A KNOWLEDGE-BASED COST ESTIMATING SYSTEM FOR MANUFACTURED RESIDENTIAL HOUSING

by

Kishore Thirulokachandran

Project report submitted to the Faculty of the

Virginia Polytechnic Institute and State University

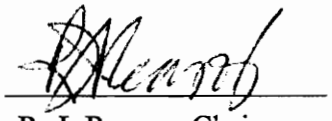
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

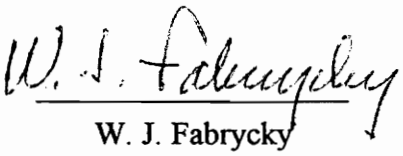
in

Industrial and Systems Engineering

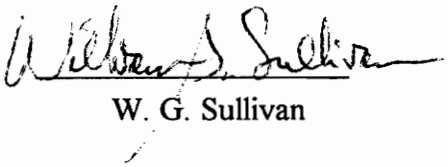
APPROVED:



R. J. Reasor, Chairman



W. J. Fabrycky



W. G. Sullivan

April 1995

Blacksburg, Virginia

c.2

A KNOWLEDGE-BASED COST ESTIMATING SYSTEM FOR MANUFACTURED RESIDENTIAL HOUSING

by

Kishore Thirulokachandran

Dr. R. J. Reasor, Chairman

Industrial and Systems Engineering

(ABSTRACT)

Cost estimation is a critical part of the manufacturing process. This is more so in the case of manufactured or "prefabricated" housing. A cost estimation approach for obtaining a quick, accurate estimate has been developed in this project. Prior to the development, literature in the areas of expert systems, construction cost estimation and artificial intelligence were read and the most relevant papers have been reviewed in the report. The estimation approach has been developed based primarily on "knowledge" derived from a firm manufacturing residential housing. It is formed based on relationships between the various sections of the building and uses formulae developed based on this "knowledge" and standard cost estimation relationships for calculating a total cost estimate. The formulae use the primary cost drivers for each section of the construction work to arrive at the estimate.

The whole process has been delivered on an expert system platform. The expert system has been developed using a windows-based expert system shell, Kappa-PC™ produced by Intellicorp™. The report contains, other than the description of the approach mentioned above, the description of the expert system which was developed, and the development methodology. A case study is also given for the purpose of a better understanding of the use of the expert system and to prove the effectiveness of the approach and formulae developed.

The complete process has been shown to have an accuracy of about 5 % based on the difference between the values calculated by a human expert and the values obtained using the formulae and procedure developed for the same input.

ACKNOWLEDGMENTS

I wish to thank my advisor, Dr. Reasor, without whom this project would not have existed. I am really grateful to him for all the advice and support he has given me during the execution of this project and its final documentation.

I also wish to thank Dr. Fabrycky and Dr. Sullivan for having been on my committee and for their suggestions with regard to the project and the documentation.

I wish to thank my good friends and roommates Munish, Jeet, Aslam, and of late, John, for having put up with me whenever I most needed it. A special thanks also to my friend Sailesh to whom I could always turn to validate my ideas about Kappa or to clear doubts about the same.

Another special thanks also to my friends Raghu Thanjavur and Ramesh, P.S. for having got me going faster than I otherwise would have with some of the work.

I also thank my other friends too numerous to mention, including Sid, Jay, Sathya, and others for their support and help throughout my stay in Virginia Tech.

A final, huge thank you, to both my parents, without whose constant support and advice I would not be here today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Cost Estimation in the Construction Industry

Effective cost estimating is one of the deciding factors in procuring any manufacturing contract. This is more so in the construction industry due to the complexities and numerous factors involved in the estimations.

Manufacturing in the construction industry has a far larger number of variables to be considered than in most other fields. A simple example would be the manufacture of prefabricated walls. Some of the main variables to be taken into consideration for the manufacturing process would be:

- Wall height;
- Wall lengths;
- Size of the studs used;
- Different wall heights for different floors, if any;
- Type of header frame assembly;
- Type and number of bottom plates;
- Type of sheathing used;
- Type of wall blocking if any; and
- Spacing of the studs;

The above are just the main variables and is by no means an exhaustive list of the variables involved for even a simple part of the building such as a prefabricated wall. This gives an idea of what makes construction estimation a complicated and time-consuming process.

In the case of large firms or federal agencies, cost estimating is normally done either by a team of cost estimation experts, or by subdividing the work and letting specialists work on each area.

In the case of small firms, the work is done by one or more cost analysts who may either hold an engineering degree in any field or may have only field experience. The steps followed in such cases are usually as follows.

- List the material requested.
- List the raw materials required for manufacture.
- Calculate the cost components.

- Add markup cost and draw up an estimate sheet.
- Depending on the requested material, a site visit may be required before estimation.

1.2 Importance of the Estimate

It is the estimate on which the procurement of the manufacturing contract rests. An exceedingly high estimate will usually result in the loss of the contract to a rival company. A miscalculated low estimate will result in a loss, or less profit, to the manufacturing company as it will have to absorb the difference between the estimate and the actual manufacturing costs. Estimation time is also of primary importance in getting the contract. Companies have on the average about one to two months to prepare a bid for the contract. In most cases the quotes are given in less than two weeks and the contract finalized in a month. The main reason for this is that in many cases the construction is already in process when the quotes are requested.

1.3 Main Problem in Preparing the Estimate

One of the main problems of estimating in the construction industry is the absence of an *universal* estimation method. Estimation techniques and construction designs vary with the concern. Every estimator uses formulae and rules specific either to him or the concern which are usually based on a combination of rules-of-thumb and standard formulae.

Another problem, especially in small firms, is that the estimator is required to be an expert in many fields. He has to have

- A strong grasp of design in order to draw up the list of raw materials required for manufacturing from the plans provided by the customer.
- An adequate knowledge of manufacturing in order to get an idea of the labor and work required.

- A good costing background to affix costs to the labor and manufacturing efforts and material and add up a profit to it.

In the case of small firms, it is unfortunately not common for the firm to have an experienced person good in all three areas of construction design, manufacture and costing. The main reason is that not too many people are well versed in *all* three fields, and the ones that are, are either not affordable or are not keen on working with small firms. One way of trying to solve this problem would be to use an expert system.

1.4 Reasons for using an Expert System (ES)

In today's world, where PCs are used to such an extent everywhere, a PC-based expert system shell may be quite conveniently used to make an *affordable* and *efficient* expert system which would give an accurate, quick appraisal for the bid based on the information provided. This system could easily supplement the design or manufacturing knowledge the estimator may be lacking, and provide a quick cost estimate. The estimator may then, if he feels the need to, improve upon the estimate as he or she wishes.

The primary advantages of using an expert system instead of, for example, a regular cost estimating or design package, may be summed up as follows:

- Expert systems can work with incomplete or uncertain information;
- An ES can, unlike common software, explain its reasoning and results. This, other than enhancing its use, helps in the development and improvement of the system;
- Other than explaining its reasoning, an ES also allows varying of only part of the previous input to produce new results;
- The ES uses knowledge *and* data;
- Execution is done using heuristics and logic instead of algorithms as in conventional software;
- Unlike conventional systems which can handle only numeric data, they can deal with both quantitative *and* qualitative data; and

- Expert systems are, unlike conventional programs, usually developed with the help of domain experts. This obviously leads to more accurate results.

  Other than the above, a few secondary advantages would be:
- They disregard irrelevant data;
- The user interface can be made simple enough for even a layperson to follow easily and at the same time detailed enough to procure all the required knowledge for the system;
- They can accept problem descriptions in lay terms;
- The knowledge base can be enlarged and improved in the form of new rules as time goes along and more knowledge is acquired;
- Changes in the rules used are easier to accomplish;
- They supplement knowledge lacking by the estimator;
- They produce quick results;
- They are reliable and can give consistent quality results; and
- They save decision time;
- In the absence of the regular estimators, they can give a result good enough to be used temporarily or as an approximate figure with the information fed in even by a layperson.

One more important advantage is that if, as is sometimes necessary, after possible site investigation, it is felt that modifications need to be made in the assumptions made by the expert systems, the changed values may be entered instead of the default or previously input values to produce the corrected estimate immediately.

1.5 Project Objective

The objective of this project has been to develop a knowledge-based approach for estimating the cost to manufacture residential housing, even in the absence of complete information. The approach has been illustrated by developing an expert system using the expert system shell, Kappa-PC™ produced by Intellicorp™. Formulae have also been

developed for the specific purpose of being used for manufactured (or prefabricated) residential housing. The expert system runs on rules and functions formed based on these formulae. Information from a real-life company has been used both during the development and during the verification and validation of the expert system to test the reliability and consistency of the results produced.

CHAPTER 2: LITERATURE REVIEW

2.1 Traditional Cost Estimation in the Construction Industry

The importance of cost estimation has been explained in the previous chapter (Section 1.2). To better understand residential project developments in the construction industry the procedure observed has been summarized below.

- *Location of a suitable site.*
- *Choosing the type and details of housing*: This is dependent on the geographical location of the site and the personal preferences of the investors.
- *Estimation of the probable project costs*: Computers have largely supplanted manual methods in this area. One such computer application is the use of cost estimating and design software. The design software is used for finding the materials required from the plans, following which the cost estimating software is used to find the costs. Some papers that were reviewed in this area have been discussed below.
- *Start construction if costs are found to be feasible.*
- *Acquisition of quotes from manufacturing concerns.*
- *Finalizing of manufacturing contract.*
- *Continuation and completion of construction.*

Al-Yousefi [1992] outlines some standards for cost estimating, and he describes techniques to help the reader budget effectively and maintain cost control. His paper also summarizes some specific practical budgeting techniques which are used by leading consultants for determining quantities and for establishing the quality of a building or a project. First, the concepts of budget estimating and cost control are defined and explained. Then the key cost drivers such as functional areas (Net sq. meters - NSM), design parameters, etc. are listed.

The owner's needs for space are programmed in terms of NSM, and the scope of construction in terms of gross square meters (GSM). The ratio of NSM/GSM is referred

to as space efficiency and varies from 0.55 to 0.90 depending on the type of space provided. After this, the building configuration has to be established. That is, decisions or assumptions regarding the number of floors, floor parameter, and floor heights have to be made. In determining the cost parameters, the most common way for estimating the cost of a new building is in terms of cost per GSM ($/GSM). The system parameters are then discussed and a section providing additional guidance on the determination of each relative system parameter in relation to both quantity and quality is given. The 12 division structures foundation, substructure, superstructure, exterior closures, roofing, interior construction, conveying systems, mechanical, electrical, general construction OH&P, equipment, and site work together are called "UNIFORMAT." Formulae are also given for each.

Finally, in conclusion, the accuracy of the system is discussed. The results of project budgets compared to actual construction bids in the USA showed an extreme deviation range of 66% and a mean deviation range of 29%. The usage of the system mentioned in the paper is alleged to improve it.

Baldwin, et. al. [1991], talks about the application of protocol analysis in the development of knowledge based systems for contractors' resource based estimating. First existing computer-aided estimating systems and their shortcomings, and the advantages that knowledge based systems have over them are discussed. These are in short a summary of the advantages of expert systems over conventional programs discussed in the previous chapter. Then protocol analysis and its use in development of expert systems is explained. Protocol analysis consists of giving a suitable subject a job that he's accustomed to doing and making him record all his actions and reasons for arriving at them. This way, the decision process is analyzed and may be duplicated in expert systems. There were a few important aspects about experiments using protocol analysis which became evident.

1. The tasks chosen for the subject to perform must be realistic, succinct tasks that are readily achievable within the time period.

2. The tasks must be undertaken on a single observer, single subject basis.
3. There is some difficulty in getting the subjects to verbalize fully their actions and thoughts.
4. The subject often feels the need to explain his/her thoughts, add additional comments on company policies or "educate" the observer on their actions.
5. It is unrealistic to expect experts to perform the task at their normal rate of working and verbalize their thoughts and actions at the same time.

The initial research has shown that protocol analysis contributes to the development of knowledge based expert systems by

- assisting in the knowledge elicitation process,
- clarifying the solution strategy, and
- highlighting the characteristics of individual experts.

The paper is ended with a note on the opinion of the authors that the technique of protocol analysis is well suited to aid the study of decision making in other tasks within the construction industry.

2.2 About Expert Systems

Ignizio [1991] has defined an expert system as "*a model and associated procedure that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert*". In simpler words, it is a computer program which operates in a specific domain, and has as a knowledge base the expertise acquired from a human expert.

Shafer, et. al. [1988], says in his article that "Expert systems are programs that respond very much like a human expert in a well-defined area (the program's domain)". The article describes a simple six step process for developing an expert system, using the programs developed by the two authors as examples. The six steps are:

1. *Defining the needs and the domain of the "Expert":* The domain of the system to be developed is decided and the scope and limits of the system are defined;

2. *Knowledge engineering: Defining the "Rules":* The expertise of the human expert is converted into computer "rules". The logic pattern which will be input into the knowledge base along with the rules is established. This is the pattern in which the rules will have to be executed, the nature and order of the questions which will have to be asked, etc.;
3. *Determining the structure:* Forward-chaining or Backward-chaining: If a rule-based expert system starts with a hypothesis and tests it, it is called "backward-chaining". One that builds toward a conclusion has a structure called "forward-chaining". The selection of the type of the reasoning to be followed is based upon the software, the application, or personal preference;
4. *Selecting the AI program:* The types of shells available can be classified as demonstration systems (meant mostly for learning about expert systems and for prototyping and testing), basic systems (meant for first-time users and for simple problems), enhanced systems (using AI languages and having built-in features useful for expert-system designers and can interact with other programs) and AI languages (programming language used to build the shell). Other than the above, other selection criteria mentioned are cost, capabilities, PC-based, etc.;
5. *Prototyping, testing, debugging:* The program having been selected, creating a prototype to test the procedure intended and debugging it to find the possible problems before making the final system is important and saves time which might be wasted if the procedure is faulty or due to bugs in the program.
6. *Implementation and training:* In the case of an expert system, training is oriented more toward acceptance of the system and the credibility of its results. It is best not to emphasize to the users that they are working with AI or an expert system, but let the usefulness and user-friendliness of the system itself win them over.

In conclusion it is noted that though many users find little conceptual difference between an expert system and a checklist, the checklist cannot handle the many different logic patterns for different conditions, nor do quantitative analysis.

Wentworth [1990] refers to knowledge-based expert systems as computer programs that include a simulation of the reasoning and problem solving processes of human experts. The purpose of this paper was to discuss some of the practical aspects of developing knowledge-based expert systems. The major differences between conventional programs and expert systems as given in the paper are in Table 2.1.

Table 2.1 Differences between Conventional programs and Knowledge-Based Expert Systems *(Wentworth [1990])*.

Conventional programs	Knowledge-based expert systems
Stated equations which can be proven. If correct numerical data is provided, a correct answer will result.	Usually based on rules of thumb (or other knowledge representation schemes) that are generally reliable, but not always correct. These are concepts and cannot be reduced to formulae or numbers.
Provides answers only.	Can explain its logic and reasoning.
Needs all data called for to operate.	Can function with incomplete data.
Often programmed in isolation from domain experts and users.	Development team includes domain experts by definition.
May be extremely difficult to examine imbedded knowledge.	Provides capability to easily examine knowledge base.

The three basic modules of an expert system architecture are stated by Wentworth to be the knowledge base (long term or static memory), inference engine (control strategies or inference mechanisms), and the short term (dynamic) memory. Three additional components are the knowledge-acquisition facility, the explanation facility, and the user interface. These modules and their relationships are represented as shown in Figure 2.1.

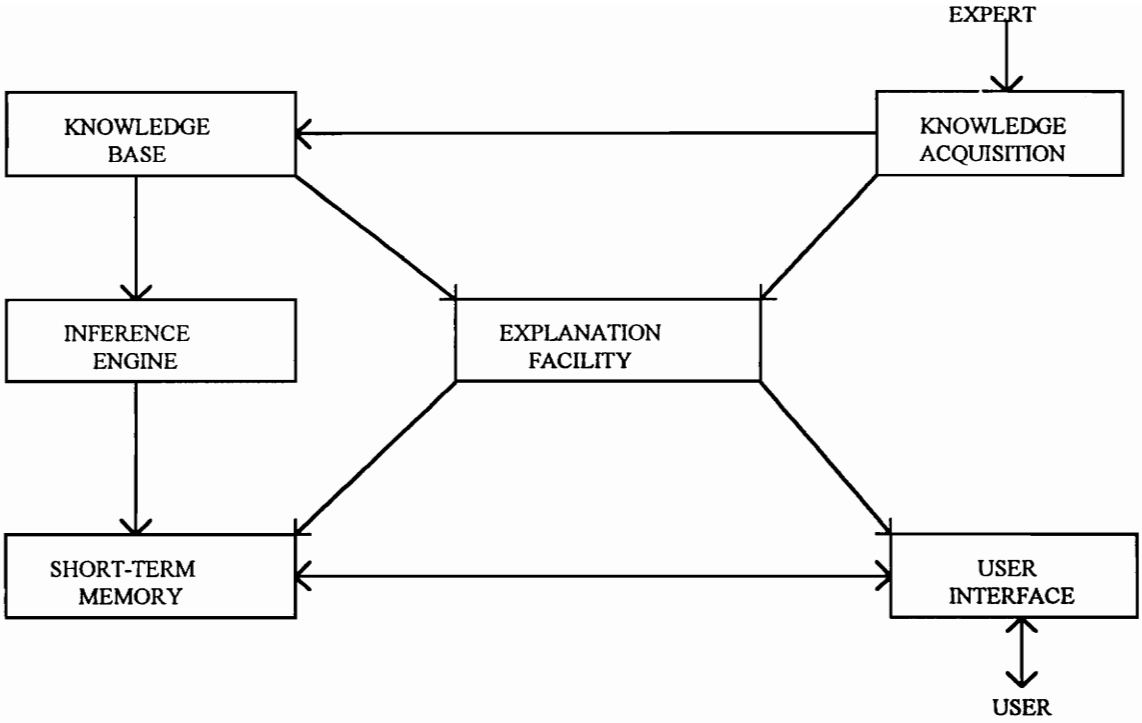


Figure 2.1 Knowledge-Based Expert System Architecture *(Wentworth [1990]).*

The different modules have been described by him in the following fashion. The knowledge base serves as a storage space for the system's domain specific knowledge such as rules or other knowledge representations (heuristics). The inference engine serves as the "brain", comprising of the control strategy or problem solving mechanism used to develop solutions to the user's problem(s). The short term memory contains the dynamic or problem specific knowledge, the user responses to questions asked by the system, and other temporary information generated by the system. The user interface provides a means for the user to communicate comfortably and efficiently with the expert system. The knowledge acquisition process is basically to build the system's knowledge base. The explanation facility gives the system the ability to explain its reasoning process and to provide definitions and other information to the user.

A list of criteria for deciding upon whether expert systems should be used, followed by a list of barriers to acceptance or reasons for failure of expert systems was also stated and has been given below. In the latter, the main reason is stated to be poor planning and execution of the development process. In identifying the need for an expert system, a structured study is recommended, including the following issues.

1. The problem/need to be addressed and the system benefits.
    - Problem clearly identified.
    - Probability of system's effectiveness in specific problems to be described and quantified.
    - Application and use of output clearly defined.
    - If standardization of results is desirable, degree to which system will improve standardization must be estimated.
    - Use of system to improve conditions by improving quality of results to be estimated.
    - Expected utility of system as training tool must be described.
    - End user involvement for duration of development must be assured.
    - Time and money savings based on projected use of system to be estimated.
2. Organizational risk factors.
    - Presence of dedicated and influential advocate for the system.
    - Management to have realistic idea of difficulties of developing the system.
    - Management support.
    - Management to have realistic idea of system performance.
    - Results of system to be applied without requiring excessive management approval.
3. Technical risk factors.
    - Availability of experts
    - Formation of development team and their dedication to the project.
    - Availability and commitment of all team members must be assured.

- Availability of manual or automated procedure to serve as a model.
- Performance of system as compared to experts.
- Interaction with other programs.

4. User risk factors.
   - Users must want the system
   - The computer proficiency and other skills of the end users must be accommodated.

Wentworth then discusses different types of problems amenable to solution by ES. These include problems of diagnosing/monitoring, interpretation/classification, prediction/forecasting, design and planning. An overview of the expert system development process and the major tasks to be performed during this process is then provided.

1. Development team: This comprises of the advocate for the system, the end users, the domain experts, and the knowledge engineer who actually builds the system.
2. Knowledge engineering: This is the phase which begins soon after the domain expert has been chosen. The job of the knowledge engineer is to take the knowledge from the expert and save, organize and imbed this knowledge in a computer program.
3. Development steps: The necessary steps are described as
   - Problem analysis: Through interviews of experts to try and determine major components of the reasoning process and attempt to identify the elements that should be included in the system.
   - Initial prototype: The knowledge developed above is implemented in the knowledge base and user interfaces, explanation facilities, etc. are developed. A demonstration is also done.
   - Expanded prototype: The initial prototype should be used extensively by experts and potential users and necessary changes incorporated in the expanded prototype. There should be at least two demonstrations at this phase.

- Once final changes are made and incorporated, the delivery system is made, optimizing it with respect to performance, memory requirements, user interface, etc.

4. User interface: The user interface is the means by which the user and the expert system communicate with each other. Some of the methods for this include menus, tables or lists, graphical reports, etc.
5. Knowledge acquisition: This may be considered as a subset of knowledge engineering, with the difference that the knowledge engineer is moved out of the picture and the experts are allowed to "educate" the system directly to aid in the development and maintenance of the system.
6. Explanation facility: This explains to the user how it reached its conclusions. It can also be used to store explanations of data input requirements, special considerations or any other information deemed useful in making the system more user friendly.
7. Representing uncertain knowledge: This is the facility by which the system can rate the certainty or accuracy of it's results for the convenience of the user. The factor usually used for this is called CF (Certainty Factor), and is rated from 0 to 10 or 0 to 100.

The verification, validation and evaluation of the system are then discussed by him. In his words, verification basically has to answer the question, "Is the system built right?". Validation is to answer the question, "Is the program doing the job it was intended to do?". Evaluation addresses the issue, "Is the system valuable?". Various methods of answering the questions were discussed.

Different software tools available are listed out as a language-tool continuum in Table 2.2.

Table 2.2 Language-Tool continuum *(Wentworth [1990]).*

HIGH-LEVEL LANGUAGES		**ENVIRONMENTS**	**GENERIC TOOLS**		**DOMAIN SPECIFIC TOOLS**
LISP	PROLOG	OPS5	KEE	EXSYS	PICON
C			ART	CRYSTAL	
PASCAL				LEVEL 5	

Finally, the distribution and maintenance of the systems were discussed. Three main criteria for facilitating distribution are:

- Identify and involve the user community before development of the system;
- Develop system using standard hardware and software;
- Use development software that does not require distribution licenses or where an unlimited distribution license can be purchased at a reasonable fee.

A few development rules for easy maintenance are:

- Design the system as straightforward and clear as possible;
- The system should be well documented;
- The system itself should have extensive "help" text and explanation text.

2.3 Artificial Intelligence for Cost Estimation by Project Developers

The use of artificial intelligence for cost estimating work has been getting a lot of interest in recent times. Some papers relevant to the use of AI systems in the construction industry are discussed below.

Watson et. al. [1992], describes the development of an expert system that produces strategic cost estimates for residential construction projects for the use of project developers. The system being talked about here, the Strategic Planning of Residential Accommodation - Budget system (SPORA - B), is one of a suite of three being developed for commercial use in Salford, UK. SPORA - B is intended for use at the feasibility stage of a residential accommodation construction project. Its purpose is to enable the reliable and accurate estimation of budget costs and the speedy evaluation of

alternative solutions. It uses a Client Centered Approach (CCA), which is a method covering the full development life cycle of an ES. It provides milestones to guide the project. These milestones (shown in figure below) refer to what the clients can see being demonstrated and not to the conventional tasks of elicitation, acquisition, and so forth.

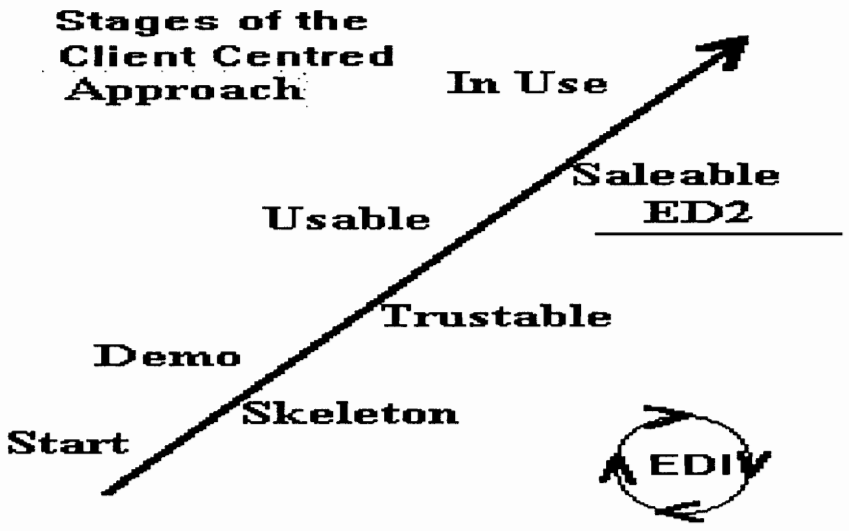


Figure 2.2 Stages of the CCA *(Watson et. al. [1992]).*

This system is being implemented using an expert system shell, Kappa-PC™ produced by Intellicorp™. It is intended to be operated by building professionals such as quantity surveyors, project managers, etc. The target user group is not assumed to be computer experts. For this reason a graphical user interface has been adopted. A basic schematic diagram of the system architecture can be seen below.

