (round
           (/
            (:? building length)
            (:? building bay_length))) 1)
       (+ (round
           (/
            (:? building width)
            (:? building bay_width))) 1)))
     1))


;;;; Mon 19-Aug-91 by Ellen Tauber
;;;; gives # in ft2
(:! PILE_CAP_EVAL AREA
    (*
     (:? pile_eval num_piles_at_each_bay)
     6.67))


;;;; Mon 19-Aug-91 by Ellen Tauber
;;;; gives # in ft
(:! PILE_CAP_EVAL DEPTH
    (/ 22 12))


;;;; Mon 19-Aug-91 by Ellen Tauber
;;;; gives # in ft3
(:! PILE_CAP_EVAL VOLUME
    (*
     (:? my area)
     (:? my depth)))


;;;; Mon 19-Aug-91 by Ellen Tauber
;;;; gives # in CY
(:! PILE_CAP_EVAL TOTAL_MASS_FOR_COST
    (/
     (*
      (:? my quantity)
      (:? my volume))
     27))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_CAP_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_CAP_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   PILE_CUT_OFF_EVAL  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 by Ellen Tauber
(:! PILE_CUT_OFF_EVAL QUANTITY
    (greater
     (round
      (*
       0.1
       (:? pile_eval num_piles_at_each_bay)
       (:? pile_cap_eval quantity)))
     1))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_CUT_OFF_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? my quantity))
      0.01)
     100.0))
```

```
;;;; Wed 11-Sep-91 by Ellen Tauber
(:! PILE_CUT_OFF_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my quantity))
      0.01)
     100.0))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;   I_P_PROGRAM_EVAL   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Mon 19-Aug-91 by Ellen Tauber
(:! I_P_PROGRAM_EVAL QUANTITY
    (greater               ;;;; function GREATER defined in ellen_funs.lisp
     2
     (round
      (/
      (:? building footprint)
      15000))))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! I_P_PROGRAM_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cos)
       (:? my quantity))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! I_P_PROGRAM_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my quantity))
      0.01)
     100.0))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;; WALL_FOOTING_EVAL ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Tue 20-Aug-91 12:27:09 by Ellen Tauber
(:! WALL_FOOTING_EVAL QUANTITY
    1)


;;;; Tue 20-Aug-91 12:36:18 by Ellen Tauber
;;;; # in ft2
(:! WALL_FOOTING_EVAL AREA
    (*
     (:? building perimeter)
     (:? wall_eval width)
     3))


;;;; Tue 20-Aug-91 by Ellen Tauber
;;;; # in ft
(:! WALL_FOOTING_EVAL DEPTH
    (case (:? top_soil material)
      (rock (/ 16 12))
      (hard_clay (/ 16 12))
      (sand 2.5)
      (silt 1.5)
      (soft_clay 1.5)
      (mud 2)
      (organic 2)))


;;;; Tue 20-Aug-91 by Ellen Tauber
;;;; # in ft3
(:! WALL_FOOTING_EVAL VOLUME
    (*
     (:? my depth)
     (:? my area)))


;;;; Tue 20-Aug-91 by Ellen Tauber
;;;; gives # in CY
(:! WALL_FOOTING_EVAL TOTAL_MASS_FOR_COST
    (/
     (:? my volume)
     27))

;;;; Wed 11-Sep-91 by Ellen Tauber
(:! WALL_FOOTING_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! WALL_FOOTING_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? my total_mass_for_cost))
      0.01)
     100.0))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   WALL_EVAL   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Tue 20-Aug-91 by Ellen Tauber
(:! WALL_EVAL QUANTITY
    1)
```

```
;;;; Tue 20-Aug-91 by Ellen Tauber
;;;; # in feet
(:! WALL_EVAL HEIGHT
    (:? building basement_depth))


;;;; Tue 20-Aug-91 by Ellen Tauber
;;;; # in feet
(:! WALL_EVAL WIDTH
    (if
     (<
      (:? building basement_depth)
      12)
     (/ 8 12)
     (/
      (:? building basement_depth)
      18)))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! WALL_EVAL MATERIAL_COST
    (/
     (round
      (*
       (:? my unit_material_cost)
       (:? building perimeter))
      0.01)
     100.0))