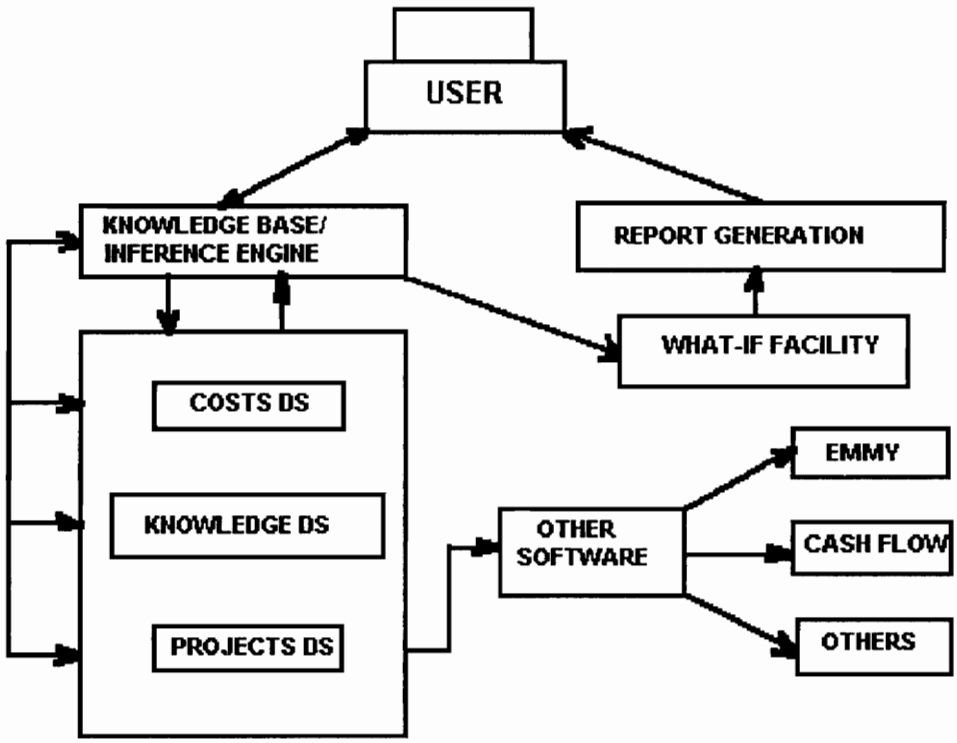


Figure 2.3 An overview of system architecture *(Watson et. al. [1992]).*

The system has three different operational phases:

1. Entry of dwelling details: In this phase, typical information asked by the system would be, other than building type, information such as site, funding, timing, specification, planning, client body, etc. The result will be a budget cost for all the dwellings on the project;
2. Entry of external works details: In this phase, the details applicable to site works such as roads, car parking, street lighting, landscaping, security, etc. are entered. The result, combined with the first phase, is the budget cost for the project as a whole; and

3. Reporting and what-iffing: This phase informs the user of the results and allows for their interrogation. The project can be saved to a project's database at this point and would also be the entry point for projects retrieved from the database.

The cost forecasting is based on standard Building Cost Information Service elements. The information about the house types to be considered and details about it are taken from the user and cost information is produced by integrating costs from the knowledge base for the standard types of housing in the database.

The reporting in done on three main levels.

1. A project wide report for the whole of the scheme. It comprises of a summary report plus supplementary reports designed to convey a picture of the scheme as a whole.
2. A house type report for each house or group of like houses. The elemental breakdown for an individual house as produced by the system is shown in Figure 2.4.
3. An assumptions report which gives the reasoning and factors considered in producing the result.

Total GIFA - 73.3992

Cost/	Element	Cost	Cost/ sqm GIFA	KUQ	KUE
1	Substructure	3563	48.55	44.54	80
4	Fittings&Furnishings	1016	13.84	1.8	1016
2B	UpperFloors	1412	19.25	36.7	30.5
2C	Roof	2895	39.44	44.54	65
2D	Stairs	641	8.73	1.8	641
2E	ExternalWalls	4824	65.72	109.64	44
2F	Windows&ExternalDoors	5991	81.63	25.97	230.74
2H	InternalDoors	196	2.67	1	196
2G1	Partitions	35	0.48	1	35
2G2	PartyWalls	30	0.41	1	30
3A	WallFinishes	25	0.35	1	25
3B	FloorFinishes	32	0.44	1	32
3C	CeilingFinishes	28	0.38	1	28
5A	Sanitary&DisposalInstallations	1516	20.65	1	1516
5D	WaterInstallations	32	0.44	1	32
5H	ElectricalInstallations	1266	17.25	1	1266
5I	GasInstallations	116	1.58	1	116
5K	ProtectiveInstallations	46	0.63	1	46
5L	CommunicationsInstallations	41	0.56	1	41
5M	SpecialInstallations	91	1.24	1	91
5N	BuildersWork	266	3.62	1	266
5G	VentilatingSystems	216	2.94	1	216
5F	SpaceHeating&AirTreatment	2016	27.47	1	2016
6B	Drainage	22	0.36	1	22
6A1	SitePreparation	26	0.35	1	26
6A2	SurfaceTreatment	28	0.38	1	28
6A3	SiteEnclosure&Division	16	0.22	1	16
6A4	Fittings&Furniture	16	0.22	1	16

PRINT REPORT VIEW REPORT EXIT

Figure 2.4 Elemental breakdown of a single house *(Watson et. al. [1992]).*

Other than the above, it also has "What-iffing" facility. This allows the user the opportunity to interrogate the results and see what would happen if system parameters are changed. The paper then talks about the differences from conventional software and benefits of the system, which are similar to the common ones listed in the first chapter.

The system is quite satisfactory from the point of view of a user who wants to know what a proposed project would cost before investing his/her money in it. But from the point of view of a manufacturer who, as a matter of fact, is going to be trying to get the contract from the same project managers who are going to be the users of the system discussed in the paper, it is unsuitable. The system discussed in the paper is not as

accurate as a system used by a manufacturer would need it to be, since unlike the case of the manufacturer for whom the inaccuracy of the system means that much possible loss on the contract, the system required by the project managers need to give them only a overall idea of the cost of the whole project.

Christian [1991] describes developments at the University of New Brunswick in investigating the possibilities of owners and contractors in the construction industry, using artificial intelligence, making financial commitments before design and scheduling occurs. The development of an expert system using a shell program to predict cost-time profiles for certain activities in the construction process is described. First, the data collection and model development for cost-time profiles are described. Then the knowledge-based system for cost-time profiles is described. The shell program for this was written in LISP and the rules in a language called Abbreviated Rule Language (ARL). The development of the pre-engineering estimate is also described. The methodology took two approaches. The first approach was the typical expert system approach of knowledge gathering from experts, and the second was to analyze the historical school construction data. This was followed by the description of the prototype computerized estimating system. For producing the estimate part, a combination of a spreadsheet program and a expert system shell was used. The spreadsheet program was used as a database and for doing cost calculations. The expert system was used to query the user for input to the spreadsheet Further development work, investigating the possibilities of using historical data and expert systems for pre-engineering estimates, before design and scheduling occurs, is also discussed.

A few comments have to be made about the paper at this point. The part of the combination spreadsheet and expert system used in the paper is a bit ungainly. The proper use of the expert system shell has not been made. It was used only as an interface, and was not used to either represent the estimating procedure nor analyze the historical data as it was quite possible to do. This makes the use of the shell itself redundant. Also, this is a

system which prepares the estimate just as a reference to help estimators predict future construction costs based on historical construction cost data.

2.4 Cost Estimation by the Manufacturers

Once the cost analysis of the project is completed and the decision to go ahead with the project has been made, the building contractors are put on the job. These contractors are usually fixed even before the project is fully planned or approved. The site preparations are first started. This includes clearing and leveling of the site, marking out and laying of building foundations, preparations of driveways, parking lots, etc. The next step is to send out to manufacturers for quotes on materials like prefabricated walls, flooring, etc. The foundations are usually laid by the contractors, but it is more time and cost effective to buy prefabricated walls, roofing, etc., rather than build them. The same applies to accessories like doors and windows, which may be bought in standard sizes and in bulk. The same may or may not apply to the roofs, depending on their design and transportability. Unlike prefabricated walls, doors or windows, which are made and assembled completely and then stacked up neatly on trucks and moved to the site, roofs, once made and assembled, can be quite bulky and unwieldy to move on trucks. It is usual to either make or at least assemble the roofs on site. In any case, the lists of materials required are made out and sent out to manufacturing companies to get quotes on them. It is at this point that the system that has been developed comes into play. It is used to estimate the various costs of manufacturing the materials requested, including the markup costs and give the total estimated cost.

Moselhi, et. al. [1991], describes a neural network type artificial intelligence model for solving the markup estimation problem. Neural networks are systems that can "learn". For example, they can generalize a solution from a set of examples representing different problem scenarios and their corresponding solutions or decisions. They simulate the decision makers' process of acquiring experience and ability of producing reliable decisions to new situations. Various existing markup estimation models have been analyzed and the

decision-governing attributes identified. Based on this, a model is proposed and described.

This model, though it sounds quite good as far as neural networks go, has only been theorized, not developed. Other than this, the fact is that neural networks, which are based basically on the idea of copying the human brain show a lot of potential and have seen a lot of development, but and have not yet been perfected as much as expert systems have been. In any case, neural networks too cannot replicate the "intuition" of human experts.

As soon as the quotes are in, the contract is given out based on the bids which are the cheapest but at the same time are also from a reliable company. The latter factor is very important, since, though a company may have given a cheaper bid, if the company deals in low quality material or does not deliver on time, there is no use going in for a saving in costs with the risk of spoiling the name of the building contractor or that of the investors to *their* clients. In cases where the project is huge, involving a very large sum of money, such as in government projects, the project managers usually go over the bids and put the contract out on auction for the lowest bid from a selected group of suppliers. But government projects do not, for the most part, use pre-fabricated housing materials for construction, since their work is mostly concrete based. Pre-fabricated housing comes in mostly only for private residential housing projects.

Once the matter of the bid is settled, the chosen manufacturer and the building contractor draw up an agreement and the deal is finalized. This usually takes place in a space of one to two months, with the first bids given within two weeks and the deals settled within the month. This is also why time is such an important factor in cost estimation work.

Once the deal is finalized and matters such as time frame, payments, etc. have been settled, the manufacturing firm starts work. It is at this stage that the work of the estimator shows its worth. Good estimates fall at their mark, making a profit for the firm, while the not so good ones allow the firm to break even, and the poor ones cause a loss.

There may be too many unforeseen eventualities which may lead to loss too, though good estimates allow for a safety factor which can account for a reasonable amount of loss.

Skitmore, [1988], wrote about the various factors affecting the accuracy of engineers estimates. The first part of the paper reviewed most of the empirical evidence to date and the latter part analyses some actual data. Some of the major factors discussed are the types of projects, the sizes of contracts, geographical locations, number of bidders, state of the market, levels of information, and ability of the estimators.

This paper does give an idea of some of the factors affecting estimates. But more importantly from our point of view, it also shows us why, at least at present, the computer cannot completely *replace* the human estimator. The primary reason, according to the paper, is that the estimate cannot do without the human "intuition." It is for this reason that the developed system has the facility of interacting with the user to let him/her make any changes in the default values that the computer uses to satisfy his/her "intuition" and equalize for any changes required due to factors such as market conditions, etc., which cannot be so easily quantified and dealt with.

2.5 Summary

With this literature review in mind and with the research objectives stated in the previous chapter, a knowledge-based cost estimating system has been developed and will be described in Chapters 3 and 4. It has been incorporated into the form of an expert system the development of which has been described in Appendix A. Real data from a small firm which manufactures and deals in pre-fabricated housing was used both during the development and for the verification and validation of the process using the expert system..

CHAPTER 3: THE APPROACH FOLLOWED

3.1 Methodology

The basic methodology followed to develop the knowledge base was as follows. Data was first obtained from a prefabricated housing manufacturing company. The data consisted basically of:

- Sample blueprints of some typical models of prefabricated houses;
- Detailed cost estimates drawn up for real orders by expert cost estimators; and
- A sample order form.

Based on the cost estimate sheets and knowledge of standard residential construction work, the data was considered in the following sections:

- *Roof*: This section included all information related to the roof;
- *Walls*: This section contained all the information related to both interior and exterior walls;
- *Floors*: This section contained all the information connected with the floors; and
- *Others*: This section contained all the information and data which could not be put together with the roof, walls, or floors, including miscellaneous components such as interior and exterior trim, insulation, doors, windows, etc.

Once the data was divided into sections, the cost estimate data available was analyzed thoroughly to develop formulae to produce a cost estimate as accurate as possible. The input required was obtained from the data sheets available, and the results were cross-checked with the cost values from the same data sheets for accuracy. Some of the assumptions made while developing the formulae and structure, such as the roof type, were made based on the blueprints. For example, it was assumed that the roof most commonly used is a gable roof, and all roof related calculations were done with this in mind.

The method by which the knowledge structure was developed and organized has been described in the next chapter. The formulae which were developed to produce a cost

estimate used some standard relationships to obtain necessary values from available data. One simple example would be the relationship used to calculate the number of floor joists in a rectangular area.

Length	=	a ft
Breadth	=	b ft
Joist Spacing	=	x inches

If length of joist > b

Number of joists	=	(a times (12/x)) + 1

In the case of the developed formulae, cost per feet of joist material is used. The number of joists calculated from the above formula is multiplied by the breadth of the building to get the total lineal feet of joist material required. This value is multiplied by the cost per feet of joist to get the total cost of joist material. The above is a simple example where a standard relationship is used to help develop a formula which has been used for this process. Other formulae were developed based directly on the cost data available. All of the formulae have been listed in the next chapter.

The representation of this knowledge and formulae in the expert system is what formed the knowledge base of the system.

3.2 The Domain

As already mentioned, the system which has been developed is meant for the use of a manufacturing firm which supplies prefabricated material such as walls, roofs, etc. and other miscellaneous material such as insulation or underlayment material for residential housing construction. It produces a total cost estimate for purposes such as preparing a bid to take up a supply contract or giving a quick initial estimate to a customer. It has been delivered on an expert system platform, created using an expert system shell by the name of Kappa-PC™ produced by Intellicorp™.

3.3 Why an Expert System

The main reason for using an expert system as a delivery system for the estimation process is that it is the best method to represent both the data structure knowledge and the formulae which are based on that structure. A conventional program would only be able to use formulae converted into computer code such as a "C" program or a spreadsheet program. It would not be possible to represent the knowledge of the developer in a conventional program. Other reasons are the advantages that an expert system has over conventional programs, which have been listed out in the first chapter.

3.4 About the shell

The main reasons for having used an expert system *shell* to create the delivery system instead of using languages such as LISP or PROLOG may be summed up as shown below:

1. Modifications and updates to the rule-base can be made by a person with minimal amount of training;
2. Far better graphical user interface;
3. Easier to trouble-shoot for a person who is not an expert programmer or computer expert;
4. Easier transportability and setting up for use;
5. Makes for easier deployment; and
6. Cheaper to create;

The Kappa shell is a windows based software, as a result of which, the graphical interface is very good. The user interface which can be created using it can be made to be quite self-explanatory. The user need not understand how to operate the system, nor need to be a person well versed in the use of computers. The system can be set up such that the user simply needs to enter information in the appropriate spots using the mouse and typing it in and then once done, clicking on the appropriate buttons using the mouse. The results

are produced in seconds. The system which has been developed makes complete use of this facility of Kappa.

The above mentioned feature is quite important since, as stated before, the system may have to *substitute* for the human expert at times to give an estimate in situations where the expert is out of town looking at another site and will not be available for some time and the customer wants a quick initial estimate. In such a scenario, the only person available to operate the system may be a layperson who can just answer the relevant questions (asked by the system) over the phone and give the results produced over the phone itself. This would, other than being helpful for the customer would also increase his/her opinion of the helpfulness and efficiency of the firm, make for better public relations and advertisement for the firm.

CHAPTER 4: THE KNOWLEDGE BASE

The methodology followed to develop the knowledge structure has been explained in detail here. As previously stated, the data was considered in sections as floors, walls, etc. and analysed. The methodology used for one area, floors, shall be described in detail as an example.

4.1 Floor

The primary raw materials required for the manufacture of prefabricated floors were first listed out based on knowledge acquired from texts on construction. It was assumed that since it was prefabricated, the components such as foundation posts and concrete pier blocks may be ignored. Even if ordered, these costs may be easily calculated manually from the cost sheets since the estimation for these is per piece per piece of material. The primary components of raw material to be considered would thus be joists, girder, floor sheathing, adhesive (if required), joist hangers and sill plates.

The cost estimate sheets were then analyzed to find the cost percentages contributed by these components to the total cost. All the joists were considered as one whole part regardless of varying size since the primary target was to get a general idea of the cost percentages and not detailed estimation using traditional methods. The average cost percentages thus found in the case of a joist and girder design are given in Table 4.1.

The major cost drivers among these components are seen to be the joists and the sheathing, the girder and sill plates being minor percentages and also isolated to the first floor only. Based on this information, cost estimation realtionships were derived for the above materials.

Table 4.1 Major Cost Percentages

	First floor	Second floor
Joists	58 %	67 %
Girder	11 %	
Sheathing	26 %	31.65 %
Adhesive	0.7 %	0.65 %
Joist hangers	0.3 %	0.7 %
Sill plates	4 %	
TOTAL	100 %	100 %

1. (a) Floor Joist: (In a joist and girder design)

Cost = [{(TL x O.C.F.) + 1} x (TW+2)] x 1.50 x 1.10

Where,

TL = Total Length of the building

O.C.F. = On Center Factor = 12/O.C. in inches

TW = Total Width of the building

1.50 = Average cost per LF of floor joist acquired from real data

1.10 = 10 % of material to account for header joists, overlapping of joists over the girder and variations in costs for the different joist thicknesses

LF = Lineal Feet

The above realationship was arrived at by the following steps. The expression {(TL x O.C.F.) + 1} gives the number rows of joists required to span the length of the building. This formula is the standard formula used in the case of a regular rectangular area to find the number of joists. In this case, even if the area considered is not a perfect rectangle, but is made up of irregular areas, the expression gives a close approximation of the number of joists required. The expression (TW+2) gives the length of the joists required. The "2" is to account for the resting of the joists on the sill plates. Though the

final joists may not be a single piece of length equal to the width of the building, the expression used will do since the cost has been considered per lineal feet of joist material. These two expressions together thus give the total lineal feet of joist material required.

In a similar way, the

1. (b) Floor Joist: (In a truss design)

Cost = FA x O.C.F. x 0.83 x 1.3

Where,

FA = Floor area

0.83 = Average cost per LF

1.3 = 30 % of the cost to account for header joists, overlapping of trusses over the girder, joist material required for the outer widths and blocking between trusses

The floor area is used since the original relationship may be reduced to the one shown above in the following manner.

(TL x O.C.F.) x TW = (TL x TW) x O.C.F. = FA x O.C.F.

There is no "1" or "2" added to the truss number or width respectively as in the case of the joist construction since they are better accounted for in the 30 % in the relationship in this case.

2. Floor T & G (tongue and groove sheathing)

Cost = (FA/32) x 16.219 x 1.16

Where,

32 = Area occupied by one 4x8' sheet which is the standard form in which sheathing material is manufactured

16.219 = Cost per sheet of sheathing material

1.16 = 16% to account for material wastage due to the irregularities such as the building not being a perfect rectangle

This is a simple and clear formula since there is a direct relationship between the area of the building and the floor sheathing area. The traditional method is to add 4 % for wastage in buildings with irregular shape, but from the data it was found that 16 % is more realistic.

3. Girder (for first floor only)

Cost = X Y

Where,

X = TL + 2 if TW < 27 feet

TL x 1.5 if TW > 27 feet

Y = Average cost

= 4 if girder width is 10"

5 if girder width is 12"

The expression {TL + 2} is used to find the length of the girder to be used. The girder spans the length of the building and acts as a support to the joists. The "2" is to account for the material which rests on sill plates at the ends of the building and for any overlapping due to use of two pieces of girder material being used instead of a single one. The expression {TL x 1.5} is used for very long lengths due to the following reason. From the analysis of the data, it was found that for lengths exceeding 27 feet, the widths were proportionately higher which meant that one girder would not be enough to let the joists span the whole area. On further analysis of the data, using more than 1.5 times the length was found to be a waste of material. The above expression was found to give good results for most of the sets of data used. The cases where it fell short of the costs in the data did not make a large difference since the girders were not a major cost driver.

4. (a) Sill plates (In a joist and girder design)

Cost = (Perimeter + 15) x 0.56 x 1.18

Where,

15 = Wastage and overlapping of material at joints

0.56 = Average cost per LF

1.18 = 18 % to account for material used for porches or garages or for interior walls which need extra support

(b) Sill plates (In a truss design)

Cost = (Perimeter + 15) x 0.56 x 1.08

Where

1.08 = 8 % to account for material used for porches or garages or for interior walls which need extra support

These were again straightforward expressions since the sill plates are normally used only for the perimater area of the building.

5. Adhesive

Cost = (FA/107) x 1.61

Where,

107 = Approximate area covered by each can of adhesive, derived from data

1.61 = Average cost

This is again a clearly straightforward expression since there is a direct relationship between floor area and adhesive amount required to cover it.

6. Joist hanger

If area < 1000 Square feet, add $5

If area > 1000 Square feet, add $10

This was an approximate value derived from the data. Since this was not a detailed estimation and the cost under consideration also not being large enough to justify it, a more detailed formula would not be required.

In a similar method, expressions were derived for the other sections of the building and have been given below.

4.2 Wall

The methodology used here is similar to that used for the floor estimation. The cost components considered have been listed out below. The height considered is the standard 8 feet. The thickness of the walls is not taken directly into account. Though it is assumed that the outer wall panels are thicker, which would also explain the higher cost per lineal feet.

1. Outer wall panels

   Cost = Exterior perimeter x 7.3 x 1.06

   Where

   7.3 = Average cost per LF

   1.06 = 6 % to account for miscellaneous material and wastage

2. Inner wall panels

   Cost = Total interior wall length x 5.4 x 1.06

   Where

   5.4 = Average cost per LF

   1.06 = 6 % to account for miscellaneous material and wastage

3. Pressure treated bottom plate (for concrete slab floor)

   Cost = Total length of all walls x 0.1 x 1.06

   Where

   0.1 = Average cost per LF

   1.06 = 6 % to account for miscellaneous material and wastage

4. Garage wall panels (10 feet high)

   Cost = (Perimeter - X - Y) x 10.26 x 1.06

   Where

   10.26 = Average cost per LF

   X = Length of garage, to account for the garage door

   Y = Dimension of side (if) adjoining the building

   1.06 = 6 % to account for miscellaneous material and wastage

5. Bracing

Cost = Exterior perimeter x 0.16 x 1.06

Where

0.16 = Average cost

1.06 = 6 % to account for miscellaneous material and wastage

4.3 Garage

The approximate costs of garage components other than wall panels, which are calculated in the wall section, have been calculated here.

1. Garage door trim

Cost = (Perimeter - breadth) x 0.46

Where

0.46 = Average cost per LF

2. Garage float header

Cost = Length x 0.4

Where

0.4 = Average cost per LF

3. Garage truss or garage beam depending on the construction

Cost = Length x 5.75

Where

5.75 = Average cost per LF

4.4 Roof

The costs of the roof system have been calculated in a similar manner to that of floors. But the safety factor needs to be higher here since the possibility of error due to the odd shapes is more. It must be remembered that the type of roof is assumed to be a gable roof.

1. Roofing materials

$$Cost = RA \times 0.33$$

Where,

RA = Roof area

= Floor area + (Perimeter x Eave Overhang)

0.33 = Average cost per square feet of roof area

2. Roof trusses

$$Cost = Total\ area \times X$$

Where,

X = 0.76 if gable is flush

= 0.92 for 1 foot overhang

= 1.00 for 1.5 feet overhang

= 1.08 for 2 feet overhang

3. Floor and sheathing OSB

$$Cost = Total\ area \times 0.5$$

Where,

0.5 = Cost per square feet of sheathing

4. Gable projection

$$Cost = Perimeter \times 2.1$$

Where,

2.1 = Cost average

5. Other gable material

$$Cost = Total\ Area \times 0.25$$

Where,

0.25 = Cost value developed based on data

The roof estimates have more error than other areas due to the irregular shapes possible for roofs.

4.5 Exterior Trim

The cost components for these are given here.

1. Plywood soffitt

$$\text{Cost} = (\text{Perimeter} + 12) \times 0.55$$

Where,

12 = Overhang

0.55 = Average cost per LF

2. Plywood boxing

$$\text{Cost} = (\text{Breadth} + 4) \times 1.5 \times 0.55$$

Where,

4 = Overhang

1.5 = Constant to account for other miscellaneous material and wastage

0.55 = Average cost per LF

3. Spruce primed

$$\text{Cost} = \text{Perimeter} \times 0.5$$

Where,

0.5 = Average cost per LF

4. Gable return box

$$\text{Cost} = (\text{Length} \times 0.14) \times 7.745$$

Where,

0.14 = Constant developed based on data

7.745 = Average cost per LF

5. Gable return board

$$\text{Cost} = \text{Length} \times 0.25$$

Where,

0.25 = Average cost per LF

6. Porch rail

Cost = Perimeter x 1.1

Where,

1.1 = Average cost per LF

7. Porch balusters

Cost = (Perimeter - length) x 1.5 x 2.75

Where,

1.5 = To account for number of balusters per LF

2.75 = Average cost

4.6 Exterior Decorative Trim

These costs are just a approximate figure and a more accurate value might be obtained using a list of prices since the costs for these components are fixed amounts. All the numbers here are the average costs.

1. Window head panel

Cost = Number of windows x 14

2. Window shutters

Cost = Number of shutters x 11

3. Porch columns

Cost = Number of columns x 34

4. Cupola with weather vane

Cost = Number required x 220

4.7 Exterior Doors and Windows

The costs for these too are approximate and better values may be obtained using a pricing sheet. The numbers here too are average costs

1. Exterior doors

Cost = (Number of single doors x 140) + (Number of double doors x 280)

Where number of double doors includes single doors of top length greater than 5 feet.

2. Garage door

Cost = Length of door x 30

3. Windows

Cost = Total window area x 8.66

4.8 Underlayment Materials

Cost = Area of Underlayment x 0.34

Where the area might be the whole area of the building or just the kitchen and baths

4.9 Insulation

Insulation costs are also a straightforward calculation. All numbers used are the average cost per square feet of area calculated from the data available.

1. Floor

Cost = Total floor area x 0.18

2. Ceiling

Cost = Total ceiling area x 0.29

3. Wall

Cost = Total wall lengths x 0.115 x 6.8

Where,

0.115 = Average cost per square feet

6.8 = Constant to account for the width of the insulation material

4. Insulation supports

Cost = 6 dollars per floor

4.10 Decking Materials

Cost = Area of decking required x 2.4

Where

2.4 = Average cost per square feet

4.11 Interior Doors and Trim

The costs for the interior doors also is an approximate one and a better value may be obtained using a pricing sheet.

1. Interior doors

   Cost = Total top lengths of all interior doors x 25

2. Window trim

   Cost = Total length occupied by all windows x 3.7

3. Door trim

   Cost = Total door lengths x 4

4. Baseboard

   Cost = Total wall length x 4

4.12 Shelving

This is the costs of all the material such as closet shelving material.

Cost = (X x 0.8) + (X x 0.85)

Where,

X = Total lineal feet of closet walls

0.8 = Cost per LF of shelving board

0.85 = Cost factor for miscellaneous shelving material

4.13 Stairways, Cabinets and Vanities

1. Stairways

   Cost = Lineal feet of handrail required x 30

2. Wall cabinet and base

   Cost = Total feet of cabinet length x X

   Where,

   X = 65 for standard set

   = 86 for upgrade set

3. Vanity cabinet and base

   Cost = Total height of cabinet in inches x X

   Where,

   X = 2.92 for standard set

   = 3.9 for upgrade set

4. Counter top

   Cost = Total length of counter top in feet x 15.25

5. Cabinet finish

   Cost = Total feet of cabinet length x 6.75

6. Sink

   Cost = 31 per piece

7. Sink faucet set

   Cost = 47 per set

The sink costs are approximate and better values may be obtained from a pricing sheet. The same is applicable for plumbing and wiring supplies.

4.14 Summary

This chapter has given the knowledge base which has been developed. The form of this knowledge base as incorporated into the system has been given in Appendix B.[1]

[1] *Note: Data from the company is confidential.*

CHAPTER 5: EXPERT SYSTEM PERFORMANCE

5.1 Performance

The Expert System model which was developed based on the knowledge base formed was used to run various sets of data to test the accuracy and performance of the formulae and methodology used. The results of three of the sets of data used has been given and discussed in this chapter.

5.2 Results obtained

The results obtained using the Expert System and the costs for the same data acquired by the human expert have been given in the Table 5.1.

Table 5.1 Comparison of Results obtained

Data	Floors	Roof	Walls	Others	Total	% Error (based on being correct)
ES 1	4362.11	2936.96	5029.75	2788.82	15117.64	1.75
Expert1	4418.34	2367.68	5304.70	3296.58	15387.30	
ES2	6981.85	4596.84	7252.10	4711.62	23542.41	4.14
Expert2	7529.83	5369.19	7389.98	4270.12	24559.12	
ES3	4574.25	3659.71	4080.65	3450.26	15764.87	7.99
Expert3	5397.02	4019.39	4485.66	3231.25	17133.32	

The basic procedure followed by the system is that the input is first entered into the system. This input, which is stored in the system in the structure that has been developed, is then processed through the formulae which were developed. The formulae

are in the form of rules in the expert system, and are run based on the input. Each rule, if operated, calculates a part of the cost and adds that cost to the total cost. Finally, after all the rules (formulae) have been executed, a markup percentage to account for the operating costs such as transportation costs, etc., is added to the total cost which has been calculated. The final result displayed includes this markup cost. The default markup percentage used in this case is 47 %. This may be changed inside the system if necessary to recalculate the total cost with the new markup value.

5.3 In Summary

Using the formula shown above in the form of rules in the system, the results obtained had an average error of 5 %. The procedure and formulae developed thus seem to meet the goals set quite well. The primary goals of developing a method to produce a rough cost estimate in a short time and to show the advantage of using an expert system for the purpose have both been met.

The error is primarily higher when the input data used does not have the all the corresponding plans with them. This obviously meant that a few assumptions were made both in developing the system and in the verification and validation. The error may be reduced further if complete sets of data are used to make slight changes as appropriate or necessary in the formulae used. The three sets of data used to get the results shown in the table had the appropriate plans. The error due to the roof costs were also seen to be, on the average, higher than the error for other parts of the building. One way of reducing this might be to use a bit more detailed estimating methods, as is done traditionally. One more factor to be considered is that this system will give quite a good initial estimate in a scenario where a rough estimate is inquired over the phone, in the absence of a plan, even by a layperson.

CHAPTER 6: A CASE STUDY

6.1: Input Data

A sample user session is given as a case study in this chapter. The input data which are used, along with the plans has been given in this section. The session takes the complete path that a real user would take through the expert system.

First floor:

Length	=	48 ft
Breadth	=	30 ft
Total internal walls	=	106 ft

Joist construction

Adhesive required

Joist hangers required

Joists spacing 16" O.C.

Pressure treated sill plates

Second floor:

Length	=	48 ft
Breadth	=	27 ft (average of left and right widths)
Total internal walls	=	170 ft

Truss construction

Adhesive required

Truss spacing 24" O.C.

Truss hangers required

Garage:

Length	=	22.33 ft
Breadth	=	20 ft

The height is assumed to be 8 ft for all floors and the garage.

Roof:

Stick construction (beam and rafter construction)

Rafter spacing is 16" O.C.

Eave overhang is 1 ft

Gable type is flush

Gable projection is 1 ft

Sheathing is 15 x 32 plywood

Walls:

Pressure treated bottom plates for both floors

No tenant separation walls

Others:

Required:

Plywood Soffitt

Plywood Boxing

Gable Return Box

Gable Return Board

Spruce Primed

Decorative Window Head Panels	=	5 pcs
Decorative Window Shutters	=	6 pcs
Decorative Porch Columns	=	2 pcs

Insulation required for all walls second floor flooring and first floor ceiling

Insulation supports for first floor

Baseboard for first floor

Underlayment area	=	248.04 sq. ft. (kitchen and bathrooms)
Shelving length	=	25.83 ft
Stairway length	=	46 ft