;;;; Wed 11-Sep-91 by Ellen Tauber
(:! WALL_EVAL LABOR_COST
    (/
     (round
      (*
       (:? my unit_labor_cost)
       (:? building perimeter))
      0.01)
     100.0))


;;;; Thur 12-Sep-91 by Ellen Tauber
(:! WALL_EVAL UNIT_MATERIAL_COST
    (let* ((ht (:? my height))
           (val
            (if
             (<= ht 8.0)
             8
             (if
              (<= ht 14.0)
              14
              20))))
      (car
       (comr::flatten
        (rdb-sql
         (format nil
                 "SELECT M_COST FROM F_COSTS WHERE ELEMENT = 'WALL'
                 AND HEIGHT = ~A;" val)
         'ELLEN)))))


;;;; Thur 12-Sep-91 by Ellen Tauber
(:! WALL_EVAL UNIT_LABOR_COST
    (let* ((ht (:? my height))
           (val
            (if
             (<= ht 8.0)
             8
             (if
              (<= ht 14.0)
              14
              20))))
      (car
       (comr::flatten
        (rdb-sql
         (format nil
                 "SELECT L_COST FROM F_COSTS WHERE ELEMENT = 'WALL'
                 AND HEIGHT = '~A';"
                 val)
         'ELLEN)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;   ALLOWABLE RULES   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! MAT_EVAL ALLOWABLE
    (if
     (or
      (equal (:? engineering_requirements bending_resistance_requirement) 'high)
      (equal (:? water_table depth) 'very_high))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! B_CAISSON_EVAL ALLOWABLE
    (if
     (or
      (equal (:? top_soil material) 'rock)
      (equal (:? sub_soil material) 'rock)
      (> (:? foundation_evaluation weight_to_one_bay) 2000000)
      (equal (:? bottom_soil bearing_support_for_piles) 'unacceptable))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! S_FOOTING_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'very_high)
      (equal (:? water_table depth) 'high)
      (equal (:? sub_soil bearing_condition) 'compressible)
      (equal (:? engineering_requirements bending_resistance_requirement) 'high)
      (equal (:? engineering_requirements differential_settlement) 'strict)
      (equal (:? foundation_evaluation spread_footing_possible) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! TIMBER_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'medium)
      (equal (:? water_table depth) 'low)
      (equal (:? foundation_evaluation pile_system_acceptable) 'false)
      (equal (:? my piles_able_to_support_load) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! STEEL_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'very_high)
      (equal (:? water_table depth) 'high)
      (equal (:? water_table depth) 'medium)
      (equal (:? foundation_evaluation pile_system_acceptable) 'false)
      (equal (:? my piles_able_to_support_load) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! PRECAST_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? foundation_evaluation pile_system_acceptable) 'false)
      (equal (:? my piles_able_to_support_load) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! CIP_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'very_high)
      (equal (:? water_table depth) 'high)
      (equal (:? foundation_evaluation pile_system_acceptable) 'false)
      (equal (:? my piles_able_to_support_load) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! PIPE_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'very_high)
      (equal (:? foundation_evaluation pile_system_acceptable) 'false)
      (equal (:? my piles_able_to_support_load) 'false))
     'no
     'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! AUGURED_P_EVAL ALLOWABLE
    (if
     (or
      (equal (:? water_table depth) 'very_high)
      (equal (:? water_table depth) 'high)
      (equal (:? water_table depth) 'medium)
      (equal (:? top_soil material) 'rock)
      (equal (:? sub_soil material) 'rock)
```