6.2: Plans used for the Input Data

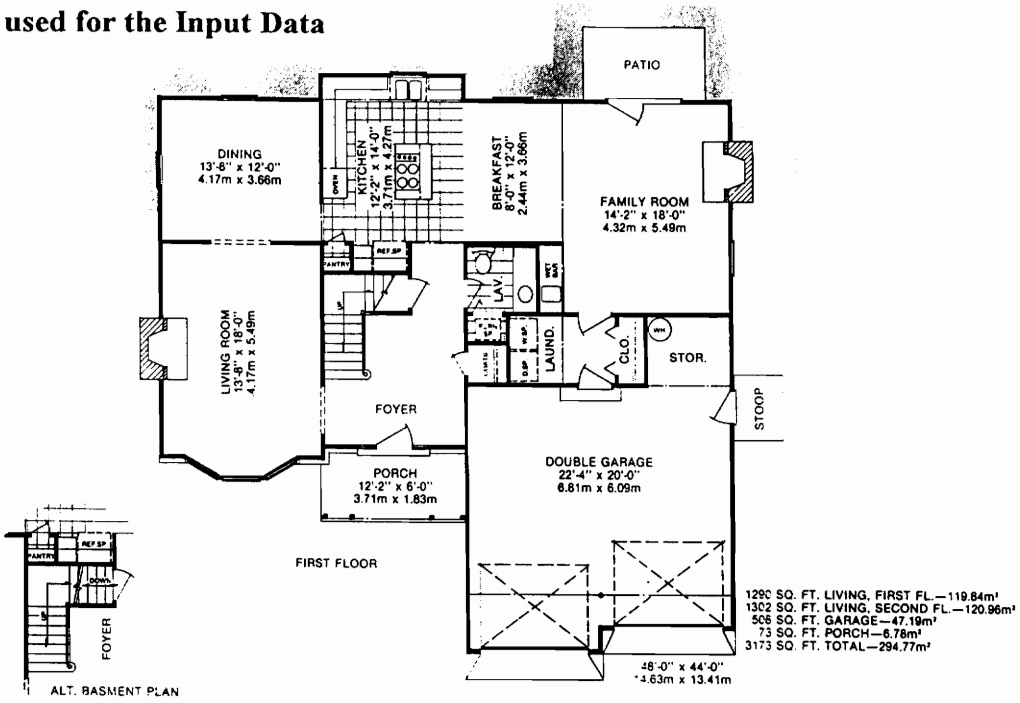


Figure 6.1 First floor plan under consideration

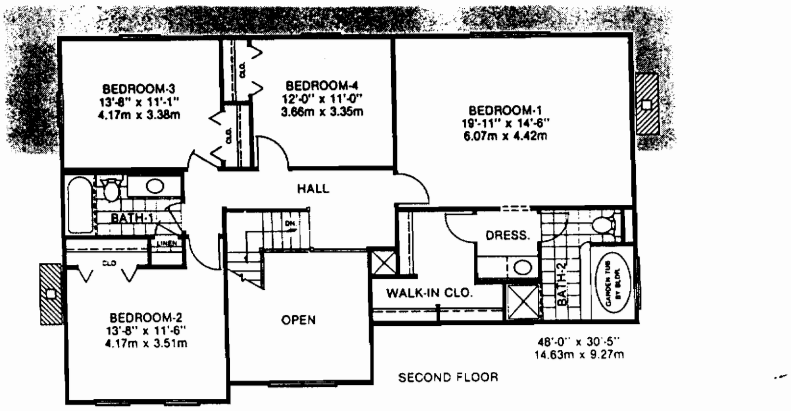


Figure 6.2 Second floor plan under consideration

6.3: The Session

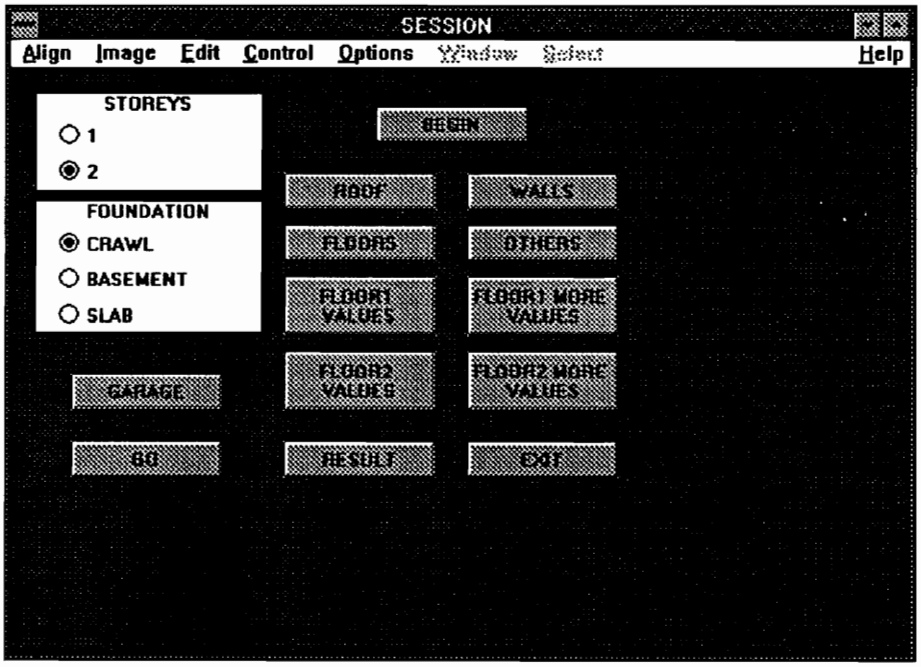


Figure 6.3 Main user interface window

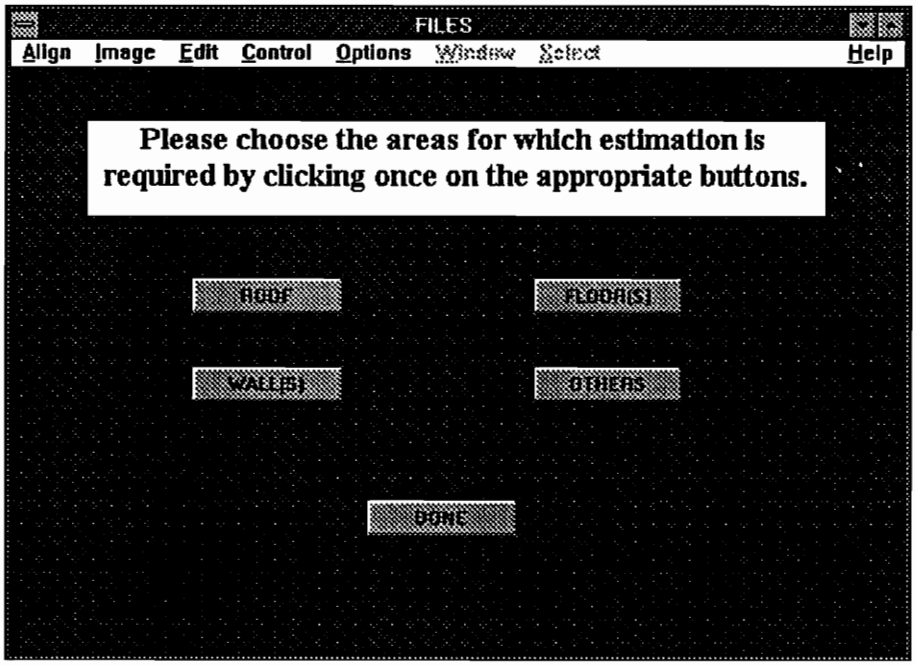


Figure 6.4 Window after clicking the "BEGIN" button

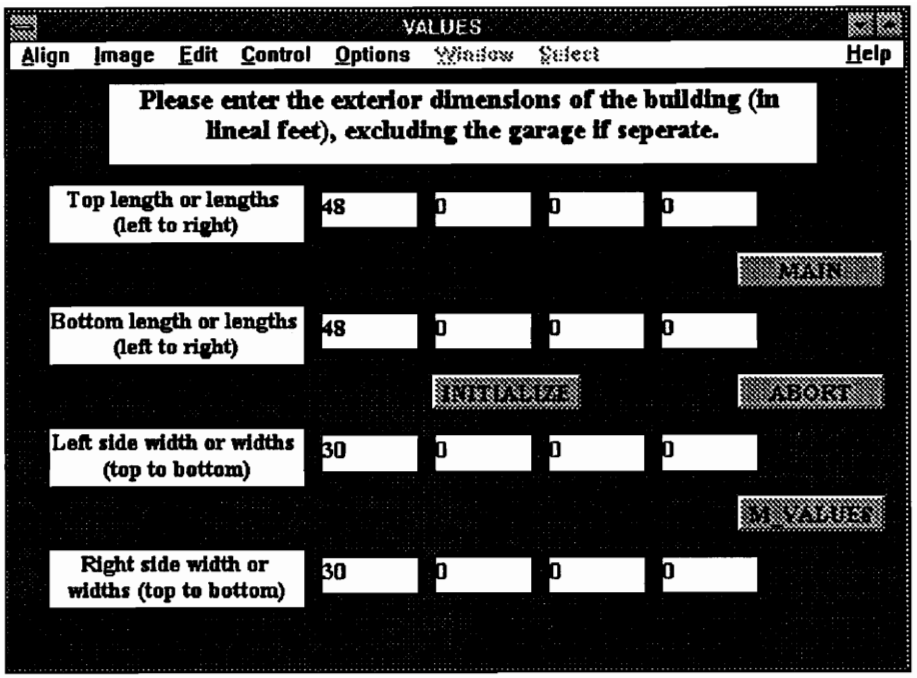


Figure 6.5 Interface window to enter first floor dimensions

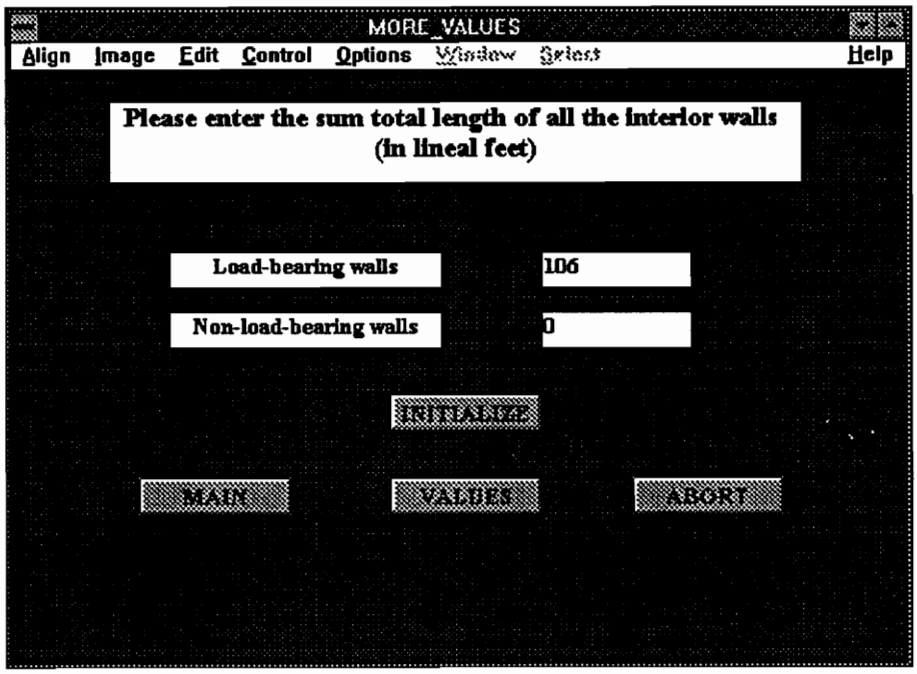


Figure 6.6 Interface window to enter other first floor dimensions

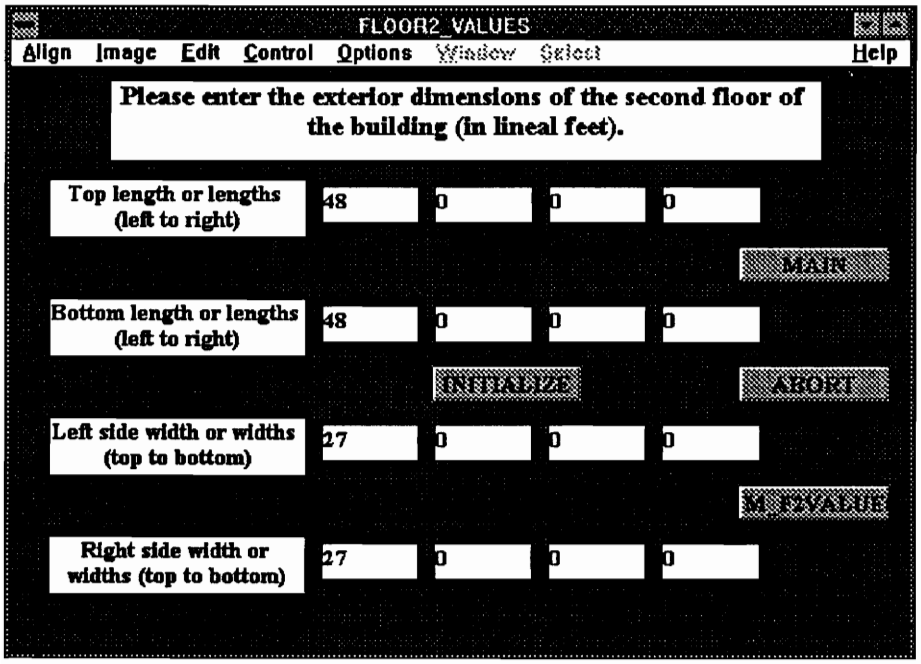


Figure 6.7 Interface window to enter second floor dimensions

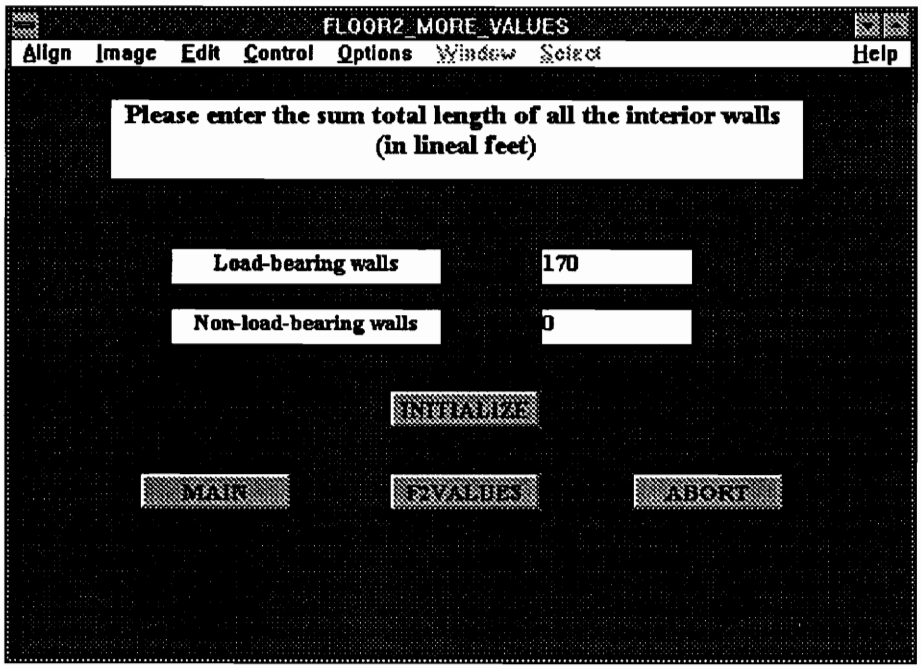


Figure 6.8 Interface window to enter other second floor dimensions

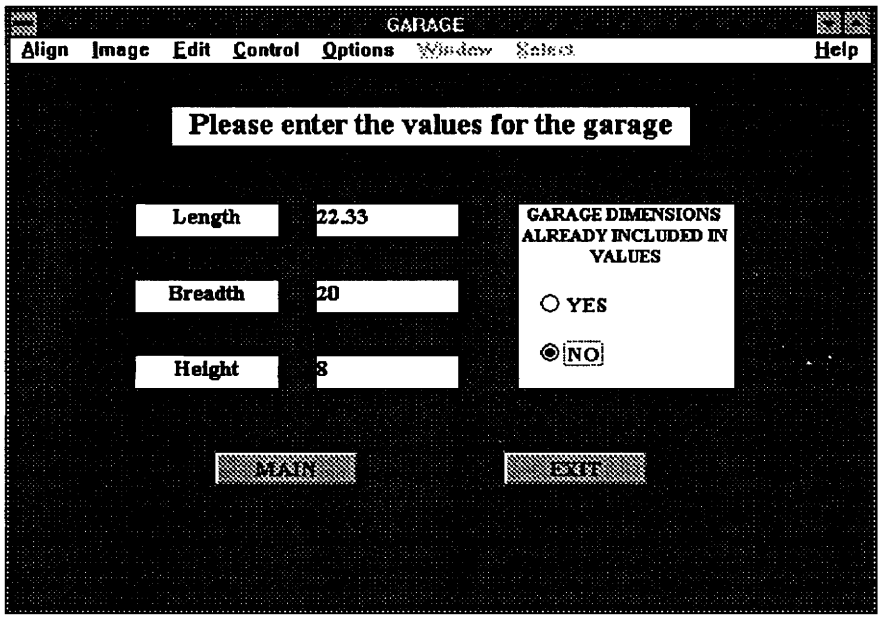


Figure 6.9 Interface window to enter garage dimensions

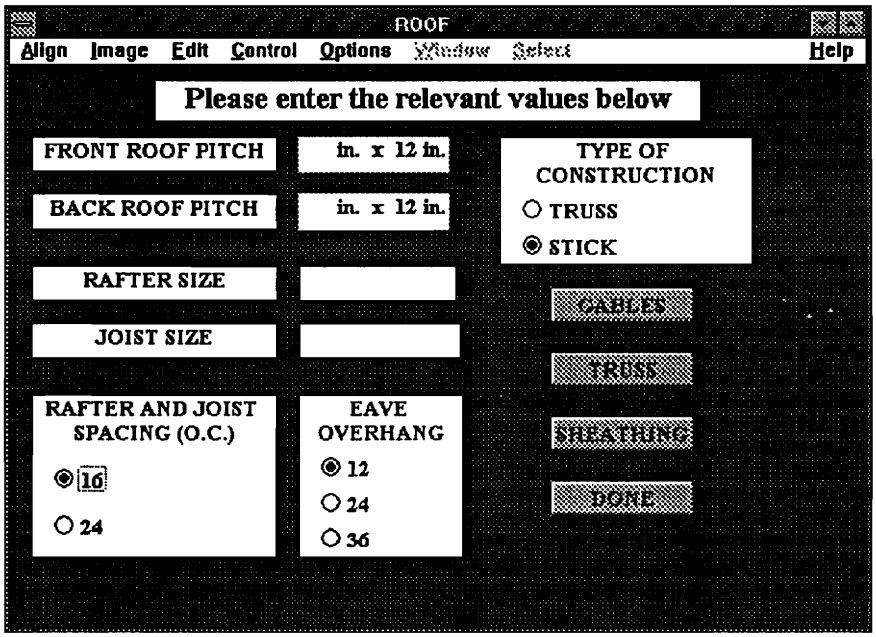


Figure 6.10 Main interface window to enter roof input

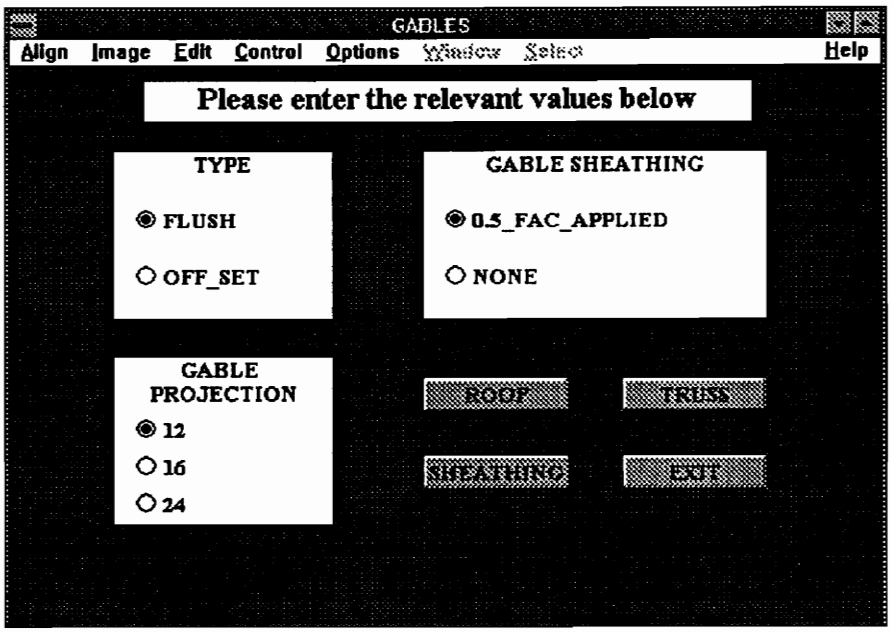


Figure 6.11 Interface window to enter gable input

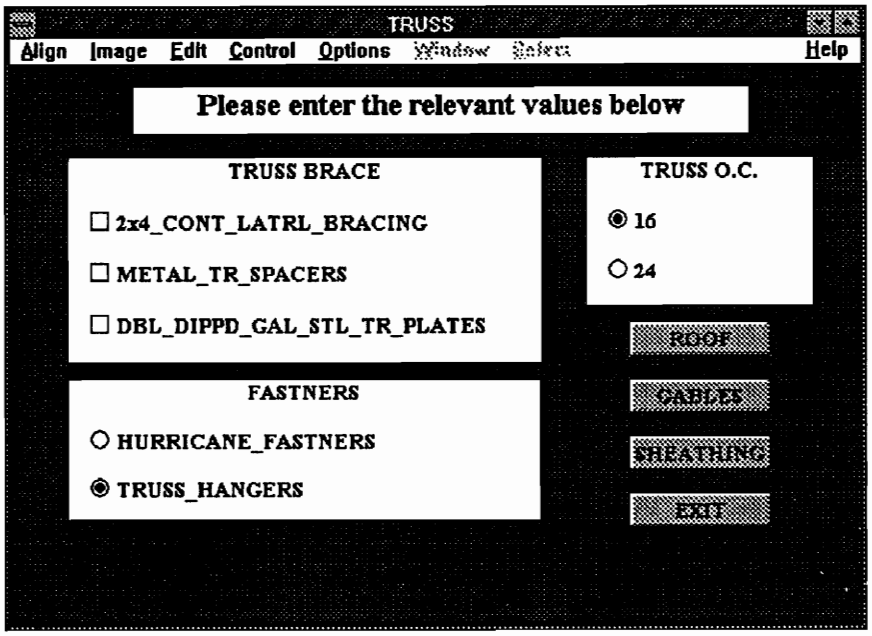


Figure 6.12 Interface window to enter truss input

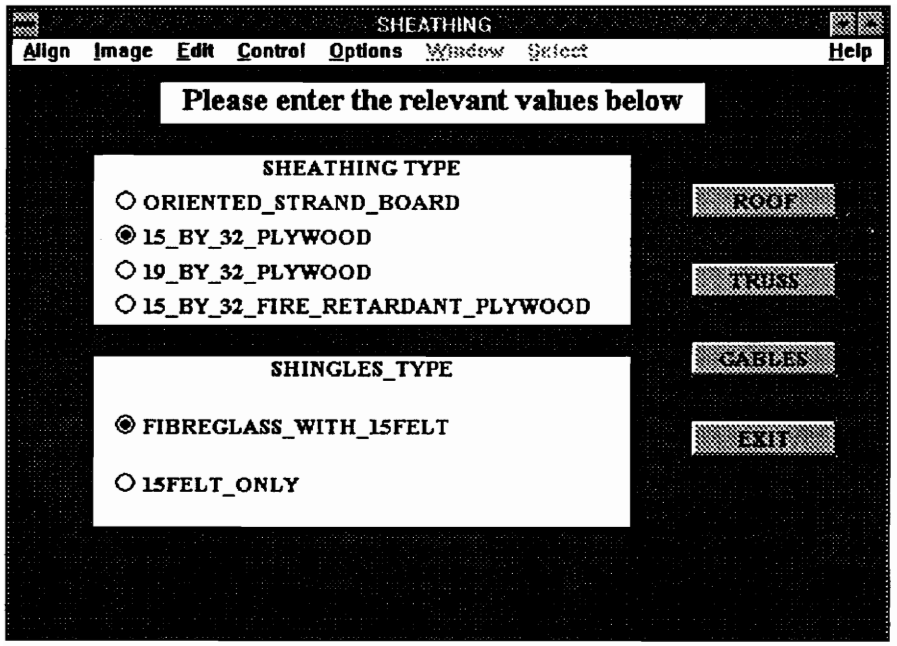


Figure 6.13 Interface window to enter sheathing input

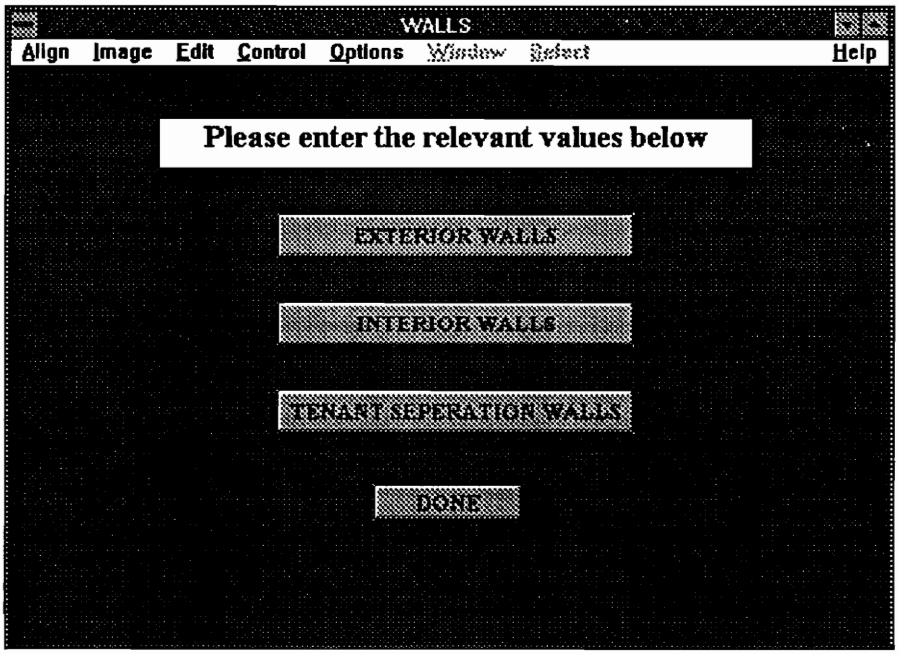


Figure 6.14 Main interface window to enter wall input

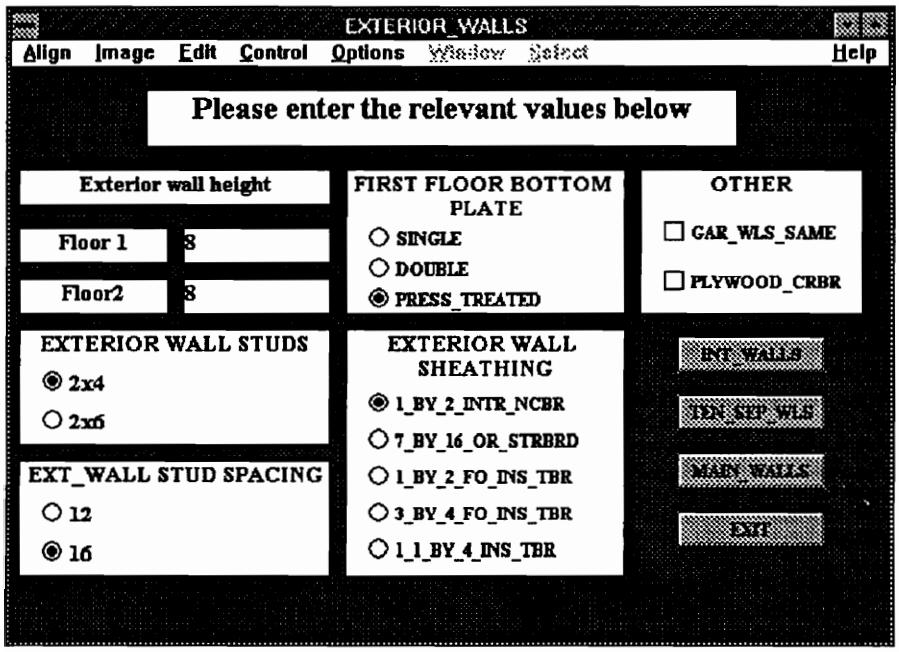


Figure 6.15 Interface window to enter exterior wall input

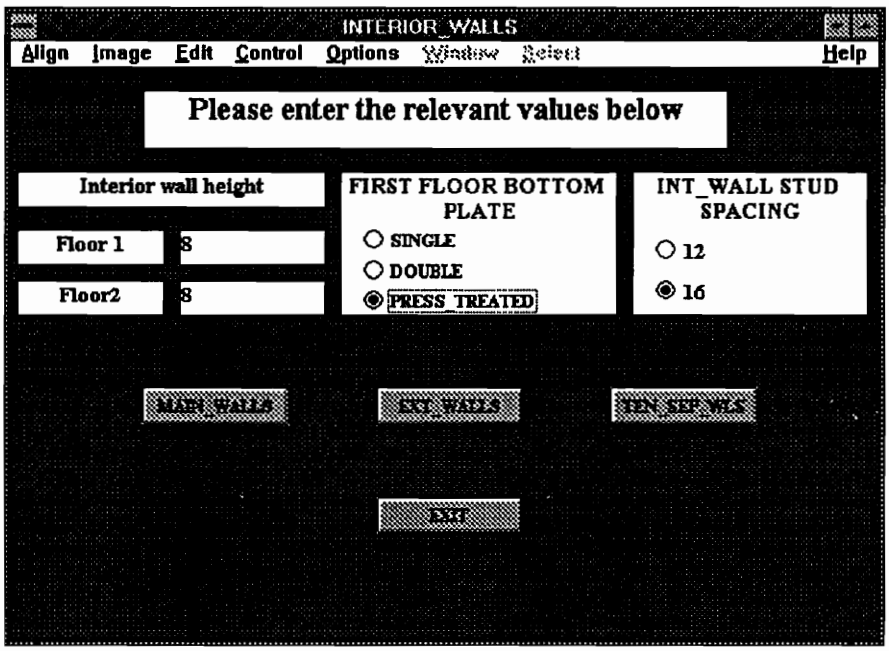


Figure 6.16 Interface window to enter interior wall input

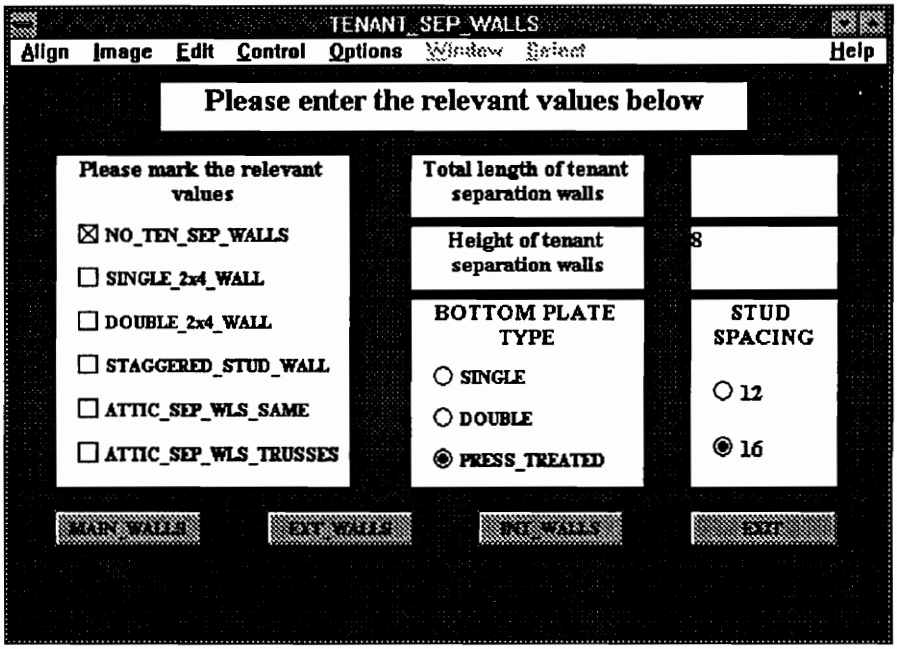


Figure 6.17 Interface window to enter tenant separation wall input

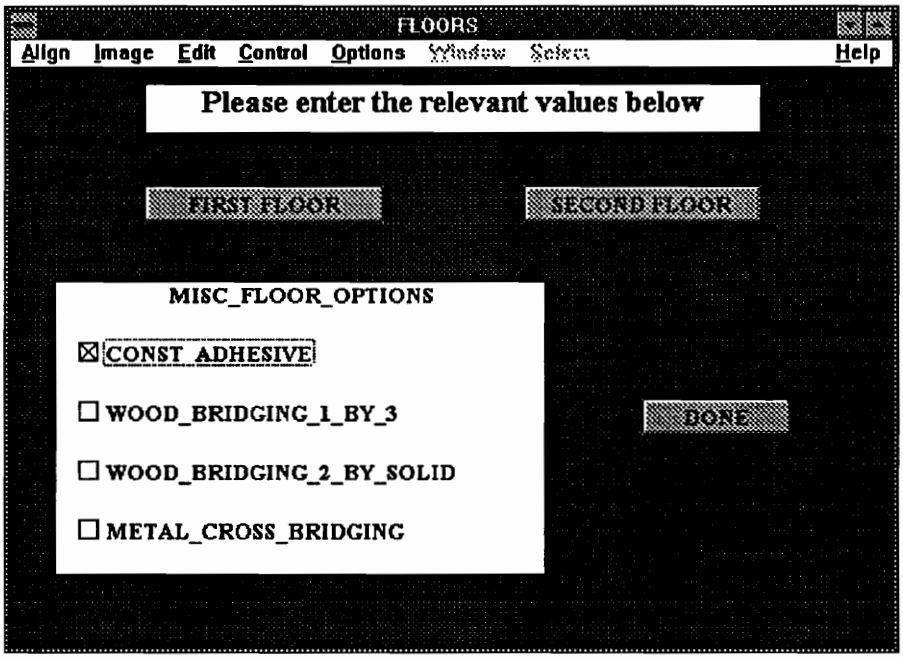


Figure 6.18 Main interface window to enter floor input

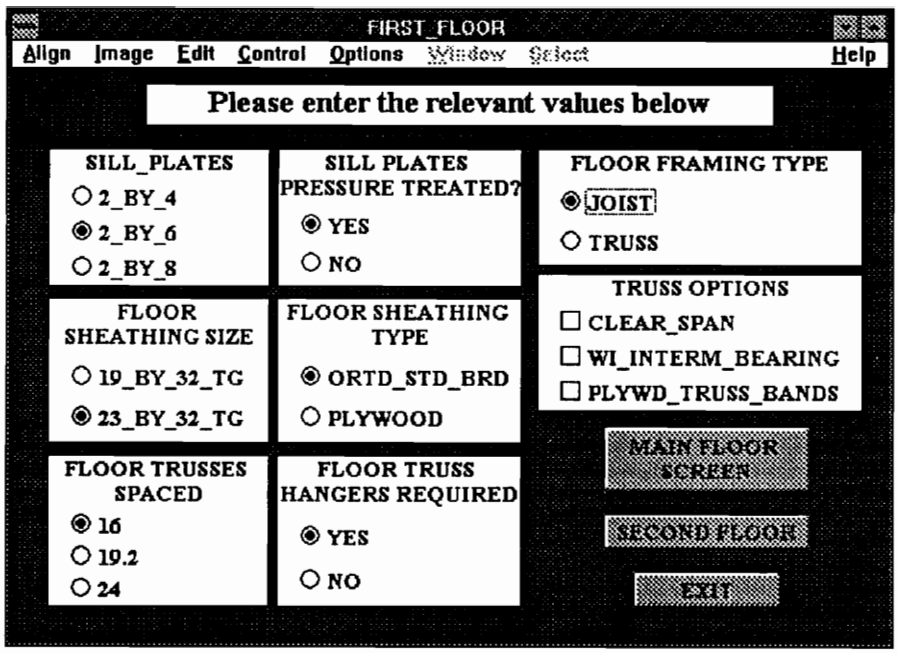


Figure 6.19 Interface window to enter first floor flooring input

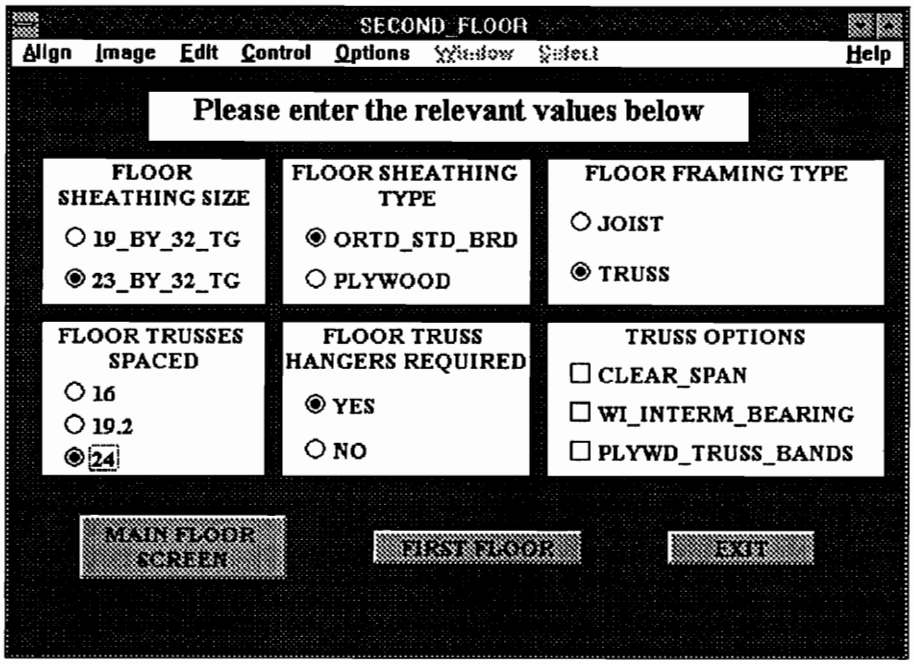


Figure 6.20 Interface window to enter second floor flooring input

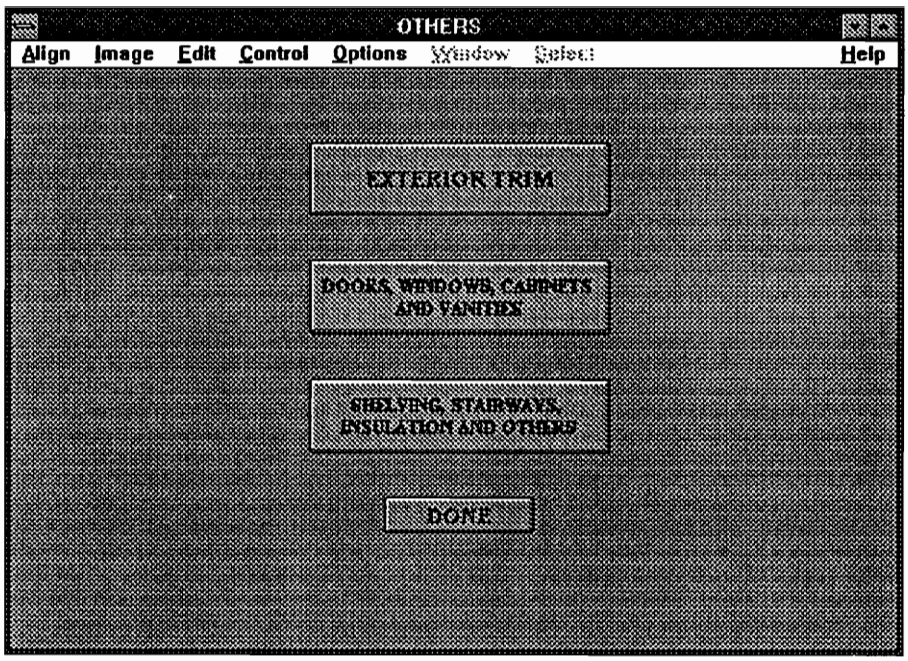


Figure 6.21 Main interface window to enter "other" input

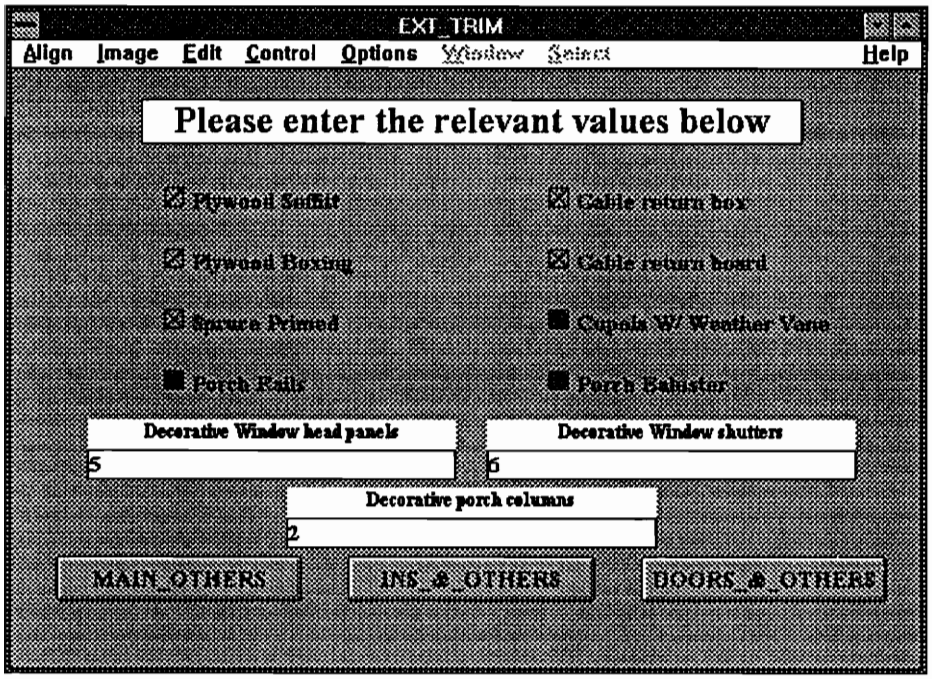


Figure 6.22 Interface window to enter exterior trim input

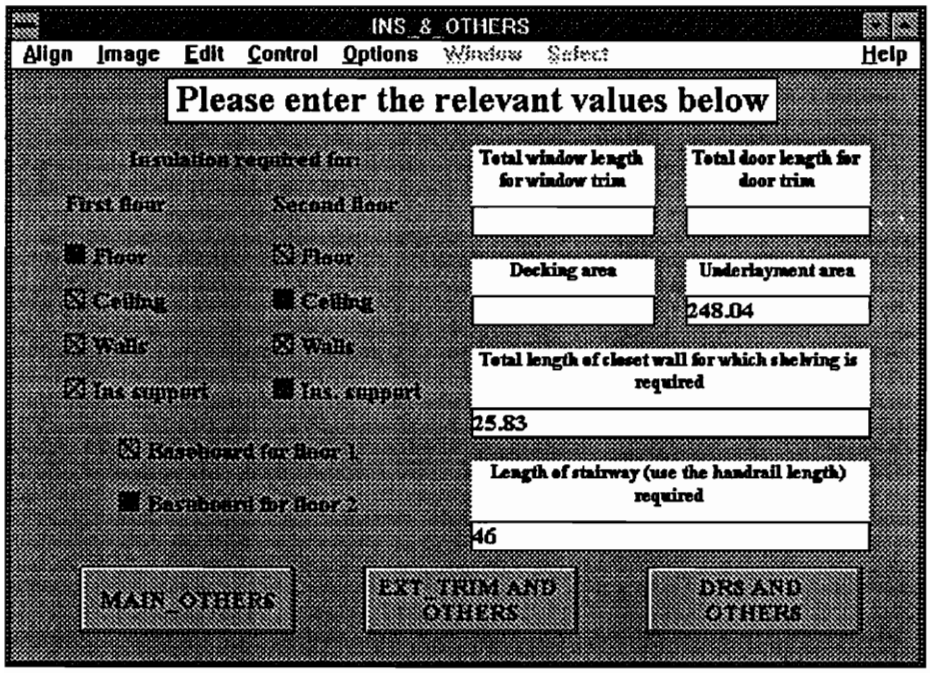


Figure 6.23 Interface window to enter insulation and other input

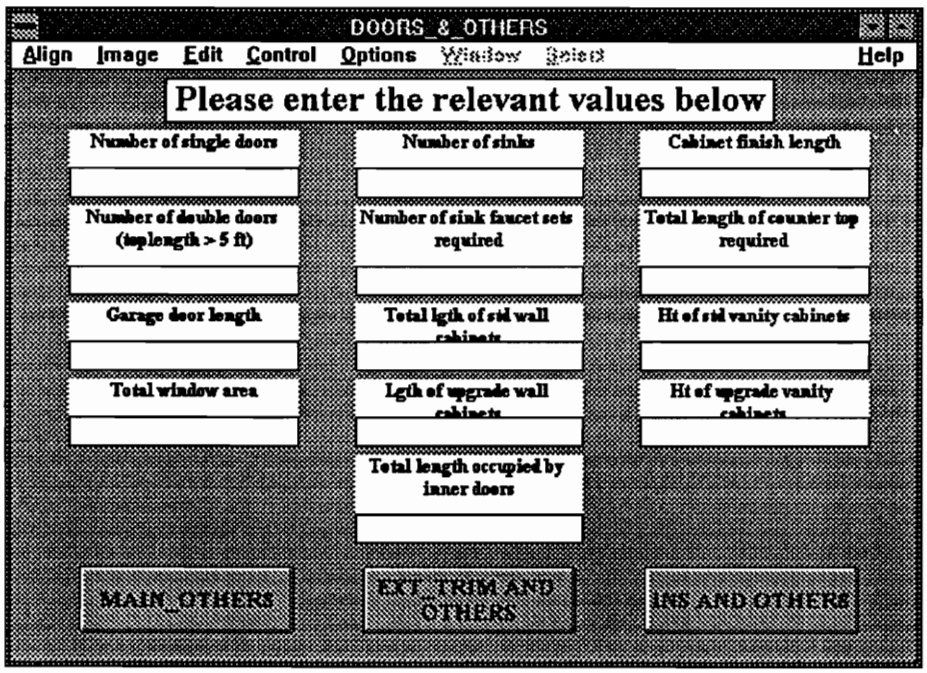


Figure 6.24 Interface window to enter doors and other input

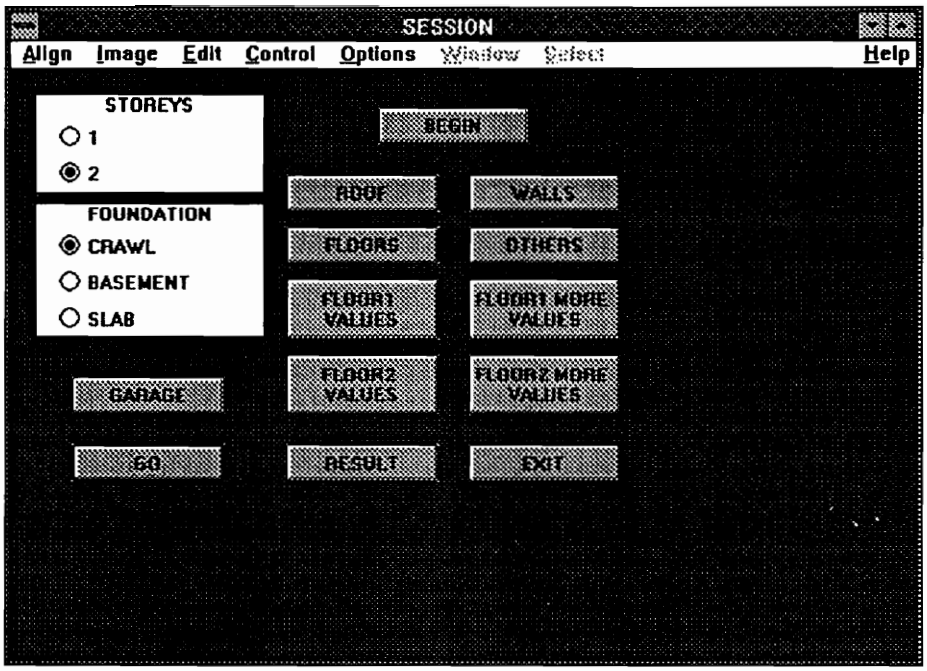


Figure 6.25 Back to the main interface window after entering all inputs

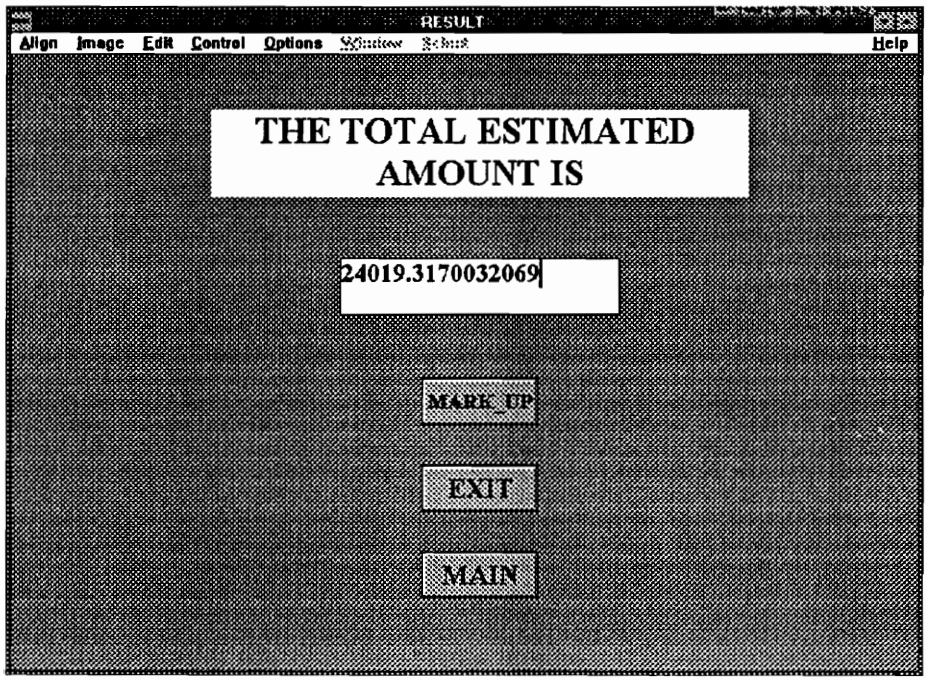


Figure 6.26 The result window

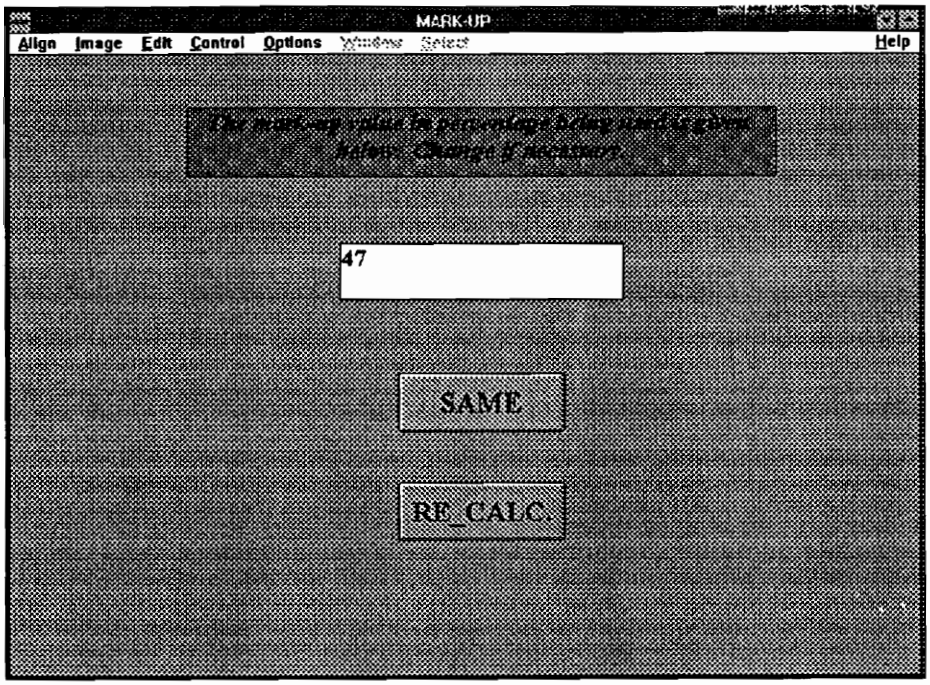


Figure 6.27 The markup value window

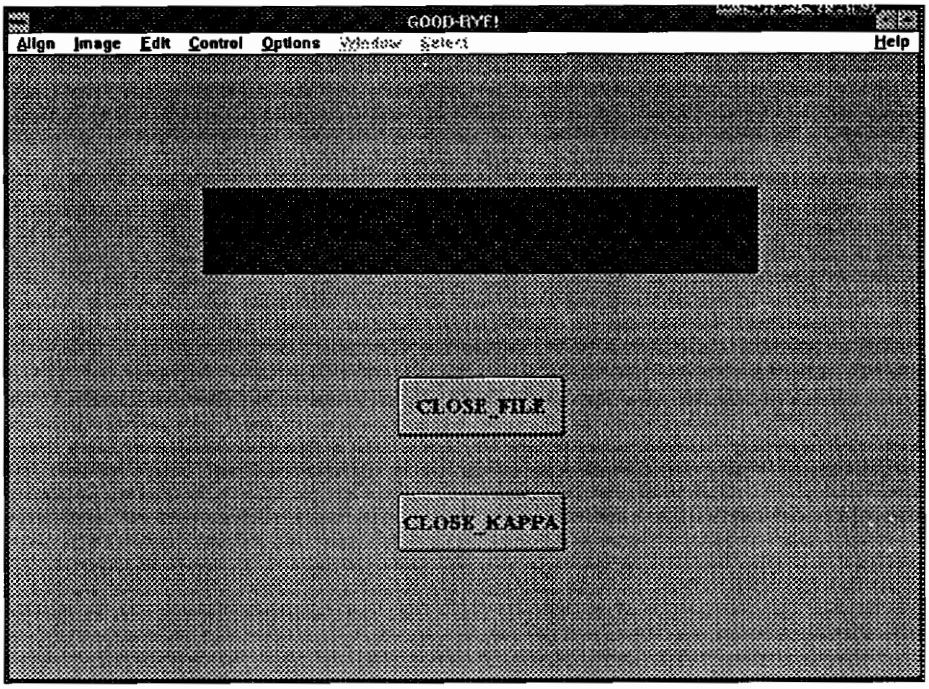


Figure 6.28 The exit window

CHAPTER 7: CONCLUSION AND DISCUSSION

In conclusion, from the results obtained, it is obvious that the research has attained the primary goals set within the limitations that were present. The knowledge based approach followed has been shown to give only an average of 5 % error. The expert system which has been developed to demonstrate this method has also been efficient enough to make use of the formulae developed and produce correct results as intended.

The knowledge based approach has thus proved to be a viable solution for procuring a quick estimate in a matter of minutes, which is also accurate enough for an initial quote. Incorporated as an expert system, it would also be faster than most human experts in giving such an estimate, and would save the time of human estimators for more important tasks.

There are of course a few improvements which can be made to the procedure and the expert system if more data and an experienced human expert were available. The system could be expanded more, removing the hardwired values and providing links to databases or spreadsheets containing constantly updated cost data. This improvement may be extremely useful if it were to be used for a commercial purpose. The formulae could be also worked on some more to improve accuracy with the use of a few more complete sets of data which include plans and the accompanying order sheets containing the exact input values. The formulae developed for the roof may also be worked on some more to improve the accuracy. The range of error is mostly due to the roof estimates, as already seen from the data. The improvement of the roof estimating formulae would certainly reduce overall error. Lastly, methods may be developed to estimate plumbing material costs, and the costs of electrical supplies which were left out in this project. These are the some of the major improvements which can be made and may provide opportunities for future research and development work.

REFERENCES

Al-Yousefi, Abdulaziz S. (1992) "Computer assisted budgeting for buildings", SAVE proceedings v 27. p 220-228, Phoenix, AZ, USA, May 31 - June 3.

Baker, Maxwell C. (1980) *Roofs design, application and maintenance,* Multiscience Publications Limited.

Baldwin, A.; Oteifa, S. A. (1991) "The application of protocol analysis in the development of knowledge based systems for contractors' resource based estimating", Second International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, p 65-70, Montreal, Quebec, Canada, September 3-5.

Christian, J. (1991) "Cost information in construction before design and scheduling using expert systems", Second International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, p 49-55, Montreal, Quebec, Canada, September 3-5.

Dagostino, Frank R. (1989) *Estimating in building construction,* Prentice Hall, NJ, USA.

Geary, Donald (1978) *Roofs and siding: A practical guide,* Reston Publishing Company, Inc.

Geddes, Spence (1985) *Estimating for building and civil engineering works,* Butterworths.

Ignizio, James P. (1991) *Introduction to expert systems,* McGraw Hill, Inc.

McCaffer, Ronald (1984) *Estimating and tendering for civil engineering works,* Granada.

Moselhi, O.; Hegazy, T.; Fazio, P. (1991) "Markup estimation using AI methodology", Second International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, p 257-266, Montreal, Quebec, Canada, September 3-5.

NBA and TRADA (1980) *Manual of timber frame housing,* The Construction Press.

Perkins, P. H. (1973) *Floors: Construction and finishes,* Cement and concrete association.

Petri, Robert W. (1979) *Construction estimating,* Reston Publishing Company, Inc.

Pulver, H. E. (1940) *Construction estimates and costs,* McGraw-Hill Book Company, Inc.

Shafer, Sam l.; Westney, Richard E. (1988) "Six steps to successful expert systems", Management Symposium - 1988, p 119-123, New Orleans, LA, USA, Jan 10-13.

Skitmore, Martin (1988) "Factors affecting accuracy of engineers estimates", Cost Engineering v 30 n 10 p 16-23, Morgantown, West Virginia, USA, October 1988.

Wass, Alonzo (1980) *Estimating residential construction,* Prentice Hall, Inc.

Watson, I. D.; Gunshon, G.; Brandon, P. S.; Basden, A. (1992) "An expert system that estimates budgets for residential construction projects", 17th International Conference on Applications of Artificial Intelligence in Engineering - AIENG/92, p 955-968, Waterloo, Ontario, Canada.

Wentworth, J. A. (1990) "Developing knowledge-based expert systems", VTT Symposium (Valtion Teknillinen Tutkimuskeskus), p 36-77, Espoo, Finland, June 26-28.

APPENDIX A: DESCRIPTON OF THE SYSTEM

The estimation process which has been developed has been incorporated as the knowledge base of an expert system. The knowledge base consists primarily of a tree structure and a rule set. The tree structure (class structure) contains the setup and relationships of and between the various components of the building and their properties. The rule set contains the estimation formulae developed in encoded form. Other than this, the parts into which the data was divided were also formed as separate files. That is, the areas of roof, walls, floors and others were considered separately and each was incorporated into a separate file, with a main file to control the component files. The process is explained in detail from the perspective of the expert system in the following material.

A.1 Scope of the System

The scope and limits of the system have been built on the principle that the primary goals of this thesis have been to:

1. Develop a method of producing a quick, approximate initial estimate for manufacturing and supplying prefabricated residential housing material; and
2. Demonstrate the advantages of employing an expert system as the delivery platform.

Primary importance was paid to developing estimation processes for the areas of floors, walls, and roofs over that of others. Estimation is basically done in the system for the following areas:

- Floor systems;
- Roof system;
- Wall systems;
- Garage system;
- Exterior and Interior trims;
- Insulation;
- Stairways;
- Decking;
- Underlayment;
- Shelving;

- Cabinets and Vanities; and
- Approximate estimations for doors and windows if required.

Doors and windows were not paid importance to since it would be easier and quicker to get the list of types and quantity of door and window units required and get the amount from a spreadsheet since their costs are fixed. The same is the case for wiring and plumbing material. The costs are fixed and have no labor costs associated in order to give a variable. The only variable in their case would be the markup cost, which is a fixed percentage used for the whole estimate.

A.2 Structure

The structure in any expert system is usually in the form of a tree. The floors, walls, etc. which are split into types are called classes. Each class is split into sub-classes, which may be split further. The final components are called instances. The structure of the complete expert system is shown in Fig. A.1. As seen, walls for example, form a class, and the types of walls, internal and external, are formed as subclasses, with the values of the walls for each floor being entered as instances to the subclasses. The properties and values of the walls are entered into the system structure as slot values to the instances or classes. In a similar form, the structures for the other components of the building are formed and the combined structure has been shown in Fig A.1. The figure shows only the basic structure in which the data is arranged in the knowledge base. Their properties are not visible.

As already mentioned, the system file structure has been developed in a modular form conforming with the sections into which the data has been split. That is, the information about floors, walls, roof and other miscellaneous parts of the building have been split up and placed in separate files which are activated from the main file as required. This has been done in this format since that is how the estimation process has been developed. Other advantages to this are:

1. Facilitates troubleshooting;
2. Easier to update information;

3. System operates faster if estimation is not required for all sections of the building; and
4. Allows the system to operate on a machine with minimum memory.

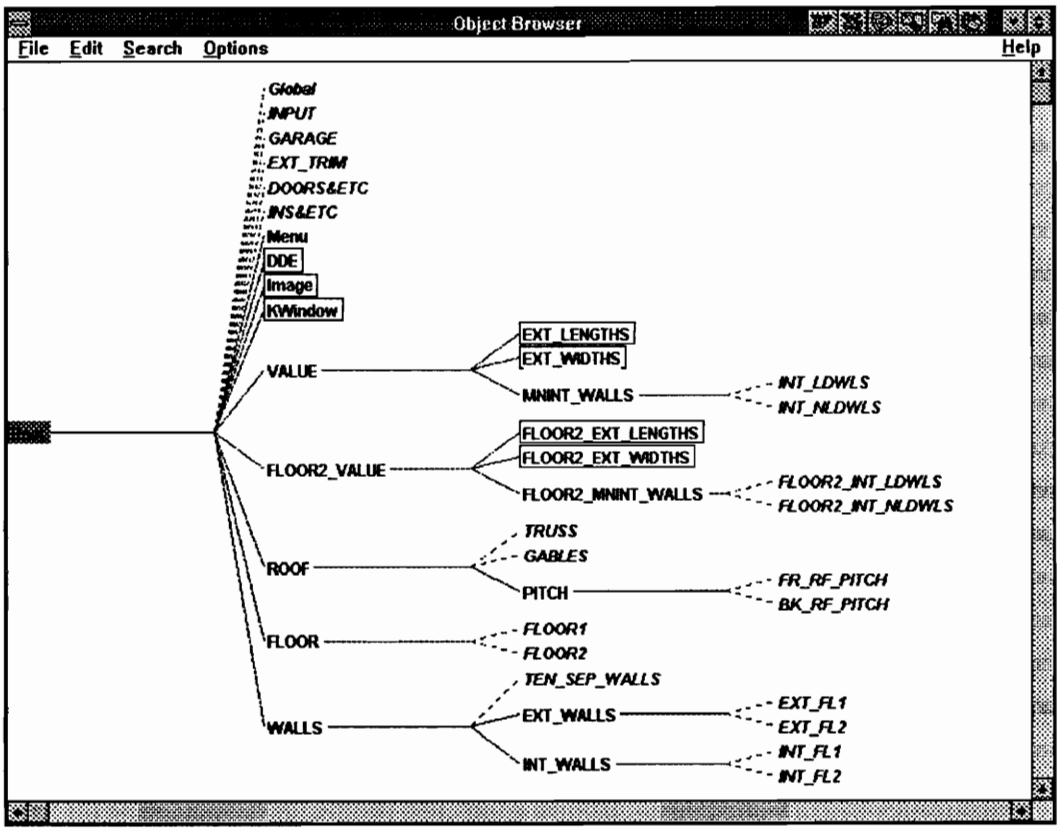


Figure A.1 Complete Class Structure of the ES.

The common information such as the dimensions of the building and the garage dimensions are stored in the main file (Main.kal), which is also the one from which the four other files (Roof.kal, Floors.kal, Walls.kal and Others.kal) are started if found necessary for the estimation. Once the other files are started at the beginning, they are "added" to the working file, and may be accessed at any time from the main user interface. This allows for the user to make any changes in the input or default values to change the

final estimate if required. On exiting, the option of saving the input values, along with the whole file which was used for the estimation, is given to the user. The complete class structure of the ES is given in Appendix B.

A.3 Rule set

The rule set consists of all the formulae developed for cost estimation. The formulae have been encoded into the form of rules, each of which contains all the formulae for that component of the building. For example, the rule for roofs contains all the formulae pertaining to calculating the cost of manufacturing the roof. As the files are activated, their rules are added to the working file from which the calculation takes place. Execution of the formulae is performed by running the rules. This is done by means of the inference engine. The inference engine runs the rules which execute the formulae based on the input values in the slots of the classes. In other words, the information fed is in processed using the rules and the results are shown as the output. An example of a rule, taken from the system, is given below.

```
MakeRule( R_PORCH_RAIL, [],
   KnownValue?( TOP, VALUE ) And KnownValue?( LEFT, VALUE ),
   Global:ESTIMATE += ( TL1:VALUE + TL2:VALUE + TL3:VALUE + TL4:VALUE
                        + LW1:VALUE + LW2:VALUE + LW3:VALUE
                        + LW4:VALUE ) * 2 * 1.1 );
```

In the above rule, the first part, "KnownValue?..", is used to find whether there are values entered for the items listed in the latter part of the "KnownValue?..", and if yes, to proceed with the execution of the rule. The latter part of the rule is the actual execution part where, based on the "If...Then..." conditions, estimates are made and added to a global point which contains the total estimate to obtain the final estimated value. The rules may be equated to subroutines in conventional programming with the major difference that it is not imperative for every rule listed and called to be used in order for the results to be produced. If some values are not there, the rule, though listed and called

for by a function, may be passed over, or in some cases may use default values to fill in gaps in the input. In the case of the example given above,

The formula given above is a combination of conventional cost estimation formulae and improvised formulae developed based on real life data from an existing firm. The same data has also been used for verification and validation of the system comes. That is, testing the accuracy and reliability of the system by comparing the results produced by it with the results that the human experts had produced.

A.4 Inference engine

The inference engine used is a built-in part of the Kappa shell. It is activated and run using functions. An example function for this is given below.

```
MakeFunction( GO, [],
{
Global:ESTIMATE = NULL;
ForwardChain([NOASSERT], NULL) ;
} );
```

In the above example, the "ESTIMATE" slot in the Global class is first set to zero as an initializing step. Then, the "ForwardChain([NOASSERT], NULL) ;"step is used to run all the rules in the rule base. The function basically makes the inference engine use ForwardChaining procedure to run all the rules and obtain the result. Since the names of the rules to be run are not specified, all the rules in the rule base are executed in the order that they are listed. Though both ForwardChaining or BackwardChaining could have been used, since the problem dealt with is of a numerical nature, ForwardChaining has been used in the system.