```
       (equal (:? sub_soil cave_in) 'true)
       (equal (:? sub_soil condition) 'unacceptable)
       (equal (:? my piles_able_to_support_load) 'false)
       (equal (:? bottom_soil bearing_support_for_piles) 'unacceptable))
      'no
      'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! EB_STEEL_P_EVAL ALLOWABLE
     (if
      (or
       (equal (:? water_table depth) 'very_high)
       (equal (:? water_table depth) 'high)
       (equal (:? water_table depth) 'medium)
       (equal (:? foundation_evaluation pile_system_acceptable) 'false)
       (equal (:? my piles_able_to_support_load) 'false)
       (equal (:? bottom_soil bearing_support_for_piles) 'unacceptable))
      'no
      'yes))


;;;; Wed 21-Aug-91 by Ellen Tauber
(:! EB_PIPE_P_EVAL ALLOWABLE
     (if
      (or
       (equal (:? water_table depth) 'very_high)
       (equal (:? foundation_evaluation pile_system_acceptable) 'false)
       (equal (:? my piles_able_to_support_load) 'false)
       (equal (:? bottom_soil bearing_support_for_piles) 'unacceptable))
      'no
      'yes))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;    FOUNDATION    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_MAT
     (if
      (equal (:? my f_type) 'mat)
      1
      0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_B_CAISSON
     (if
      (equal (:? my f_type) 'bc)
      1
      0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_S_FOOTING
     (if
      (equal (:? my f_type) 'sf)
      1
      0))
```

```
;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_TIMBER_P
    (if
     (equal (:? my f_type) 'timber)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_STEEL_P
    (if
     (equal (:? my f_type) 'steel)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_PRECAST_P
    (if
     (equal (:? my f_type) 'pre)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_CIP_P
    (if
     (equal (:? my f_type) 'cip)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_PIPE_P
    (if
     (equal (:? my f_type) 'pipe)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_AUGURED_P
    (if
     (equal (:? my f_type) 'aug)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_EB_STEEL_P
    (if
     (equal (:? my f_type) 'eb_steel)
     1
     0))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! FOUNDATION NR_EB_PIPE_P
    (if
     (equal (:? my f_type) 'eb_pipe)
     1
     0))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   NR_RULES   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! S_FOOTING NUMBER_OF_FOOTINGS
    (greater
     (round
      (*
       (+ (round
           (/
            (:? building length)
            (:? building bay_length))) 1)
       (+ (round
           (/
            (:? building width)
            (:? building bay_width))) 1)))
     1))


;;;; Fri 23-Aug-91 14:15:38 by Ellen Tauber
(:! TYPES_OF_FOUNDATIONS NUMBER_OF_PILE_CAPS
    (if
        (or_equal (:? my type) '(mat bc sf))
        0
      (greater
       (round
        (*
         (+ (round
          (/
           (:? building length)
           (:? building bay_length))) 1)
         (+ (round
         (/
          (:? building width)
          (:? building bay_width)))1)))
       1)))


;;;; Fri 23-Aug-91 by Ellen Tauber
(:! TYPES_OF_FOUNDATIONS NUMBER_OF_PILES
    (if
        (or_equal (:? my type) '(mat bc sf))
        0
      (case (:? my type)
        (timber
         (:? (:?1 pile_eval
                  (:?1 timber_p_eval
                       (:?1 foundation_evaluation foundation)))
             quantity))
        (steel
         (:? (:?1 pile_eval
                  (:?1 steel_p_eval
                       (:?1 foundation_evaluation foundation)))
             quantity))
        (pre
         (:? (:?1 pile_eval
                  (:?1 precast_p_eval
                       (:?1 foundation_evaluation foundation)))
             quantity))
        (cip
         (:? (:?1 pile_eval
                  (:?1 CIP_p_eval
                    (:?1 foundation_evaluation foundation)))
         quantity))
     (pipe
      (:? (:?1 pile_eval
               (:?1 pipe_p_eval
                    (:?1 foundation_evaluation foundation)))
         quantity))
     (aug
      (:? (:?1 pile_eval
               (:?1 augured_p_eval
                    (:?1 foundation_evaluation foundation)))
         quantity))
     (eb_steel
      (:? (:?1 pile_eval
               (:?1 eb_steel_p_eval
                    (:?1 foundation_evaluation foundation)))
         quantity))
     (eb_pipe
      (:? (:?1 pile_eval
               (:?1 eb_pipe_p_eval
                    (:?1 foundation_evaluation foundation)))
         quantity)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   RULES FOR GEOMETRY   ;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;; Tue 19-May-92 22:37:19 by Gaye Oralkan
;;;  distance of grid from building base point in x direction
(:! HOR_GRIDLINE X_LOC
   -(+
     (*
      (- (:? self ^index) 1)
      (/ (- (:? building width)
            (/ (/ (:? (:ANY WALL_FOOTING_EVAL) area)
                  (:? BUILDING perimeter))
               3))
         (- (:? (:PARENT (:ANY HOR_GRIDLINE)) NR_HOR_GRIDLINE)1)))
     (+ (first (:? BUILDING BASE_POINT))
        (/ (/ (:? (:ANY WALL_FOOTING_EVAL) area) (:? BUILDING perimeter)) 6))))


;;;; Tue 19-May-92 22:47:25 by Gaye Oralkan
;;;  distance of grid from building basepoint in y direction
(:! VERT_GRIDLINE Y_LOC
   (+ (+ (second (:? BUILDING BASE_POINT))
         (/ (/ (:? (:ANY WALL_FOOTING_EVAL) area) (:? BUILDING perimeter)) 6))
      (*
       (- (:? self ^index) 1)
       (/ (- (:? building length)
             (/ (/ (:? (:ANY WALL_FOOTING_EVAL) area)
                   (:? BUILDING perimeter))
                3))
          (- (:? (:ANY HOR_GRIDLINE) NR_VERT_GRIDLINE) 1)))))


;;;; Sun 17-May-92 18:51:56 by Gaye Oralkan
;;;  Box height (along Y-axis in basic orientation)
(:! FOOTING GEO_HEIGHT
   (sqrt (:? FOOTING_EVAL AREA)))
```

```
;;;; Sun 17-May-92 19:00:00 by Gaye Oralkan
;;;  Box length (along Z-axis in basic orientation)
(:! FOOTING GEO_LENGTH
    (:? FOOTING_EVAL DEPTH))


;;;; Sun 17-May-92 19:01:00 by GAYE
;;;  Box width (along X-axis in basic orientation)
(:! FOOTING GEO_WIDTH
    (sqrt (:? FOOTING_EVAL AREA)))


;;;; Tue 19-May-92 22:27:59 by Gaye Oralkan
;;;  adjusted bay length after the no of grids is rounded
(:! BUILDING ADJ_BAY_LENGTH
    (/ (:? BUILDING LENGTH)
       (round
        (/ (:? BUILDING LENGTH) (:? BUILDING BAY_LENGTH)))))


;;;; Tue 19-May-92 22:30:57 by Gaye Oralkan
;;;  adjusted bay width after the no of grids are rounded
(:! BUILDING ADJ_BAY_WIDTH
    (/ (:? BUILDING WIDTH)
       (round
        (/ (:? BUILDING WIDTH) (:? BUILDING BAY_WIDTH)))))


;;;; Tue 19-May-92 22:35:32 by Gaye Oralkan
;;;  no of vertical gridlines
(:! HOR_GRIDLINE  NR_VERT_GRIDLINE
    (+ (/ (:? BUILDING LENGTH)(:? BUILDING ADJ_BAY_LENGTH)) 1))


;;;; Tue 19-May-92 23:55:08 by Gaye Oralkan
;;;  number of piles at a grid point i.e., per pile cap
(:! VERT_GRIDLINE NR_PILE
    (ceiling (/
     (:? (:parent (:parent self)) NUMBER_OF_PILES)
     (:? (:parent (:parent self)) NUMBER_OF_PILE_CAPS))))


;;;; Wed 20-May-92 00:51:36 by Gaye Oralkan
;;;  no of horizontal gridlines
(:! TYPES_OF_FOUNDATIONS NR_HOR_GRIDLINE
    (+ (/ (:? BUILDING WIDTH)(:? BUILDING ADJ_BAY_WIDTH)) 1))