A.5 User interface

The user interface for the system has been developed using the order form which was available with the other data. It has been set up such that the first window, "MAIN",

allows the common input such as the dimensions of the building, type of foundation, etc. to be input first into the system. As previously mentioned, it also provides for the user to activate the files necessary for the estimation job at hand when he "begins". Once the common input has been acquired, the specific information for the various sections of "ROOF", "WALLS", "FLOORS" and "OTHERS" may be input into the system using the appropriate buttons in the "MAIN" window. One thing to be remembered is that unlike conventional software, it is not absolutely necessary that all the blanks spaces have to be filled. The system will still give results based on incomplete information. The user need only fill in the information that is relevant, using the information given in the order form. The system disregards the information which is irrelevant for obtaining the estimate required by the user and substitutes default values where required. The interface operations have been set up to operate via functions written in the system. These functions will open new interface windows, close or destroy old ones, and in short, run the user through the complete system. Some example functions are given below.

```
MakeFunction( WALLS_FILE, [],
{
InterpretFile(walls.kal);
} );
```

The above function appends the walls.kal file to the working file. One more example of a function is given below.

```
MakeInstance( GOODBYE, KSession );
GOODBYE:X = 10;
GOODBYE:Y = 10;
GOODBYE:Title = "GOOD-BYE!";
GOODBYE:SessionNumber = 7;
```

```
GOODBYE:Width = GetScreenWidth() - 20;
GOODBYE:Height = GetScreenHeight() - 20;
GOODBYE:Visible = FALSE;
GOODBYE:State = HIDDEN;
SetValue( GOODBYE:BackgroundColor, 0, 255, 255 );
GOODBYE:Menu = TRUE;
GOODBYE:Titlebar = TRUE;
GOODBYE:Sizebox = TRUE;
ResetWindow( GOODBYE );
MakeInstance( Close_file, Button );
Close_file:SessionNumber = 7;
Close_file:Title = CLOSE_FILE;
Close_file:Visible = TRUE;
Close_file:X = GetScreenWidth()/2 - 70;
Close_file:Y = GetScreenHeight()/2 - 25;
SetValue( Close_file:ForegroundColor, 0, 0, 0 );
SetValue( Close_file:BackgroundColor, 192, 192, 192 );
Close_file:Width = 140;
Close_file:Height = 50;
Close_file:Font = "Times New Roman";
Close_file:TextSize = 14;
Close_file:Bold = TRUE;
Close_file:Underline = FALSE;
Close_file:Italic = FALSE;
Close_file:StrikeOut = FALSE;
Close_file:Action = CLOSE_FILE;
Close_file:FunctionKey = NULL;
ResetImage( Close_file );
```

```
MakeInstance( Close_kappa, Button );
Close_kappa:SessionNumber = 7;
Close_kappa:Title = CLOSE_KAPPA;
Close_kappa:Visible = TRUE;
Close_kappa:X = GetScreenWidth()/2 - 70;
Close_kappa:Y = GetScreenHeight()/2 + GetScreenHeight()/6 - 25;
SetValue( Close_kappa:ForegroundColor, 0, 0, 0 );
SetValue( Close_kappa:BackgroundColor, 192, 192, 192 );
Close_kappa:Width = 140;
Close_kappa:Height = 50;
Close_kappa:Font = "Times New Roman";
Close_kappa:TextSize = 14;
Close_kappa:Bold = TRUE;
Close_kappa:Underline = FALSE;
Close_kappa:Italic = FALSE;
Close_kappa:StrikeOut = FALSE;
Close_kappa:Action = CLOSE_KAPPA;
Close_kappa:FunctionKey = NULL;
ResetImage( Close_kappa );
MakeInstance( Day, Text );
Day:SessionNumber = 7;
Day:Title = "HAVE A NICE DAY!";
Day:Visible = TRUE;
Day:X = GetScreenWidth()/2 - 235;
Day:Y = GetScreenHeight()/2 - GetScreenHeight()/4 - 37;
SetValue( Day:ForegroundColor, 0, 0, 0 );
SetValue( Day:BackgroundColor, 64, 0, 128 );
Day:Width = 470;
```

```
Day:Height = 74;
Day:Font = "Times New Roman";
Day:TextSize = 36;
Day:Bold = TRUE;
Day:Underline = FALSE;
Day:Italic = TRUE;
Day:StrikeOut = FALSE;
Day:ShowBorder = TRUE;
Day:Justification = CENTER;
ResetImage( Day );
ShowWindow(GOODBYE);
} );
```

The above function creates a window, "GOODBYE", just before exiting and closing the file or closing the Kappa session itself. The above examples give an idea of how functions have been used. The system which has been developed runs almost entirely on functions. All manipulations between the interface windows are done using functions.

A.6 Operation

The operation of the system is in the following manner. The main user interface first used to get the basic information into the system. It is also used to append the other necessary files for estimation. The specific input about the parts of the building for which an estimate is required is then fed into the system by the user. Once all the input has been fed in, the inference engine is activated to run the rules. The result is then displayed as a total estimate. Kappa also has a built-in explanation function by means of which the result can be traced back to the rules by which it was calculated.

One other thing which is possible with the system is that it allows for interaction with it to change not only the input values, but also the markup values to change the final

estimate. This would be required to make the prepared bid competitive enough to get the contract. This has been incorporated in the system in the form of a separate interface, the option for which can be accessed at any time from the RESULT screen. This is also the reason why the file with all the input values can be saved separately. That way, all the values do not have to be input all over again. Only some of the values need to be changed and the recalculation done. The main reason for this facility is to compensate for the major drawback that any Artificial Intelligence system lacks, and which is what makes them still have that appellation "Artificial" to their name - "Intuition". Something only humans can provide. This comes into play especially when the bids have to be prepared. There would be changes and fluctuations in the market which cannot be quantified or put or paper. These can be compensated only by intuition. Without the facility to change the output of the system in such a manner as described here, the bid may quite very well be lost. It is also at this point that one of the major advantages of the expert system over the conventional systems is shown. There is no need, as in conventional systems, to keep reentering all the input values over and over again till a satisfactory result is got. What would take hours, or even days depending on the volume of input, using a conventional system, takes only a few minutes using the expert system.

A.7 Summary

This appendix has basically explained how the cost estimation knowledge and formulae developed have been incorporated into the expert system. It has also explained about the user interface and operation of the system. The formulae which were actually incorporated as rules in the system have been listed out and explained in Appendix B. The assumptions which were made while developing the system have also been listed and explained there.

APPENDIX B: CLASS STRUCTURE AND RULE BASE

The structure and rule set have been explained in Appendix A. The representation of this structure and the rules in the ES which has been developed is shown below. The structure is in the form of classes and instances. The formulae which were developed and incorporated as rules have also been given here.

B.1: THE CLASSES IN THE SYSTEM

```
    /************************************
     **** CLASS: VALUE
     ************************************/
MakeClass( VALUE, Root );
MakeSlot( VALUE:VALUE );
SetSlotOption( VALUE:VALUE, VALUE_TYPE, NUMBER );
SetSlotOption( VALUE:VALUE, MINIMUM_VALUE, 0 );
SetSlotOption( VALUE:VALUE, MAXIMUM_VALUE, 5000 );
VALUE:VALUE = 0;
MakeSlot( VALUE:TOTAL_LENGTH );
SetSlotOption( VALUE:TOTAL_LENGTH, INHERIT, FALSE );
SetSlotOption( VALUE:TOTAL_LENGTH, VALUE_TYPE, NUMBER );
SetSlotOption( VALUE:TOTAL_LENGTH, MINIMUM_VALUE, 0 );
SetSlotOption( VALUE:TOTAL_LENGTH, MAXIMUM_VALUE, 500 );
MakeSlot( VALUE:TOTAL_WIDTH );
SetSlotOption( VALUE:TOTAL_WIDTH, INHERIT, FALSE );
SetSlotOption( VALUE:TOTAL_WIDTH, VALUE_TYPE, NUMBER );
SetSlotOption( VALUE:TOTAL_WIDTH, MINIMUM_VALUE, 0 );
SetSlotOption( VALUE:TOTAL_WIDTH, MAXIMUM_VALUE, 500 );
MakeSlot( VALUE:TOTAL_AREA );
SetSlotOption( VALUE:TOTAL_AREA, INHERIT, FALSE );
SetSlotOption( VALUE:TOTAL_AREA, VALUE_TYPE, NUMBER );
SetSlotOption( VALUE:TOTAL_AREA, MINIMUM_VALUE, 0 );
SetSlotOption( VALUE:TOTAL_AREA, MAXIMUM_VALUE, 250000 );

    /************************************
     **** CLASS: EXT_LENGTHS
     ************************************/
MakeClass( EXT_LENGTHS, VALUE );
```

```
    /************************************
     **** CLASS: TOP
     ************************************/
MakeClass( TOP, EXT_LENGTHS );

    /************************************
     **** CLASS: BOTTOM
     ************************************/
MakeClass( BOTTOM, EXT_LENGTHS );

    /************************************
     **** CLASS: EXT_WIDTHS
     ************************************/
MakeClass( EXT_WIDTHS, VALUE );

    /************************************
     **** CLASS: LEFT
     ************************************/
MakeClass( LEFT, EXT_WIDTHS );

    /************************************
     **** CLASS: RIGHT
     ************************************/
MakeClass( RIGHT, EXT_WIDTHS );

    /************************************
     **** CLASS: MNINT_WALLS
     ************************************/
MakeClass( MNINT_WALLS, VALUE );
MakeSlot( MNINT_WALLS:TOTAL_INT_LENGTH );
SetSlotOption( MNINT_WALLS:TOTAL_INT_LENGTH, INHERIT, FALSE );
SetSlotOption( MNINT_WALLS:TOTAL_INT_LENGTH, VALUE_TYPE, NUMBER
);
SetSlotOption( MNINT_WALLS:TOTAL_INT_LENGTH, MINIMUM_VALUE, 0 );
SetSlotOption( MNINT_WALLS:TOTAL_INT_LENGTH, MAXIMUM_VALUE, 500
);

    /************************************
     **** CLASS: FLOOR2_VALUE
     ************************************/
MakeClass( FLOOR2_VALUE, Root );
MakeSlot( FLOOR2_VALUE:VALUE );
```

```
SetSlotOption( FLOOR2_VALUE:VALUE, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR2_VALUE:VALUE, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR2_VALUE:VALUE, MAXIMUM_VALUE, 5000 );
FLOOR2_VALUE:VALUE = 0;
MakeSlot( FLOOR2_VALUE:TOTAL_LENGTH );
SetSlotOption( FLOOR2_VALUE:TOTAL_LENGTH, INHERIT, FALSE );
SetSlotOption( FLOOR2_VALUE:TOTAL_LENGTH, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR2_VALUE:TOTAL_LENGTH, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR2_VALUE:TOTAL_LENGTH, MAXIMUM_VALUE, 500 );
MakeSlot( FLOOR2_VALUE:TOTAL_WIDTH );
SetSlotOption( FLOOR2_VALUE:TOTAL_WIDTH, INHERIT, FALSE );
SetSlotOption( FLOOR2_VALUE:TOTAL_WIDTH, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR2_VALUE:TOTAL_WIDTH, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR2_VALUE:TOTAL_WIDTH, MAXIMUM_VALUE, 500 );
MakeSlot( FLOOR2_VALUE:TOTAL_AREA );
SetSlotOption( FLOOR2_VALUE:TOTAL_AREA, INHERIT, FALSE );
SetSlotOption( FLOOR2_VALUE:TOTAL_AREA, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR2_VALUE:TOTAL_AREA, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR2_VALUE:TOTAL_AREA, MAXIMUM_VALUE, 250000 );

    /************************************
     **** CLASS: FLOOR2_EXT_LENGTHS
     ************************************/
MakeClass( FLOOR2_EXT_LENGTHS, FLOOR2_VALUE );

    /************************************
     **** CLASS: FLOOR2_TOP
     ************************************/
MakeClass( FLOOR2_TOP, FLOOR2_EXT_LENGTHS );

    /************************************
     **** CLASS: FLOOR2_BOTTOM
     ************************************/
MakeClass( FLOOR2_BOTTOM, FLOOR2_EXT_LENGTHS );

    /************************************
     **** CLASS: FLOOR2_EXT_WIDTHS
     ************************************/
MakeClass( FLOOR2_EXT_WIDTHS, FLOOR2_VALUE );

    /************************************
     **** CLASS: FLOOR2_LEFT
     ************************************/
```

```
MakeClass( FLOOR2_LEFT, FLOOR2_EXT_WIDTHS );

    /************************************
     **** CLASS: FLOOR2_RIGHT
     ************************************/
MakeClass( FLOOR2_RIGHT, FLOOR2_EXT_WIDTHS );

    /************************************
     **** CLASS: FLOOR2_MNINT_WALLS
     ************************************/
MakeClass( FLOOR2_MNINT_WALLS, FLOOR2_VALUE );
MakeSlot( FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH );
SetSlotOption( FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH, INHERIT, FALSE
);
SetSlotOption( FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH, VALUE_TYPE,
NUMBER );
SetSlotOption(                FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH,
MINIMUM_VALUE, 0 );
SetSlotOption(                FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH,
MAXIMUM_VALUE, 500 );

    /************************************
     **** CLASS: ROOF
     ************************************/
MakeClass( ROOF, Root );
MakeSlot( ROOF:RAFTER_SIZE );
SetSlotOption( ROOF:RAFTER_SIZE, INHERIT, FALSE );
ROOF:RAFTER_SIZE = NULL;
SetSlotOption( ROOF:RAFTER_SIZE, IMAGE, Rafter_size );
MakeSlot( ROOF:JOIST_SIZE );
SetSlotOption( ROOF:JOIST_SIZE, INHERIT, FALSE );
ROOF:JOIST_SIZE = NULL;
SetSlotOption( ROOF:JOIST_SIZE, IMAGE, Joist_size );
MakeSlot( ROOF:RF_JT_OC );
SetSlotOption( ROOF:RF_JT_OC, INHERIT, FALSE );
SetSlotOption( ROOF:RF_JT_OC, ALLOWABLE_VALUES, 16, 24 );
ROOF:RF_JT_OC = 16;
SetSlotOption( ROOF:RF_JT_OC, IMAGE, Roof_jst_oc );
MakeSlot( ROOF:OVERHANG );
SetSlotOption( ROOF:OVERHANG, INHERIT, FALSE );
SetSlotOption( ROOF:OVERHANG, ALLOWABLE_VALUES, 12, 24, 36 );
ROOF:OVERHANG = 12;
SetSlotOption( ROOF:OVERHANG, IMAGE, Eave_overhang );
```

```
MakeSlot( ROOF:RF_CONST_TYPE );
SetSlotOption( ROOF:RF_CONST_TYPE, INHERIT, FALSE );
SetSlotOption( ROOF:RF_CONST_TYPE, ALLOWABLE_VALUES, TRUSS, STICK
);
ROOF:RF_CONST_TYPE = STICK;
SetSlotOption( ROOF:RF_CONST_TYPE, IMAGE, Roof_const_type );
MakeSlot( ROOF:SHEATHING_TYPE );
SetSlotOption( ROOF:SHEATHING_TYPE, INHERIT, FALSE );
SetSlotOption( ROOF:SHEATHING_TYPE, ALLOWABLE_VALUES,
ORIENTED_STRAND_BOARD, 15_BY_32_PLYWOOD, 19_BY_32_PLYWOOD,
15_BY_32_FIRE_RETARDANT_PLYWOOD );
SetSlotOption( ROOF:SHEATHING_TYPE, IMAGE, Roof_sheathing );
MakeSlot( ROOF:SHINGLES_TYPE );
SetSlotOption( ROOF:SHINGLES_TYPE, INHERIT, FALSE );
SetSlotOption( ROOF:SHINGLES_TYPE, ALLOWABLE_VALUES,
FIBREGLASS_WITH_15FELT, 15FELT_ONLY );
SetSlotOption( ROOF:SHINGLES_TYPE, IMAGE, Roof_shingles );
MakeSlot( ROOF:ROOF_TRUSSES );
SetSlotOption( ROOF:ROOF_TRUSSES, INHERIT, FALSE );
SetSlotOption( ROOF:ROOF_TRUSSES, VALUE_TYPE, NUMBER );
SetSlotOption( ROOF:ROOF_TRUSSES, MINIMUM_VALUE, 0 );
SetSlotOption( ROOF:ROOF_TRUSSES, MAXIMUM_VALUE, 99999 );
MakeSlot( ROOF:RFING_MTLS );
SetSlotOption( ROOF:RFING_MTLS, INHERIT, FALSE );
SetSlotOption( ROOF:RFING_MTLS, VALUE_TYPE, NUMBER );
SetSlotOption( ROOF:RFING_MTLS, MINIMUM_VALUE, 0 );
SetSlotOption( ROOF:RFING_MTLS, MAXIMUM_VALUE, 99999 );
MakeSlot( ROOF:FL&SHOSB );
SetSlotOption( ROOF:FL&SHOSB, INHERIT, FALSE );
SetSlotOption( ROOF:FL&SHOSB, VALUE_TYPE, NUMBER );
SetSlotOption( ROOF:FL&SHOSB, MINIMUM_VALUE, 0 );
SetSlotOption( ROOF:FL&SHOSB, MAXIMUM_VALUE, 99999 );
MakeSlot( ROOF:GAB_PROJ );
SetSlotOption( ROOF:GAB_PROJ, INHERIT, FALSE );
SetSlotOption( ROOF:GAB_PROJ, VALUE_TYPE, NUMBER );
SetSlotOption( ROOF:GAB_PROJ, MINIMUM_VALUE, 0 );
SetSlotOption( ROOF:GAB_PROJ, MAXIMUM_VALUE, 9999 );

    /************************************
     **** CLASS: PITCH
     ************************************/
MakeClass( PITCH, ROOF );
MakeSlot( PITCH:PITCH_HT );
```

```
SetSlotOption( PITCH:PITCH_HT, VALUE_TYPE, NUMBER );
SetSlotOption( PITCH:PITCH_HT, MINIMUM_VALUE, 1 );
SetSlotOption( PITCH:PITCH_HT, MAXIMUM_VALUE, 12 );

    /************************************
     **** CLASS: FLOOR
     ************************************/
MakeClass( FLOOR, Root );
MakeSlot( FLOOR:MISC );
SetSlotOption( FLOOR:MISC, INHERIT, FALSE );
SetSlotOption( FLOOR:MISC, MULTIPLE );
SetSlotOption( FLOOR:MISC, ALLOWABLE_VALUES, CONST_ADHESIVE,
WOOD_BRIDGING_1_BY_3, WOOD_BRIDGING_2_BY_SOLID,
METAL_CROSS_BRIDGING );
ClearList( FLOOR:MISC );
SetSlotOption( FLOOR:MISC, IMAGE, Misc_fl_options );
MakeSlot( FLOOR:FLOOR_SHEATH_SIZE );
SetSlotOption( FLOOR:FLOOR_SHEATH_SIZE, ALLOWABLE_VALUES,
19_BY_32_TG, 23_BY_32_TG );
MakeSlot( FLOOR:FLOOR_SHEATH_TYPE );
SetSlotOption( FLOOR:FLOOR_SHEATH_TYPE, ALLOWABLE_VALUES,
ORTD_STD_BRD, PLYWOOD );
MakeSlot( FLOOR:TRUSS_HANGERS );
SetSlotOption( FLOOR:TRUSS_HANGERS, ALLOWABLE_VALUES, YES, NO );
MakeSlot( FLOOR:TRUSS_SPACING );
SetSlotOption( FLOOR:TRUSS_SPACING, ALLOWABLE_VALUES, 16, 19.2, 24 );
MakeSlot( FLOOR:TRUSS_OPTIONS );
SetSlotOption( FLOOR:TRUSS_OPTIONS, MULTIPLE );
SetSlotOption( FLOOR:TRUSS_OPTIONS, ALLOWABLE_VALUES, CLEAR_SPAN,
WI_INTERM_BEARING, PLYWD_TRUSS_BANDS );
ClearList( FLOOR:TRUSS_OPTIONS );
MakeSlot( FLOOR:FRAME_TYPE );
SetSlotOption( FLOOR:FRAME_TYPE, ALLOWABLE_VALUES, JOIST, TRUSS );
MakeSlot( FLOOR:OCF );
SetSlotOption( FLOOR:OCF, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR:OCF, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR:OCF, MAXIMUM_VALUE, 1 );
MakeSlot( FLOOR:JST_COST );
SetSlotOption( FLOOR:JST_COST, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR:JST_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR:JST_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( FLOOR:T&G_COST );
SetSlotOption( FLOOR:T&G_COST, VALUE_TYPE, NUMBER );
```

```
SetSlotOption( FLOOR:T&G_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR:T&G_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( FLOOR:ADHESIVE_COST );
SetSlotOption( FLOOR:ADHESIVE_COST, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR:ADHESIVE_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR:ADHESIVE_COST, MAXIMUM_VALUE, 100 );
MakeSlot( FLOOR:JST_HGR_COST );
SetSlotOption( FLOOR:JST_HGR_COST, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR:JST_HGR_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR:JST_HGR_COST, MAXIMUM_VALUE, 10 );

    /************************************
     **** CLASS: WALLS
     ************************************/
MakeClass( WALLS, Root );
MakeSlot( WALLS:HEIGHT );
SetSlotOption( WALLS:HEIGHT, VALUE_TYPE, NUMBER );
SetSlotOption( WALLS:HEIGHT, MINIMUM_VALUE, 7 );
SetSlotOption( WALLS:HEIGHT, MAXIMUM_VALUE, 10 );
MakeSlot( WALLS:STUD_SIZE );
SetSlotOption( WALLS:STUD_SIZE, ALLOWABLE_VALUES, 2x4, 2x6 );
WALLS:STUD_SIZE = 2x4;
MakeSlot( WALLS:STUD_SPACING );
SetSlotOption( WALLS:STUD_SPACING, ALLOWABLE_VALUES, 12, 16 );
WALLS:STUD_SPACING = 16;
MakeSlot( WALLS:BOTTOM_PLATE );
SetSlotOption( WALLS:BOTTOM_PLATE, ALLOWABLE_VALUES, SINGLE,
DOUBLE, PRESS_TREATED );
WALLS:BOTTOM_PLATE = SINGLE;
MakeSlot( WALLS:OUTWL_COST );
SetSlotOption( WALLS:OUTWL_COST, VALUE_TYPE, NUMBER );
SetSlotOption( WALLS:OUTWL_COST, MINIMUM_VALUE, 0 );
SetSlotOption( WALLS:OUTWL_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( WALLS:INTWL_COST );
SetSlotOption( WALLS:INTWL_COST, VALUE_TYPE, NUMBER );
SetSlotOption( WALLS:INTWL_COST, MINIMUM_VALUE, 0 );
SetSlotOption( WALLS:INTWL_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( WALLS:PTBTPLT_COST );
SetSlotOption( WALLS:PTBTPLT_COST, VALUE_TYPE, NUMBER );
SetSlotOption( WALLS:PTBTPLT_COST, MINIMUM_VALUE, 0 );
SetSlotOption( WALLS:PTBTPLT_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( WALLS:BRACING_COST );
SetSlotOption( WALLS:BRACING_COST, VALUE_TYPE, NUMBER );
```

```
SetSlotOption( WALLS:BRACING_COST, MINIMUM_VALUE, 0 );
SetSlotOption( WALLS:BRACING_COST, MAXIMUM_VALUE, 9999 );

    /************************************
     **** CLASS: EXT_WALLS
     ************************************/
MakeClass( EXT_WALLS, WALLS );
SetSlotOption( EXT_WALLS:STUD_SIZE, IMAGE, Ext_wls_std_size );
SetSlotOption( EXT_WALLS:STUD_SPACING, IMAGE, Ext_wls_std_spacing );
MakeSlot( EXT_WALLS:SHEATHING );
SetSlotOption( EXT_WALLS:SHEATHING, INHERIT, FALSE );
SetSlotOption( EXT_WALLS:SHEATHING, ALLOWABLE_VALUES,
1_BY_2_INTR_NCBR, 7_BY_16_OR_STRBRD, 1_BY_2_FO_INS_TBR,
3_BY_4_FO_INS_TBR, 1_1_BY_4_INS_TBR );
EXT_WALLS:SHEATHING = 1_BY_2_INTR_NCBR;
SetSlotOption( EXT_WALLS:SHEATHING, IMAGE, Ext_wls_sheath );
MakeSlot( EXT_WALLS:MISC );
SetSlotOption( EXT_WALLS:MISC, INHERIT, FALSE );
SetSlotOption( EXT_WALLS:MISC, MULTIPLE );
SetSlotOption( EXT_WALLS:MISC, ALLOWABLE_VALUES, GAR_WLS_SAME,
PLYWOOD_CRBR );
ClearList( EXT_WALLS:MISC );
SetSlotOption( EXT_WALLS:MISC, IMAGE, Ext_wls_misc );

    /************************************
     **** CLASS: INT_WALLS
     ************************************/
MakeClass( INT_WALLS, WALLS );
SetSlotOption( INT_WALLS:STUD_SPACING, IMAGE, Int_wls_std_spacing );
```