;;;; Wed 20-May-92 22:05:06 by Gaye Oralkan
;;;  Object location in cartesian coordinate system
(:! FOOTING GEO_LOC
    (list (- (:? (:PARENT (:PARENT self)) x_loc)
             (* .5 (:? self geo_width)))
          (- (:? (:PARENT self) y_loc)
             (* .5 (:? self geo_height)))
          (third (:? BUILDING BASE_POINT))))


;;;; Thu 21-May-92 01:00:54 by Gaye Oralkan
;;;  no of basement walls
(:! FOUNDATION NR_WALL
    (CASE (:? BUILDING BASEMENT_DEPTH)
     (0 0)
     (t 4)))


;;;; Thu 21-May-92 01:21:28 by Gaye Oralkan
;;;  Box width (along X-axis in basic orientation)
(:! WALL GEO_WIDTH
    (CASE (:? self ^index)
     (1 (:? BUILDING WIDTH))
     (2 (:? BUILDING WIDTH))
     (3 (:? (:ANY WALL_EVAL) WIDTH))
     (4 (:? (:ANY WALL_EVAL) WIDTH))))


;;;; Thu 21-May-92 01:22:46 by ELLEN
;;;  Box height (along Y-axis in basic orientation)
(:! WALL GEO_HEIGHT
    (CASE (:? self ^index)
     (3 -(:? BUILDING LENGTH))
     (4 -(:? BUILDING LENGTH))
     (1  (:? (:ANY WALL_EVAL) WIDTH))
     (2  (:? (:ANY WALL_EVAL) WIDTH))))


;;;; Thu 21-May-92 01:26:47 by ELLEN
;;;  Box length (along Z-axis in basic orientation)
(:! WALL GEO_LENGTH
     (:? BUILDING BASEMENT_DEPTH))


;;;; Thu 21-May-92 01:27:52 by ELLEN
;;;  Object location in cartesian coordinate system
(:! WALL GEO_LOC
    (CASE (:? self ^index)
          (3 (:ABOVE BUILDING BASE_POINT (:? FOOTING_EVAL DEPTH)))
          (4 (:RIGHT-OF BUILDING BASE_POINT
                        (- (:? BUILDING WIDTH)
                           (:? (:ANY WALL_EVAL) WIDTH)))
             (:ABOVE BUILDING BASE_POINT (:? FOOTING_EVAL DEPTH)))

          (2 (:BEHIND BUILDING BASE_POINT
                      (- (:? BUILDING LENGTH)
                         (:? (:ANY WALL_EVAL) WIDTH)))
             (:ABOVE BUILDING BASE_POINT (:? FOOTING_EVAL DEPTH)))

          (1 (:ABOVE BUILDING BASE_POINT (:? FOOTING_EVAL DEPTH)))))


;;;; Thu 21-May-92 01:00:54 by Gaye Oralkan
;;;  no of basement walls
(:! TYPES_OF_FOUNDATIONS NR_WALL
    (CASE (:? BUILDING BASEMENT_DEPTH)
     (0 0)
     (t 4)))


;;;; Thu 21-May-92 10:58:38 by ELLEN
;;;  Box height (along Y-axis in basic orientation)
(:! SLAB GEO_HEIGHT
    (- (:? BUILDING LENGTH)
       (* (:? (:ANY WALL_EVAL) WIDTH)
          2)))


;;;; Thu 21-May-92 11:02:26 by ELLEN
```

```
;;;  Box width (along X-axis in basic orientation)
(:! SLAB GEO_WIDTH
    (- (:? BUILDING WIDTH)
       (* (:? (:ANY WALL_EVAL) WIDTH)
          2)))

;;;; Thu 21-May-92 11:05:18 by ELLEN
;;;  Box length (along Z-axis in basic orientation)
(:! SLAB GEO_LENGTH
    (:? (:ANY SLAB_EVAL) DEPTH))


;;;; Thu 21-May-92 11:06:22 by ELLEN
;;;  Object location in cartesian coordinate system
(:! SLAB GEO_LOC
    (:RIGHT-OF BUILDING BASE_POINT
               (:? (:ANY WALL_EVAL) WIDTH))
    (:BEHIND BUILDING BASE_POINT
               (:? (:ANY WALL_EVAL) WIDTH))
    (:ABOVE BUILDING BASE_POINT (:? FOOTING_EVAL DEPTH)))