B.2: THE INSTANCES IN THE SYSTEM

```
MakeSlot( Global:ESTIMATE );
SetSlotOption( Global:ESTIMATE, VALUE_TYPE, NUMBER );
SetSlotOption( Global:ESTIMATE, MINIMUM_VALUE, 0 );
SetSlotOption( Global:ESTIMATE, MAXIMUM_VALUE, 9999999999 );
Global:ESTIMATE = 0;
SetSlotOption( Global:ESTIMATE, IMAGE, Answer );
MakeSlot( Global:TEMP );
SetSlotOption( Global:TEMP, VALUE_TYPE, NUMBER );
SetSlotOption( Global:TEMP, MINIMUM_VALUE, 0 );
SetSlotOption( Global:TEMP, MAXIMUM_VALUE, 9999999999 );
```

```
Global:TEMP = 0;
MakeSlot( Global:MARKUP );
SetSlotOption( Global:MARKUP, VALUE_TYPE, NUMBER );
SetSlotOption( Global:MARKUP, MINIMUM_VALUE, 0 );
SetSlotOption( Global:MARKUP, MAXIMUM_VALUE, 500 );
Global:MARKUP = 47;

    /************************************
     **** INSTANCE: SESSION
     ************************************/
SESSION:X = 20;
SESSION:Y = 20;
SESSION:Title = SESSION;
SESSION:SessionNumber = 0;
SESSION:Width = 600;
SESSION:Height = 440;
SESSION:Menu = TRUE;
SESSION:Visible = TRUE;
SESSION:State = ICON;
SetValue( SESSION:BackgroundColor, 0, 0, 255 );
SESSION:Titlebar = TRUE;
SESSION:Sizebox = TRUE;
ResetWindow( SESSION );

    /************************************
     **** INSTANCE: INPUT
     ************************************/
MakeInstance( INPUT, Root );
MakeSlot( INPUT:STOREYS );
SetSlotOption( INPUT:STOREYS, ALLOWABLE_VALUES, 1, 2 );
INPUT:STOREYS = 1;
SetSlotOption( INPUT:STOREYS, IMAGE, SingleListBox1, RadioButtonGroup1 );
MakeSlot( INPUT:FOUNDATION );
SetSlotOption( INPUT:FOUNDATION, ALLOWABLE_VALUES, CRAWL,
BASEMENT, SLAB );
INPUT:FOUNDATION = CRAWL;
SetSlotOption( INPUT:FOUNDATION, IMAGE, RadioButtonGroup2 );

    /************************************
     **** INSTANCE: TL1
     ************************************/
MakeInstance( TL1, TOP );
SetSlotOption( TL1:VALUE, IMAGE, Tlength1 );
```

```
    /************************************
     **** INSTANCE: TL2
     ************************************/
MakeInstance( TL2, TOP );
SetSlotOption( TL2:VALUE, IMAGE, Tlength2 );

    /************************************
     **** INSTANCE: TL3
     ************************************/
MakeInstance( TL3, TOP );
SetSlotOption( TL3:VALUE, IMAGE, Tlength3 );

    /************************************
     **** INSTANCE: TL4
     ************************************/
MakeInstance( TL4, TOP );
SetSlotOption( TL4:VALUE, IMAGE, Tlength4 );

    /************************************
     **** INSTANCE: BL1
     ************************************/
MakeInstance( BL1, BOTTOM );
SetSlotOption( BL1:VALUE, IMAGE, Blength1 );

    /************************************
     **** INSTANCE: BL2
     ************************************/
MakeInstance( BL2, BOTTOM );
SetSlotOption( BL2:VALUE, IMAGE, Blength2 );

    /************************************
     **** INSTANCE: BL3
     ************************************/
MakeInstance( BL3, BOTTOM );
SetSlotOption( BL3:VALUE, IMAGE, Blength3 );

    /************************************
     **** INSTANCE: BL4
     ************************************/
MakeInstance( BL4, BOTTOM );
SetSlotOption( BL4:VALUE, IMAGE, Blength4 );
```

```
    /************************************
     **** INSTANCE: LW1
     ************************************/
MakeInstance( LW1, LEFT );
SetSlotOption( LW1:VALUE, IMAGE, Lwidth1 );

    /************************************
     **** INSTANCE: LW2
     ************************************/
MakeInstance( LW2, LEFT );
SetSlotOption( LW2:VALUE, IMAGE, Lwidth2 );

    /************************************
     **** INSTANCE: LW3
     ************************************/
MakeInstance( LW3, LEFT );
SetSlotOption( LW3:VALUE, IMAGE, Lwidth3 );

    /************************************
     **** INSTANCE: LW4
     ************************************/
MakeInstance( LW4, LEFT );
SetSlotOption( LW4:VALUE, IMAGE, Lwidth4 );

    /************************************
     **** INSTANCE: RW1
     ************************************/
MakeInstance( RW1, RIGHT );
SetSlotOption( RW1:VALUE, IMAGE, Rwidth1 );

    /************************************
     **** INSTANCE: RW2
     ************************************/
MakeInstance( RW2, RIGHT );
SetSlotOption( RW2:VALUE, IMAGE, Rwidth2 );

    /************************************
     **** INSTANCE: RW3
     ************************************/
MakeInstance( RW3, RIGHT );
SetSlotOption( RW3:VALUE, IMAGE, Rwidth3 );

    /************************************
```

```
     **** INSTANCE: RW4
     ***********************************/
MakeInstance( RW4, RIGHT );
SetSlotOption( RW4:VALUE, IMAGE, Rwidth4 );

     /***********************************
     **** INSTANCE: INT_LDWLS
     ***********************************/
MakeInstance( INT_LDWLS, MNINT_WALLS );
SetSlotOption( INT_LDWLS:VALUE, IMAGE, Ldwls );

     /***********************************
     **** INSTANCE: INT_NLDWLS
     ***********************************/
MakeInstance( INT_NLDWLS, MNINT_WALLS );
SetSlotOption( INT_NLDWLS:VALUE, IMAGE, Nldwls );

     /***********************************
     **** INSTANCE: FLOOR2_TL1
     ***********************************/
MakeInstance( FLOOR2_TL1, FLOOR2_TOP );
SetSlotOption( FLOOR2_TL1:VALUE, IMAGE, F2tlength1 );

     /***********************************
     **** INSTANCE: FLOOR2_TL2
     ***********************************/
MakeInstance( FLOOR2_TL2, FLOOR2_TOP );
SetSlotOption( FLOOR2_TL2:VALUE, IMAGE, F2tlength2 );

     /***********************************
     **** INSTANCE: FLOOR2_TL3
     ***********************************/
MakeInstance( FLOOR2_TL3, FLOOR2_TOP );
SetSlotOption( FLOOR2_TL3:VALUE, IMAGE, F2tlength3 );

     /***********************************
     **** INSTANCE: FLOOR2_TL4
     ***********************************/
MakeInstance( FLOOR2_TL4, FLOOR2_TOP );
SetSlotOption( FLOOR2_TL4:VALUE, IMAGE, F2tlength4 );

     /***********************************
     **** INSTANCE: FLOOR2_BL1
```

```
    ************************************/
MakeInstance( FLOOR2_BL1, FLOOR2_BOTTOM );
SetSlotOption( FLOOR2_BL1:VALUE, IMAGE, F2blength1 );

    /************************************
     **** INSTANCE: FLOOR2_BL2
     ************************************/
MakeInstance( FLOOR2_BL2, FLOOR2_BOTTOM );
SetSlotOption( FLOOR2_BL2:VALUE, IMAGE, F2blength2 );

    /************************************
     **** INSTANCE: FLOOR2_BL3
     ************************************/
MakeInstance( FLOOR2_BL3, FLOOR2_BOTTOM );
SetSlotOption( FLOOR2_BL3:VALUE, IMAGE, F2blength3 );

    /************************************
     **** INSTANCE: FLOOR2_BL4
     ************************************/
MakeInstance( FLOOR2_BL4, FLOOR2_BOTTOM );
SetSlotOption( FLOOR2_BL4:VALUE, IMAGE, F2blength4 );

    /************************************
     **** INSTANCE: FLOOR2_LW1
     ************************************/
MakeInstance( FLOOR2_LW1, FLOOR2_LEFT );
SetSlotOption( FLOOR2_LW1:VALUE, IMAGE, F2lwidth1 );

    /************************************
     **** INSTANCE: FLOOR2_LW2
     ************************************/
MakeInstance( FLOOR2_LW2, FLOOR2_LEFT );
SetSlotOption( FLOOR2_LW2:VALUE, IMAGE, F2lwidth2 );

    /************************************
     **** INSTANCE: FLOOR2_LW3
     ************************************/
MakeInstance( FLOOR2_LW3, FLOOR2_LEFT );
SetSlotOption( FLOOR2_LW3:VALUE, IMAGE, F2lwidth3 );

    /************************************
     **** INSTANCE: FLOOR2_LW4
     ************************************/
```

```
MakeInstance( FLOOR2_LW4, FLOOR2_LEFT );
SetSlotOption( FLOOR2_LW4:VALUE, IMAGE, F2lwidth4 );

    /************************************
     **** INSTANCE: FLOOR2_RW1
     ************************************/
MakeInstance( FLOOR2_RW1, FLOOR2_RIGHT );
SetSlotOption( FLOOR2_RW1:VALUE, IMAGE, F2rwidth1 );

    /************************************
     **** INSTANCE: FLOOR2_RW2
     ************************************/
MakeInstance( FLOOR2_RW2, FLOOR2_RIGHT );
SetSlotOption( FLOOR2_RW2:VALUE, IMAGE, F2rwidth2 );

    /************************************
     **** INSTANCE: FLOOR2_RW3
     ************************************/
MakeInstance( FLOOR2_RW3, FLOOR2_RIGHT );
SetSlotOption( FLOOR2_RW3:VALUE, IMAGE, F2rwidth3 );

    /************************************
     **** INSTANCE: FLOOR2_RW4
     ************************************/
MakeInstance( FLOOR2_RW4, FLOOR2_RIGHT );
SetSlotOption( FLOOR2_RW4:VALUE, IMAGE, F2rwidth4 );

    /************************************
     **** INSTANCE: FLOOR2_INT_LDWLS
     ************************************/
MakeInstance( FLOOR2_INT_LDWLS, FLOOR2_MNINT_WALLS );
SetSlotOption( FLOOR2_INT_LDWLS:VALUE, IMAGE, F2ldwls );

    /************************************
     **** INSTANCE: FLOOR2_INT_NLDWLS
     ************************************/
MakeInstance( FLOOR2_INT_NLDWLS, FLOOR2_MNINT_WALLS );
SetSlotOption( FLOOR2_INT_NLDWLS:VALUE, IMAGE, F2nldwls );

    /************************************
     **** INSTANCE: GARAGE
     ************************************/
MakeInstance( GARAGE, Root );
```

```
MakeSlot( GARAGE:LENGTH );
SetSlotOption( GARAGE:LENGTH, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:LENGTH, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:LENGTH, MAXIMUM_VALUE, 100 );
GARAGE:LENGTH = NULL;
SetSlotOption( GARAGE:LENGTH, IMAGE, Gar_length );
MakeSlot( GARAGE:BREADTH );
SetSlotOption( GARAGE:BREADTH, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:BREADTH, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:BREADTH, MAXIMUM_VALUE, 100 );
GARAGE:BREADTH = NULL;
SetSlotOption( GARAGE:BREADTH, IMAGE, Gar_breadth );
MakeSlot( GARAGE:HEIGHT );
SetSlotOption( GARAGE:HEIGHT, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:HEIGHT, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:HEIGHT, MAXIMUM_VALUE, 15 );
GARAGE:HEIGHT = NULL;
SetSlotOption( GARAGE:HEIGHT, IMAGE, Gar_height );
MakeSlot( GARAGE:DEFAULT );
SetSlotOption( GARAGE:DEFAULT, ALLOWABLE_VALUES, YES, NO );
GARAGE:DEFAULT = NO;
SetSlotOption( GARAGE:DEFAULT, IMAGE, Gar_default );
MakeSlot( GARAGE:DOOR_TRIM );
SetSlotOption( GARAGE:DOOR_TRIM, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:DOOR_TRIM, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:DOOR_TRIM, MAXIMUM_VALUE, 10000 );
MakeSlot( GARAGE:FLOAT_HEADER );
SetSlotOption( GARAGE:FLOAT_HEADER, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:FLOAT_HEADER, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:FLOAT_HEADER, MAXIMUM_VALUE, 10000 );
MakeSlot( GARAGE:BEAM );
SetSlotOption( GARAGE:BEAM, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:BEAM, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:BEAM, MAXIMUM_VALUE, 10000 );
MakeSlot( GARAGE:OUTER_WALL );
SetSlotOption( GARAGE:OUTER_WALL, VALUE_TYPE, NUMBER );
SetSlotOption( GARAGE:OUTER_WALL, MINIMUM_VALUE, 0 );
SetSlotOption( GARAGE:OUTER_WALL, MAXIMUM_VALUE, 9999 );

    /***********************************
     **** INSTANCE: FR_RF_PITCH
     ***********************************/
MakeInstance( FR_RF_PITCH, PITCH );
```

```
FR_RF_PITCH:PITCH_HT = NULL;
SetSlotOption( FR_RF_PITCH:PITCH_HT, IMAGE, Fr_rf_pitch_ht );

    /************************************
     **** INSTANCE: BK_RF_PITCH
     ************************************/
MakeInstance( BK_RF_PITCH, PITCH );
BK_RF_PITCH:PITCH_HT = NULL;
SetSlotOption( BK_RF_PITCH:PITCH_HT, IMAGE, Bk_rf_pitch_ht );

    /************************************
     **** INSTANCE: TRUSS
     ************************************/
MakeInstance( TRUSS, ROOF );
MakeSlot( TRUSS:OC );
SetSlotOption( TRUSS:OC, ALLOWABLE_VALUES, 16, 24 );
TRUSS:OC = 16;
SetSlotOption( TRUSS:OC, IMAGE, Truss_oc );
MakeSlot( TRUSS:TRUSS_BRACE );
SetSlotOption( TRUSS:TRUSS_BRACE, MULTIPLE );
SetSlotOption(          TRUSS:TRUSS_BRACE,          ALLOWABLE_VALUES,
2x4_CONT_LATRL_BRACING,                              METAL_TR_SPACERS,
DBL_DIPPD_GAL_STL_TR_PLATES );
ClearList( TRUSS:TRUSS_BRACE );
SetSlotOption( TRUSS:TRUSS_BRACE, IMAGE, Truss_brace );
MakeSlot( TRUSS:FASTNERS );
SetSlotOption(          TRUSS:FASTNERS,          ALLOWABLE_VALUES,
HURRICANE_FASTNERS, TRUSS_HANGERS );
SetSlotOption( TRUSS:FASTNERS, IMAGE, Fastners_type );

    /************************************
     **** INSTANCE: GABLES
     ************************************/
MakeInstance( GABLES, ROOF );
MakeSlot( GABLES:TYPE );
SetSlotOption( GABLES:TYPE, ALLOWABLE_VALUES, FLUSH, OFF_SET );
GABLES:TYPE = FLUSH;
SetSlotOption( GABLES:TYPE, IMAGE, Gables_type );
MakeSlot( GABLES:PROJECTION );
SetSlotOption( GABLES:PROJECTION, ALLOWABLE_VALUES, 12, 16, 24 );
GABLES:PROJECTION = 12;
SetSlotOption( GABLES:PROJECTION, IMAGE, Gable_projection );
MakeSlot( GABLES:SHEATHING );
```

```
SetSlotOption( GABLES:SHEATHING, ALLOWABLE_VALUES,
0.5_FAC_APPLIED, NONE );
GABLES:SHEATHING = 0.5_FAC_APPLIED;
SetSlotOption( GABLES:SHEATHING, IMAGE, Gable_sheathing );
MakeSlot( GABLES:MISC );
SetSlotOption( GABLES:MISC, VALUE_TYPE, NUMBER );
SetSlotOption( GABLES:MISC, MINIMUM_VALUE, 0 );
SetSlotOption( GABLES:MISC, MAXIMUM_VALUE, 9999 );

    /************************************
     **** INSTANCE: FLOOR1
     ************************************/
MakeInstance( FLOOR1, FLOOR );
MakeSlot( FLOOR1:SILL_PLATES );
SetSlotOption( FLOOR1:SILL_PLATES, ALLOWABLE_VALUES, 2_BY_4, 2_BY_6,
2_BY_8 );
SetSlotOption( FLOOR1:SILL_PLATES, IMAGE, Sill_plates );
MakeSlot( FLOOR1:SILL_PRESS );
SetSlotOption( FLOOR1:SILL_PRESS, ALLOWABLE_VALUES, YES, NO );
SetSlotOption( FLOOR1:SILL_PRESS, IMAGE, Sill_press );
SetSlotOption( FLOOR1:FLOOR_SHEATH_SIZE, IMAGE, Fl1_sh_size );
SetSlotOption( FLOOR1:FLOOR_SHEATH_TYPE, IMAGE, Fl1_sh_type );
SetSlotOption( FLOOR1:TRUSS_SPACING, IMAGE, Fl1_tr_space );
SetSlotOption( FLOOR1:TRUSS_HANGERS, IMAGE, Fl1_tr_hangers );
SetSlotOption( FLOOR1:FRAME_TYPE, IMAGE, Fl1_frame );
SetSlotOption( FLOOR1:TRUSS_OPTIONS, IMAGE, Floor1_tr_options );
MakeSlot( FLOOR1:GIRDER_COST );
SetSlotOption( FLOOR1:GIRDER_COST, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR1:GIRDER_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR1:GIRDER_COST, MAXIMUM_VALUE, 99999 );
MakeSlot( FLOOR1:SLPT_COST );
SetSlotOption( FLOOR1:SLPT_COST, VALUE_TYPE, NUMBER );
SetSlotOption( FLOOR1:SLPT_COST, MINIMUM_VALUE, 0 );
SetSlotOption( FLOOR1:SLPT_COST, MAXIMUM_VALUE, 99999 );

    /************************************
     **** INSTANCE: FLOOR2
     ************************************/
MakeInstance( FLOOR2, FLOOR );
SetSlotOption( FLOOR2:FLOOR_SHEATH_SIZE, IMAGE, Fl2_sh_size );
SetSlotOption( FLOOR2:FLOOR_SHEATH_TYPE, IMAGE, Fl2_sh_type );
SetSlotOption( FLOOR2:TRUSS_SPACING, IMAGE, Fl2_tr_space );
SetSlotOption( FLOOR2:TRUSS_HANGERS, IMAGE, Fl2_tr_hangers );
```

```
SetSlotOption( FLOOR2:FRAME_TYPE, IMAGE, Fl2_frame );
SetSlotOption( FLOOR2:TRUSS_OPTIONS, IMAGE, Floor2_tr_options );

    /************************************
     **** INSTANCE: EXT_FL1
     ************************************/
MakeInstance( EXT_FL1, EXT_WALLS );
SetSlotOption( EXT_FL1:HEIGHT, IMAGE, Ext_fl1_ht );
SetSlotOption( EXT_FL1:BOTTOM_PLATE, IMAGE, Ext_fl1_btmplt );

    /************************************
     **** INSTANCE: EXT_FL2
     ************************************/
MakeInstance( EXT_FL2, EXT_WALLS );
SetSlotOption( EXT_FL2:HEIGHT, IMAGE, Ext_fl2_ht );

    /************************************
     **** INSTANCE: INT_FL1
     ************************************/
MakeInstance( INT_FL1, INT_WALLS );
SetSlotOption( INT_FL1:HEIGHT, IMAGE, Int_fl1_ht );
SetSlotOption( INT_FL1:BOTTOM_PLATE, IMAGE, Int_fl1_btmplt );

    /************************************
     **** INSTANCE: INT_FL2
     ************************************/
MakeInstance( INT_FL2, INT_WALLS );
SetSlotOption( INT_FL2:HEIGHT, IMAGE, Int_fl2_ht );

    /************************************
     **** INSTANCE: TEN_SEP_WALLS
     ************************************/
MakeInstance( TEN_SEP_WALLS, WALLS );
SetSlotOption( TEN_SEP_WALLS:HEIGHT, IMAGE, Tensep_wls_ht );
SetSlotOption( TEN_SEP_WALLS:STUD_SPACING, IMAGE, Tensep_std_spacing );
SetSlotOption( TEN_SEP_WALLS:BOTTOM_PLATE, IMAGE, Tensep_btmplt );
MakeSlot( TEN_SEP_WALLS:MISC );
SetSlotOption( TEN_SEP_WALLS:MISC, MULTIPLE );
SetSlotOption( TEN_SEP_WALLS:MISC, ALLOWABLE_VALUES,
NO_TEN_SEP_WALLS, SINGLE_2x4_WALL, DOUBLE_2x4_WALL,
STAGGERED_STUD_WALL, ATTIC_SEP_WLS_SAME,
ATTIC_SEP_WLS_TRUSSES );
ClearList( TEN_SEP_WALLS:MISC );
```

```
SetSlotOption( TEN_SEP_WALLS:MISC, IMAGE, Tensepwls_misc );
MakeSlot( TEN_SEP_WALLS:COVERAGE );
SetSlotOption( TEN_SEP_WALLS:COVERAGE, ALLOWABLE_VALUES, NONE,
ONE_LAYER, TWO_LAYERS, FOUR_LAYERS );
TEN_SEP_WALLS:COVERAGE = NONE;
MakeSlot( TEN_SEP_WALLS:LENGTH );
SetSlotOption( TEN_SEP_WALLS:LENGTH, VALUE_TYPE, NUMBER );
SetSlotOption( TEN_SEP_WALLS:LENGTH, MINIMUM_VALUE, 0 );
SetSlotOption( TEN_SEP_WALLS:LENGTH, MAXIMUM_VALUE, 100 );
TEN_SEP_WALLS:LENGTH = 0;

    /************************************
     **** INSTANCE: EXT_TRIM
     ************************************/
MakeInstance( EXT_TRIM, Root );
MakeSlot( EXT_TRIM:PLYWD_SOFFITT );
SetSlotOption( EXT_TRIM:PLYWD_SOFFITT, IMAGE, Ply_soff );
MakeSlot( EXT_TRIM:PLYWD_BOXING );
SetSlotOption( EXT_TRIM:PLYWD_BOXING, IMAGE, Ply_box );
MakeSlot( EXT_TRIM:SPRUCE_PRIMED );
SetSlotOption( EXT_TRIM:SPRUCE_PRIMED, IMAGE, Spruce_prim );
MakeSlot( EXT_TRIM:GAB_RET_BOX );
SetSlotOption( EXT_TRIM:GAB_RET_BOX, IMAGE, Gab_ret_box );
MakeSlot( EXT_TRIM:GAB_RET_BOARD );
SetSlotOption( EXT_TRIM:GAB_RET_BOARD, IMAGE, Gab_ret_brd );
MakeSlot( EXT_TRIM:PORCH_RAIL );
SetSlotOption( EXT_TRIM:PORCH_RAIL, IMAGE, Por_rails );
MakeSlot( EXT_TRIM:PORCH_BALUSTER );
SetSlotOption( EXT_TRIM:PORCH_BALUSTER, IMAGE, Por_bal );
MakeSlot( EXT_TRIM:WIN_HD_PANEL );
SetSlotOption( EXT_TRIM:WIN_HD_PANEL, IMAGE, Win_hd_panels_e );
MakeSlot( EXT_TRIM:WIN_SHUTTERS );
SetSlotOption( EXT_TRIM:WIN_SHUTTERS, IMAGE, Win_shutters_e );
MakeSlot( EXT_TRIM:PORCH_COLS );
SetSlotOption( EXT_TRIM:PORCH_COLS, IMAGE, Porch_cols_e );
MakeSlot( EXT_TRIM:CUPOLA_WH_VANE );
SetSlotOption( EXT_TRIM:CUPOLA_WH_VANE, IMAGE, Cup_vanes );

    /************************************
     **** INSTANCE: DOORS&ETC
     ************************************/
MakeInstance( DOORS&ETC, Root );
MakeSlot( DOORS&ETC:EXTDRS_SINGLE );
```

```
SetSlotOption( DOORS&ETC:EXTDRS_SINGLE, IMAGE, Door1_num_e );
MakeSlot( DOORS&ETC:EXTDRS_DOUBLE );
SetSlotOption( DOORS&ETC:EXTDRS_DOUBLE, IMAGE, Door2_num_e );
MakeSlot( DOORS&ETC:GARDR_LENGTH );
SetSlotOption( DOORS&ETC:GARDR_LENGTH, IMAGE, Gardoor_length_e );
MakeSlot( DOORS&ETC:INTDR_LENGTHS );
SetSlotOption( DOORS&ETC:INTDR_LENGTHS, IMAGE, Intdoor_length_e );
MakeSlot( DOORS&ETC:WIN_AREA );
SetSlotOption( DOORS&ETC:WIN_AREA, IMAGE, Win_area_e );
MakeSlot( DOORS&ETC:SINK_NUM );
SetSlotOption( DOORS&ETC:SINK_NUM, IMAGE, Sink_num_e );
MakeSlot( DOORS&ETC:SINKFAU_NUM );
SetSlotOption( DOORS&ETC:SINKFAU_NUM, IMAGE, Sinkfau_num_e );
MakeSlot( DOORS&ETC:WALLCAB_STD_NUM );
SetSlotOption( DOORS&ETC:WALLCAB_STD_NUM, IMAGE, Wlcab_std_e );
MakeSlot( DOORS&ETC:WALLCAB_UP2_NUM );
SetSlotOption( DOORS&ETC:WALLCAB_UP2_NUM, IMAGE, Wlcab_up2_e );
MakeSlot( DOORS&ETC:VANCAB_STD_NUM );
SetSlotOption( DOORS&ETC:VANCAB_STD_NUM, IMAGE, Vancab_std_e );
MakeSlot( DOORS&ETC:VANCAB_UP2_NUM );
SetSlotOption( DOORS&ETC:VANCAB_UP2_NUM, IMAGE, Vancab_up2_e );
MakeSlot( DOORS&ETC:COUNTTOP_LENGTH );
SetSlotOption( DOORS&ETC:COUNTTOP_LENGTH, IMAGE, Count_top_e );
MakeSlot( DOORS&ETC:CABFIN_LENGTH );
SetSlotOption( DOORS&ETC:CABFIN_LENGTH, IMAGE, Cab_fin_e );

    /************************************
     **** INSTANCE: INS&ETC
     ************************************/
MakeInstance( INS&ETC, Root );
MakeSlot( INS&ETC:DECKING_AREA );
SetSlotOption( INS&ETC:DECKING_AREA, IMAGE, Decking_area_e );
MakeSlot( INS&ETC:UNDERLAY_AREA );
SetSlotOption( INS&ETC:UNDERLAY_AREA, IMAGE, Underlayment_area_e );
MakeSlot( INS&ETC:WIN_TRIM );
SetSlotOption( INS&ETC:WIN_TRIM, IMAGE, Win_trim_e );
MakeSlot( INS&ETC:DOOR_TRIM );
SetSlotOption( INS&ETC:DOOR_TRIM, IMAGE, Door_trim_e );
MakeSlot( INS&ETC:SHELVING_LENGTH );
SetSlotOption( INS&ETC:SHELVING_LENGTH, IMAGE, Shelving_length_e );
MakeSlot( INS&ETC:STAIRWAY_LENGTH );
SetSlotOption( INS&ETC:STAIRWAY_LENGTH, IMAGE, Stairway_length_e );
MakeSlot( INS&ETC:FLOOR1_INS );
```

```
SetSlotOption( INS&ETC:FLOOR1_INS, IMAGE, Floor1_ins );
MakeSlot( INS&ETC:CEILING1_INS );
SetSlotOption( INS&ETC:CEILING1_INS, IMAGE, Ceiling1_ins );
MakeSlot( INS&ETC:WALLS1_INS );
SetSlotOption( INS&ETC:WALLS1_INS, IMAGE, Walls1_ins );
MakeSlot( INS&ETC:WALLS2_INS );
SetSlotOption( INS&ETC:WALLS2_INS, IMAGE, Walls2_ins );
MakeSlot( INS&ETC:FLOOR2_INS );
SetSlotOption( INS&ETC:FLOOR2_INS, IMAGE, Floor2_ins );
MakeSlot( INS&ETC:CEILING2_INS );
SetSlotOption( INS&ETC:CEILING2_INS, IMAGE, Ceiling2_ins );
MakeSlot( INS&ETC:INSSUPP1 );
SetSlotOption( INS&ETC:INSSUPP1, IMAGE, Inssupp1 );
MakeSlot( INS&ETC:INSSUPP2 );
SetSlotOption( INS&ETC:INSSUPP2, IMAGE, Inssupp2 );
MakeSlot( INS&ETC:FLOOR1_BSBRD );
SetSlotOption( INS&ETC:FLOOR1_BSBRD, IMAGE, Floor1_bsbrd );
MakeSlot( INS&ETC:FLOOR2_BSBRD );
SetSlotOption( INS&ETC:FLOOR2_BSBRD, IMAGE, Floor2_bsbrd );
```