;;;; Thu 21-May-92 15:58:10 by ELLEN
;;;  Box height (along Y-axis in basic orientation)
(:! WALL_FOOTING GEO_HEIGHT
    (CASE (:? self ^index)
     (3 -(:? BUILDING LENGTH))
     (4 -(:? BUILDING LENGTH))
     (1  (* 3(:? (:ANY WALL_EVAL) WIDTH)))
     (2  (* 3(:? (:ANY WALL_EVAL) WIDTH)))))


;;;; Thu 21-May-92 15:59:20 by ELLEN
;;;  Box length (along Z-axis in basic orientation)
(:! WALL_FOOTING GEO_LENGTH
    (CASE (:? BUILDING BASEMENT_DEPTH)
     (0 1.25)
     (t (:? (:ANY WALL_FOOTING_EVAL) DEPTH))))


;;;; Thu 21-May-92 16:01:25 by ELLEN
;;;  Box width (along X-axis in basic orientation)
(:! WALL_FOOTING GEO_WIDTH
    (CASE (:? self ^index)
          (1 (:? BUILDING WIDTH))
          (2 (:? BUILDING WIDTH))
          (3 (CASE (:? BUILDING BASEMENT_DEPTH)
                   (0 2)
                   (t (* 3 (:? (:ANY WALL_EVAL) WIDTH)))))
          (4 (CASE (:? BUILDING BASEMENT_DEPTH)
                   (0 2)
                   (t (* 3 (:? (:ANY WALL_EVAL) WIDTH))))))


;;;; Thu 21-May-92 16:05:09 by ELLEN
;;;  Object location in cartesian coordinate system
(:! WALL_FOOTING GEO_LOC
    (CASE (:? self ^index)
          (1 (:FRONT-OF BUILDING BASE_POINT
                        (:? (:ANY WALL_EVAL) WIDTH)))
          (4 (:LEFT-OF BUILDING BASE_POINT
                        (:? (:ANY WALL_EVAL) WIDTH)))
          (3 (:RIGHT-OF BUILDING BASE_POINT
                        (- (:? BUILDING WIDTH)
                               (* 2(:? (:ANY WALL_EVAL) WIDTH)))))
          (2 (:BEHIND BUILDING BASE_POINT
                        (- (:? BUILDING LENGTH)
                           (* (:? (:ANY WALL_EVAL) WIDTH) 2))))))


;;;; Wed  3-Jun-92 10:52:58 by Gaye Oralkan
;;;  Object location in cartesian coordinate system
(:! PILE_CAP GEO_LOC
    (list (- (:? (:PARENT (:PARENT self)) x_loc)
             (* .5 (:? self geo_width)))
          (- (:? (:PARENT self) y_loc)
             (* .5 (:? self geo_height)))
          (third (:? BUILDING BASE_POINT))))


;;;; Wed  3-Jun-92 11:04:01 by Gaye Oralkan
;;;  Cylinder length
(:! PILE GEO_LENGTH
    (:? (:ANY pile_eval) depth))


;;;; Wed  3-Jun-92 14:23:48 by Gaye Oralkan
;;;  Box height (along Y-axis in basic orientation)
(:! PILE_CAP GEO_HEIGHT
    (sqrt (:? (:ANY PILE_CAP_EVAL) AREA)))


;;;; Wed  3-Jun-92 14:25:06 by Gaye Oralkan
;;;  Box length (along Z-axis in basic orientation)
(:! PILE_CAP GEO_LENGTH
    (:? (:ANY PILE_CAP_EVAL) depth))


;;;; Wed  3-Jun-92 14:25:59 by Gaye Oralkan
;;;  Box width (along X-axis in basic orientation)
(:! PILE_CAP GEO_WIDTH
    (sqrt (:? (:ANY PILE_CAP_EVAL) AREA)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   END OF FILE   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```