B.3: THE RULES IN THE SYSTEM

```
    /************************************
     **** RULE: R_FL1TOTAL_DIMS
     ************************************/
MakeRule( R_FL1TOTAL_DIMS, [],
  {
  ( KnownValue?( TL1, VALUE ) Or KnownValue?( BL1, VALUE ) )
    And ( KnownValue?( LW1, VALUE ) Or KnownValue?( RW1, VALUE ) );
  },
  {
  VALUE:TOTAL_LENGTH = ( TL1:VALUE + TL2:VALUE + TL3:VALUE +
                TL4:VALUE );
  VALUE:TOTAL_WIDTH = ( LW1:VALUE + LW2:VALUE + LW3:VALUE +
                LW4:VALUE );
  VALUE:TOTAL_AREA = ( VALUE:TOTAL_LENGTH * VALUE:TOTAL_WIDTH
);
  } );

    /************************************
     **** RULE: R_FL2TOTAL_DIMS
     ************************************/
```

```
MakeRule( R_FL2TOTAL_DIMS, [],
  {
  ( KnownValue?( FLOOR2_TL1, VALUE ) Or KnownValue?( FLOOR2_BL1,
                              VALUE ) )
            And ( KnownValue?( FLOOR2_LW1, VALUE ) Or KnownValue?(
FLOOR2_RW1,
                                    VALUE ) );
  },
  {
          FLOOR2_VALUE:TOTAL_LENGTH = ( FLOOR2_TL1:VALUE +
FLOOR2_TL2:VALUE
                        + FLOOR2_TL3:VALUE + FLOOR2_TL4:VALUE );
           FLOOR2_VALUE:TOTAL_WIDTH = ( FLOOR2_LW1:VALUE +
FLOOR2_LW2:VALUE
                        + FLOOR2_LW3:VALUE + FLOOR2_LW4:VALUE );
     FLOOR2_VALUE:TOTAL_AREA = ( FLOOR2_VALUE:TOTAL_LENGTH *
FLOOR2_VALUE:TOTAL_WIDTH );
  } );

    /************************************
     **** RULE: R_GARAGE
     ************************************/
MakeRule( R_GARAGE, [],
  KnownValue?( GARAGE, LENGTH ) Or KnownValue?( GARAGE, BREADTH ),
  {
  GARAGE:DOOR_TRIM = ( GARAGE:LENGTH * 2 * 0.46 )
    + ( GARAGE:BREADTH * 0.46 );
  GARAGE:FLOAT_HEADER = GARAGE:LENGTH * 0.4;
  GARAGE:BEAM = GARAGE:LENGTH * 5.75;
  GARAGE:OUTER_WALL = ( ( GARAGE:LENGTH * 2 ) + GARAGE:BREADTH )
    * 10.26 * 1.06;
     Global:TEMP += GARAGE:DOOR_TRIM + GARAGE:FLOAT_HEADER +
GARAGE:BEAM
    + GARAGE:OUTER_WALL;
  } );

    /************************************
     **** RULE: R_FL2INTWL_DIMS
     ************************************/
MakeRule( R_FL2INTWL_DIMS, [],
        KnownValue?( FLOOR2_INT_LDWLS, VALUE ) Or KnownValue?(
FLOOR2_INT_NLDWLS,
                              VALUE ),
```

```
                        FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH      =
FLOOR2_INT_LDWLS:VALUE
    + FLOOR2_INT_NLDWLS:VALUE );

    /************************************
     **** RULE: R_FL1INTWL_DIMS
     ************************************/
MakeRule( R_FL1INTWL_DIMS, [],
  KnownValue?( INT_LDWLS, VALUE ) Or KnownValue?( INT_NLDWLS,
                        VALUE ),
          MNINT_WALLS:TOTAL_INT_LENGTH  =  INT_LDWLS:VALUE  +
INT_NLDWLS:VALUE );

    /************************************
     **** RULE: R_ROOF_COST
     ************************************/
MakeRule( R_ROOF_COST, [],
   KnownValue?( ROOF:JOIST_SIZE ) Or KnownValue?( ROOF:SHEATHING_TYPE
),
  {
  If ( GABLES:TYPE #= FLUSH )
    Then ( ROOF:ROOF_TRUSSES = ( VALUE:TOTAL_AREA * 1.1 ) )
          Else  ROOF:ROOF_TRUSSES  =  (  VALUE:TOTAL_AREA  *  (  (  (
GABLES:PROJECTION
                                    /
                                    6 )
                                   *
                                    0.08 )
                                  + 0.76 ) );
  ROOF:RFING_MTLS = ( VALUE:TOTAL_AREA + ( ( ROOF:OVERHANG *
                              ( VALUE:TOTAL_LENGTH
                                  + VALUE:TOTAL_WIDTH )
                             * 2 ) / 12 ) )
    / 3;
  ROOF:FL&SHOSB = VALUE:TOTAL_AREA * 0.5;
  GABLES:MISC = VALUE:TOTAL_AREA * 0.25;
  ROOF:GAB_PROJ = ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH )
    * 2 * 1.65;
       Global:TEMP  +=  ROOF:ROOF_TRUSSES  +  ROOF:RFING_MTLS  +
ROOF:FL&SHOSB
    + ROOF:GAB_PROJ + GABLES:MISC;
  } );
```

```
    /************************************
     **** RULE: R_OCF
     ************************************/
MakeRule( R_OCF, [],
  KnownValue?( FLOOR1, TRUSS_SPACING ) Or KnownValue?( FLOOR2,
                                TRUSS_SPACING ),
  {
  If KnownValue?( FLOOR1, TRUSS_SPACING )
   Then {
      FLOOR1:OCF = 12 / FLOOR1:TRUSS_SPACING;
      };
  If KnownValue?( FLOOR2, TRUSS_SPACING )
   Then {
      FLOOR2:OCF = 12 / FLOOR2:TRUSS_SPACING;
      };
  } );

    /************************************
     **** RULE: R_FL1_JOIST
     ************************************/
MakeRule( R_FL1_JOIST, [],
  FLOOR1:FRAME_TYPE #= JOIST,
  {
  FLOOR1:JST_COST = ( ( ( VALUE:TOTAL_LENGTH * FLOOR1:OCF )
                  + 1 ) * ( VALUE:TOTAL_WIDTH + 2 ) )
    * 1.5 * 1.1;
  FLOOR1:SLPT_COST = ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
                  + 7.5 ) * 2 * 0.56 * 1.18;
  FLOOR1:ADHESIVE_COST = ( VALUE:TOTAL_AREA * 1.61 )
    / 107;
  FLOOR1:T&G_COST = ( VALUE:TOTAL_AREA * 16.219 * 1.16 )
    / 32;
  If ( VALUE:TOTAL_WIDTH < 27 )
   Then ( FLOOR1:GIRDER_COST = ( VALUE:TOTAL_LENGTH + 2 )
          * 4.5 )
   Else FLOOR1:GIRDER_COST = VALUE:TOTAL_LENGTH * 4.5 * 1.5;
  If ( VALUE:TOTAL_AREA < 1000 )
   Then ( FLOOR1:JST_HGR_COST = 5 )
   Else FLOOR1:JST_HGR_COST = 10;
       Global:TEMP += FLOOR1:JST_HGR_COST + FLOOR1:JST_COST +
FLOOR1:SLPT_COST
                  + FLOOR1:ADHESIVE_COST + FLOOR1:T&G_COST +
FLOOR1:GIRDER_COST;
```

```
  } );

    /************************************
     **** RULE: R_FL1_TRUSS
     ************************************/
MakeRule( R_FL1_TRUSS, [],
  FLOOR1:FRAME_TYPE #= TRUSS,
  {
  FLOOR1:JST_COST = VALUE:TOTAL_AREA * FLOOR1:OCF * 0.83 * 1.3;
  FLOOR1:SLPT_COST = ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
                + 7.5 ) * 2 * 0.56 * 1.08;
  FLOOR1:ADHESIVE_COST = ( VALUE:TOTAL_AREA * 1.61 )
    / 107;
  FLOOR1:T&G_COST = ( VALUE:TOTAL_AREA * 16.219 * 1.16 )
    / 32;
  If ( VALUE:TOTAL_WIDTH < 27 )
   Then ( FLOOR1:GIRDER_COST = ( VALUE:TOTAL_LENGTH + 2 )
          * 4.5 )
   Else FLOOR1:GIRDER_COST = VALUE:TOTAL_LENGTH * 4.5 * 1.5;
        Global:TEMP  +=  FLOOR1:JST_COST  +  FLOOR1:SLPT_COST  +
FLOOR1:ADHESIVE_COST
    + FLOOR1:T&G_COST + FLOOR1:GIRDER_COST;
  } );

    /************************************
     **** RULE: R_FL2_JOIST
     ************************************/
MakeRule( R_FL2_JOIST, [],
  FLOOR2:FRAME_TYPE #= JOIST,
  {
  FLOOR2:JST_COST = ( ( ( FLOOR2_VALUE:TOTAL_LENGTH * FLOOR2:OCF )
                + 1 ) * ( FLOOR2_VALUE:TOTAL_WIDTH
                        + 2 ) )
    * 1.5 * 1.1;
  FLOOR2:ADHESIVE_COST = ( FLOOR2_VALUE:TOTAL_AREA * 1.61 )
    / 107;
  FLOOR2:T&G_COST = ( FLOOR2_VALUE:TOTAL_AREA * 16.219 * 1.16 )
    / 32;
  If ( FLOOR2_VALUE:TOTAL_AREA < 1000 )
   Then ( FLOOR2:JST_HGR_COST = 5 )
   Else FLOOR2:JST_HGR_COST = 10;
       Global:TEMP  +=  FLOOR2:JST_HGR_COST  +  FLOOR2:JST_COST  +
FLOOR2:ADHESIVE_COST
```

```
    + FLOOR2:T&G_COST;
  } );

    /************************************
     **** RULE: R_FL2_TRUSS
     ************************************/
MakeRule( R_FL2_TRUSS, [],
  FLOOR2:FRAME_TYPE #= TRUSS,
  {
  FLOOR2:JST_COST = FLOOR2_VALUE:TOTAL_AREA * FLOOR2:OCF * 0.83
    * 1.3;
  FLOOR2:ADHESIVE_COST = ( FLOOR2_VALUE:TOTAL_AREA * 1.61 )
    / 107;
  FLOOR2:T&G_COST = ( FLOOR2_VALUE:TOTAL_AREA * 16.219 * 1.16 )
    / 32;
      Global:TEMP += FLOOR2:JST_COST + FLOOR2:ADHESIVE_COST +
FLOOR2:T&G_COST;
  } );

    /************************************
     **** RULE: R_FL1_WALLS
     ************************************/
MakeRule( R_FL1_WALLS, [],
  KnownValue?( TEN_SEP_WALLS, HEIGHT ) Or KnownValue?( EXT_FL1,
                                HEIGHT )
    Or KnownValue?( INT_FL1, HEIGHT ),
  {
            EXT_FL1:OUTWL_COST = ( VALUE:TOTAL_LENGTH +
VALUE:TOTAL_WIDTH )
    * 2 * 7.3;
        EXT_FL1:PTBTPLT_COST = ( ( ( VALUE:TOTAL_LENGTH +
VALUE:TOTAL_WIDTH )
                    * 2 ) + MNINT_WALLS:TOTAL_INT_LENGTH )
    * 0.1;
  INT_FL1:INTWL_COST = MNINT_WALLS:TOTAL_INT_LENGTH * 5.4;
           EXT_FL1:BRACING_COST = ( VALUE:TOTAL_LENGTH +
VALUE:TOTAL_WIDTH )
    * 2 * 0.16;
  Global:TEMP += ( EXT_FL1:OUTWL_COST + EXT_FL1:PTBTPLT_COST
              + INT_FL1:INTWL_COST + EXT_FL1:BRACING_COST )
    * 1.06;
  } );
```

```
    /************************************
     **** RULE: R_FL2_WALLS
     ************************************/
MakeRule( R_FL2_WALLS, [],
  KnownValue?( TEN_SEP_WALLS, HEIGHT ) Or KnownValue?( EXT_FL2,
                                HEIGHT )
    Or KnownValue?( INT_FL2, HEIGHT ),
  {
        EXT_FL2:OUTWL_COST = ( FLOOR2_VALUE:TOTAL_LENGTH +
FLOOR2_VALUE:TOTAL_WIDTH )
    * 2 * 7.3;
     EXT_FL2:PTBTPLT_COST = ( ( ( FLOOR2_VALUE:TOTAL_LENGTH +
FLOOR2_VALUE:TOTAL_WIDTH )
                        * 2 ) + FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH )
    * 0.1;
  INT_FL2:INTWL_COST = FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH *
    5.4;
       EXT_FL2:BRACING_COST = ( FLOOR2_VALUE:TOTAL_LENGTH +
FLOOR2_VALUE:TOTAL_WIDTH )
    * 2 * 0.16;
  Global:TEMP += ( EXT_FL2:OUTWL_COST + EXT_FL2:PTBTPLT_COST
              + INT_FL2:INTWL_COST + EXT_FL2:BRACING_COST )
    * 1.06;
  } );

    /************************************
     **** RULE: R_EXT_TRIM
     ************************************/
MakeRule( R_EXT_TRIM, [],
  KnownValue?( EXT_TRIM, CUPOLA_WH_VANE ) Or KnownValue?( EXT_TRIM,
                                GAB_RET_BOARD )
    Or KnownValue?( EXT_TRIM, GAB_RET_BOX ) Or KnownValue?( EXT_TRIM,
                                PLYWD_BOXING )
          Or KnownValue?( EXT_TRIM, PLYWD_SOFFITT ) Or KnownValue?(
EXT_TRIM,
                                PORCH_BALUSTER )
    Or KnownValue?( EXT_TRIM, PORCH_COLS ) Or KnownValue?( EXT_TRIM,
                                PORCH_RAIL )
          Or KnownValue?( EXT_TRIM, SPRUCE_PRIMED ) Or KnownValue?(
EXT_TRIM,
                                WIN_HD_PANEL )
    Or KnownValue?( EXT_TRIM, WIN_SHUTTERS ),
  {
```

```
  If ( StringLength( EXT_TRIM:CUPOLA_WH_VANE ) #= 4 )
   Then Global:TEMP += 220;
  If ( StringLength( EXT_TRIM:GAB_RET_BOARD ) #= 4 )
   Then Global:TEMP += ( VALUE:TOTAL_LENGTH / 1.4 );
  If ( StringLength( EXT_TRIM:GAB_RET_BOX ) #= 4 )
   Then Global:TEMP += ( VALUE:TOTAL_LENGTH * 7.745 * 0.14 );
  If ( StringLength( EXT_TRIM:PLYWD_BOXING ) #= 4 )
   Then Global:TEMP += ( ( VALUE:TOTAL_WIDTH + 4 )
                    * 1.5 * 0.55 );
  If ( StringLength( EXT_TRIM:PLYWD_SOFFITT ) #= 4 )
   Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
                     + 6 ) * 2 * 0.55 );
  If ( StringLength( EXT_TRIM:PORCH_BALUSTER ) #= 4 )
              Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + (
VALUE:TOTAL_WIDTH
                                   * 2 ) )
                    * 1.5 * 2.75 );
  If KnownValue?( EXT_TRIM:PORCH_COLS )
   Then Global:TEMP += ( EXT_TRIM:PORCH_COLS * 34 );
  If KnownValue?( EXT_TRIM:WIN_HD_PANEL )
   Then Global:TEMP += ( EXT_TRIM:WIN_HD_PANEL * 14 );
  If KnownValue?( EXT_TRIM:WIN_SHUTTERS )
   Then Global:TEMP += ( EXT_TRIM:WIN_SHUTTERS * 11 );
  If ( StringLength( EXT_TRIM:PORCH_RAIL ) #= 4 )
    Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
)
                    * 2 * 1.1 );
  If ( StringLength( EXT_TRIM:SPRUCE_PRIMED ) #= 4 )
    Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
)
                    * 2 * 0.5 );
  } );

   /************************************
    **** RULE: R_DOORS&ETC
    ************************************/
MakeRule( R_DOORS&ETC, [],
        KnownValue?( DOORS&ETC, CABFIN_LENGTH ) Or KnownValue?(
DOORS&ETC,
                                COUNTTOP_LENGTH )
        Or KnownValue?( DOORS&ETC, EXTDRS_DOUBLE ) Or KnownValue?(
DOORS&ETC,
                                EXTDRS_SINGLE )
```

```
        Or KnownValue?( DOORS&ETC, GARDR_LENGTH ) Or KnownValue?(
DOORS&ETC,
                                INTDR_LENGTHS )
            Or KnownValue?( DOORS&ETC, SINK_NUM ) Or KnownValue?(
DOORS&ETC,
                                SINKFAU_NUM )
    Or KnownValue?( DOORS&ETC, VANCAB_STD_NUM )
    Or KnownValue?( DOORS&ETC, VANCAB_UP2_NUM )
    Or KnownValue?( DOORS&ETC, WALLCAB_STD_NUM )
    Or KnownValue?( DOORS&ETC, WALLCAB_UP2_NUM )
    Or KnownValue?( DOORS&ETC, WIN_AREA ),
  {
  If KnownValue?( DOORS&ETC:WIN_AREA )
   Then Global:TEMP += ( DOORS&ETC:WIN_AREA * 8.66 );
  If KnownValue?( DOORS&ETC:WALLCAB_UP2_NUM )
   Then Global:TEMP += ( DOORS&ETC:WALLCAB_UP2_NUM * 86 );
  If KnownValue?( DOORS&ETC:WALLCAB_STD_NUM )
   Then Global:TEMP += ( DOORS&ETC:WALLCAB_STD_NUM * 65 );
  If KnownValue?( DOORS&ETC:VANCAB_STD_NUM )
   Then Global:TEMP += ( DOORS&ETC:VANCAB_STD_NUM * 12 * 2.92 );
  If KnownValue?( DOORS&ETC:VANCAB_UP2_NUM )
   Then Global:TEMP += ( DOORS&ETC:VANCAB_UP2_NUM * 12 * 3.9 );
  If KnownValue?( DOORS&ETC:SINKFAU_NUM )
   Then Global:TEMP += ( DOORS&ETC:SINKFAU_NUM * 47 );
  If KnownValue?( DOORS&ETC:SINK_NUM )
   Then Global:TEMP += ( DOORS&ETC:SINK_NUM * 31 );
  If KnownValue?( DOORS&ETC:CABFIN_LENGTH )
   Then Global:TEMP += ( DOORS&ETC:CABFIN_LENGTH * 6.75 );
  If KnownValue?( DOORS&ETC:COUNTTOP_LENGTH )
   Then Global:TEMP += ( DOORS&ETC:COUNTTOP_LENGTH * 15.25 );
  If KnownValue?( DOORS&ETC:EXTDRS_DOUBLE )
   Then Global:TEMP += ( DOORS&ETC:EXTDRS_DOUBLE * 280 );
  If KnownValue?( DOORS&ETC:EXTDRS_SINGLE )
   Then Global:TEMP += ( DOORS&ETC:EXTDRS_SINGLE * 140 );
  If KnownValue?( DOORS&ETC:GARDR_LENGTH )
   Then Global:TEMP += ( DOORS&ETC:GARDR_LENGTH * 30 );
  If KnownValue?( DOORS&ETC:INTDR_LENGTHS )
   Then Global:TEMP += ( DOORS&ETC:INTDR_LENGTHS * 25 );
  } );

   /************************************
    **** RULE: R_INS&ETC
    ************************************/
```

```
MakeRule( R_INS&ETC, [],
  KnownValue?( INS&ETC, CEILING1_INS ) Or KnownValue?( INS&ETC,
                                CEILING2_INS )
    Or KnownValue?( INS&ETC, DECKING_AREA ) Or KnownValue?( INS&ETC,
                                DOOR_TRIM )
    Or KnownValue?( INS&ETC, FLOOR1_BSBRD ) Or KnownValue?( INS&ETC,
                                FLOOR1_INS )
    Or KnownValue?( INS&ETC, FLOOR2_INS ) Or KnownValue?( INS&ETC,
                                FLOOR2_BSBRD )
    Or KnownValue?( INS&ETC, INSSUPP1 ) Or KnownValue?( INS&ETC,
                                INSSUPP2 )
        Or KnownValue?( INS&ETC, SHELVING_LENGTH ) Or KnownValue?(
INS&ETC,
                                STAIRWAY_LENGTH )
        Or KnownValue?( INS&ETC, UNDERLAY_AREA ) Or KnownValue?(
INS&ETC,
                                WALLS1_INS )
    Or KnownValue?( INS&ETC, WALLS2_INS ) Or KnownValue?( INS&ETC,
                                WIN_TRIM ),
  {
  If KnownValue?( INS&ETC:WIN_TRIM )
   Then Global:TEMP += ( INS&ETC:WIN_TRIM * 3.7 );
  If KnownValue?( INS&ETC:DOOR_TRIM )
   Then Global:TEMP += ( INS&ETC:DOOR_TRIM * 4 );
  If KnownValue?( INS&ETC:UNDERLAY_AREA )
   Then Global:TEMP += ( ( INS&ETC:UNDERLAY_AREA * 11 )
                / 32 );
  If KnownValue?( INS&ETC:SHELVING_LENGTH )
   Then Global:TEMP += ( INS&ETC:SHELVING_LENGTH * 1.65 );
  If KnownValue?( INS&ETC:STAIRWAY_LENGTH )
   Then Global:TEMP += ( INS&ETC:STAIRWAY_LENGTH * 30 );
  If KnownValue?( INS&ETC:DECKING_AREA )
   Then Global:TEMP += ( INS&ETC:DECKING_AREA * 2.4 );
  If ( StringLength( INS&ETC:CEILING1_INS ) #= 4 )
   Then Global:TEMP += ( VALUE:TOTAL_AREA * 0.29 );
  If ( StringLength( INS&ETC:CEILING2_INS ) #= 4 )
   Then Global:TEMP += ( FLOOR2_VALUE:TOTAL_AREA * 0.29 );
  If ( StringLength( INS&ETC:FLOOR1_BSBRD ) #= 4 )
   Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
                + MNINT_WALLS:TOTAL_INT_LENGTH )
               * 0.4 );
  If ( StringLength( INS&ETC:FLOOR1_INS ) #= 4 )
   Then Global:TEMP += ( VALUE:TOTAL_AREA * 0.18 );
```

```
  If ( StringLength( INS&ETC:FLOOR2_BSBRD ) #= 4 )
            Then Global:TEMP += ( ( FLOOR2_VALUE:TOTAL_LENGTH +
FLOOR2_VALUE:TOTAL_WIDTH
                      + FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH )
                     * 0.4 );
  If ( StringLength( INS&ETC:FLOOR2_INS ) #= 4 )
   Then Global:TEMP += ( FLOOR2_VALUE:TOTAL_AREA * 0.29 );
  If ( StringLength( INS&ETC:INSSUPP1 ) #= 4 )
   Then Global:TEMP += 6;
  If ( StringLength( INS&ETC:INSSUPP2 ) #= 4 )
   Then Global:TEMP += 6;
  If ( StringLength( INS&ETC:WALLS1_INS ) #= 4 )
   Then Global:TEMP += ( ( VALUE:TOTAL_LENGTH + VALUE:TOTAL_WIDTH
                      + MNINT_WALLS:TOTAL_INT_LENGTH )
                     * 6.8 * 0.115 );
  If ( StringLength( INS&ETC:WALLS2_INS ) #= 4 )
            Then Global:TEMP += ( ( FLOOR2_VALUE:TOTAL_LENGTH +
FLOOR2_VALUE:TOTAL_WIDTH
                      + FLOOR2_MNINT_WALLS:TOTAL_INT_LENGTH )
                     * 6.8 * 0.115 );
  } );
```

VITA

Kishore Thirulokachandran was born on October 25, 1970, in Madras, India. He completed his Bachelor's in Mechanical Engineering from Crescent Engineering College, University of Madras, in May 1991. He completed his Master's in Industrial and Systems Engineering from Virginia Polytechnic Institute and State University in April 1995.