A FRAMEWORK FOR BUILDING ASSEMBLY SELECTION AND GENERATION

by

Khaled Nassar

Dissertation Submitted to the Faculty of

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

in

Environmental Design and Planning

APPROVED

Dr. Yvan Beliveau

(Chairman)

Dr. Walid Thabet

Dr. Quincan Qao

Dr. Michael Ellis

Dr. Jim Jones

June, 1999

Blacksburg, Virginia

Intelligent Building Assemblies

by

Khaled Nassar

Yvan Beliveau, Chairman

Department of Building Construction

(ABSTRACT)

In practice, the building design process can be divided into three major stages; schematic design, design implementation and construction documents development. The majority of the time in the building design delivery process is spent in the latter two stages. Computers can greatly aid the designer in the latter two stages, by providing a tool that helps in choosing the best assemblies for a particular design and, helping in automating the process of construction detail generation. There is lack of such a tool in the architecture design domain.

In this dissertation, a novel approach for the selection and generation of building assemblies is presented. A building product model is described. In this model the building is broken down into assemblies. Each assembly has a graphical representation. By using the assemblies' representations a designer can specify his/her design concept. These assemblies are intelligent. They know how to select the correct assembly constructions for each particular design situation, based on a set of defined criteria and constraints. The different kinds of criteria and constraints that affect the selection of assemblies are identified, and examples are provided. A selection procedure is developed that can perform the selection taking into consideration the various criteria and constraints to produce a best compromise solution.

A computer prototype is developed on top of a traditional computer graphics package (AutoCAD) as a proof of concept. In the prototype, the design knowledge is encapsulated and intelligence is added to the building assemblies of a specific construction type. This intelligence allows the assemblies to be automatically selected and analyzed. Several examples of assemblies are developed in the computer prototype.

The treatment of building components as intelligent objects will significantly increase the efficiency of design in terms of economy and performance. This is because issues related to the specific design can be addressed in an organized way. Issues like cost, constructability, and other performances can be taken into consideration at the design level. The approach described here provides a more efficient and time saving way for selection of building assembly constructions.

“This is of that which my Lord hath taught me”

Interpretation of the meaning of verse 37 Sūrah XII JOSEPH

Acknowledgments

I wish to express my sincere appreciation and gratitude to my advisor Professor Dr. Yvan Beliveau for his continuos support and encouragement as well as for his academic and moral support. His constructive suggestions were vital for this research.

I wish to express my gratitude to all my advisory committee for their time and support during the course of this research. My thanks go to Dr. Walid Thabet, Dr. Mike Ellis, Dr. Quincan Qao and Dr. Jim Jones.

Most obviously I would like to express my deep appreciation to my family to whom I dedicate this thesis, my father professor Dr. Mohammed Nassar, my mother, my brother Walid and my sister Yasmin and to my wife Dina and son Yousef.

I wish to thank all my friends, whose continuos discussions and support greatly helped in this research. My thanks go to Bakari Simba, Ahmed Waly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

In early ages ‘Designing’ a building could not be differentiated from ‘building’ it, since the Master Builder was also the designer. With the increased complexity of design, this arrangement changed. In current practice, the building design process can be divided into three major stages; schematic design, design implementation, and detailing stage. The detailing stage involves generation of complete design documents including detailed drawings of the facility. The various construction and design issues make the amount of information needed to produce even a simple detail, very large. Given the fact that a designer has to produce different types of details, we can see why the task of design detailing requires a lot of experience and knowledge. Even the most experienced designer will rely on many software tools and sources of information like references, other professionals, and previous case histories. The detailing techniques have therefore evolved to try to support this complexity.

At first detailing techniques were mainly concentrated on two-dimensional (2D) hand drawings because of the limitation of the available tools. With the introduction of computers in the building delivery design stages, Computer-Aided Drafting and Design (CADD) techniques minimized the time and mistakes in the detailing phase. However, even with the advent of CADD, a recent survey [DeVries 1996] showed that CADD’s widespread application still focused on ‘automated 2D drawing boards’.

With development of new hardware it was possible to store CADD details. The details were stored as 2D drawings and reused by ‘cut-and-paste’. Even though this procedure saved time it has proved to be unreliable in many cases. Discrepancies often occur between the actual design

and the detail selected from the library. Also conflicts often arise with specifications and other parts of the construction documents.

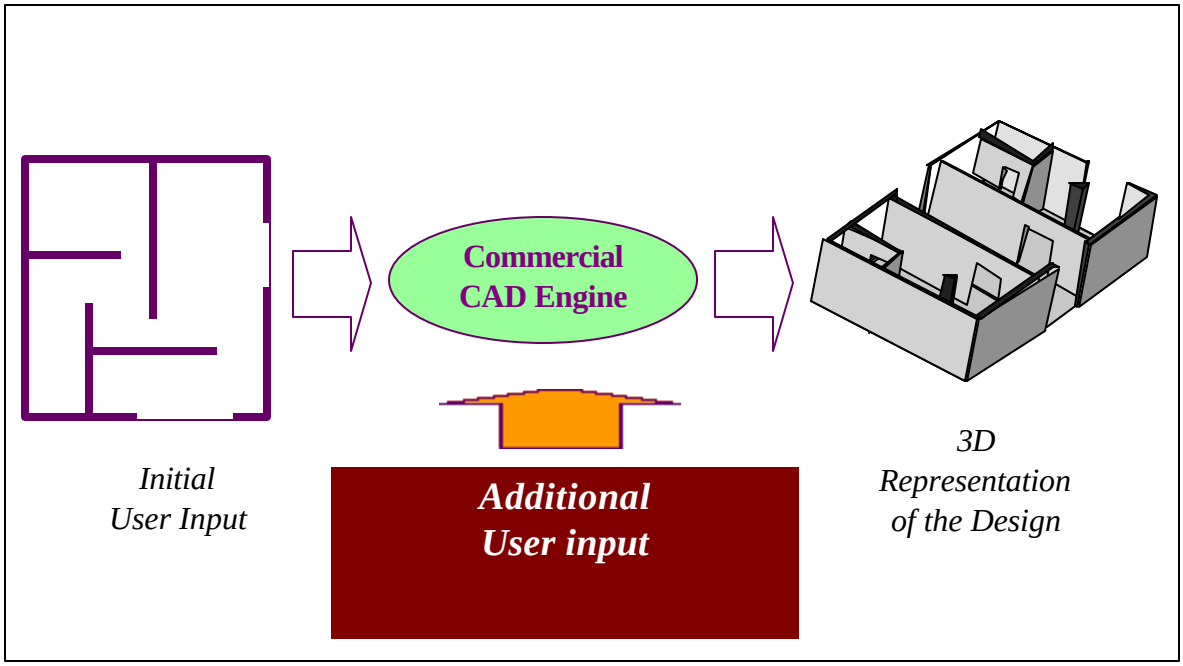


Figure 1.1. Commercial CAD Packages

The change from two dimensional to volumetric design representation (3D) was made possible by the ever-increasing hardware capabilities of today's computers. The three-dimensional (3D) design representation exactly represents the geometry of the building elements, thus eliminating misinterpretations. In the building detail design domain we recently have seen state-of-the-art 3D CADD tools that add drafting intelligence to the detailing process[1]. These tools allow a designer to describe the geometric representations of the building elements (walls, roofs, spaces etc…). The geometric representations of the building elements are usually simple geometric primitives like lines or boxes, as can be shown in Figure 1.1. The designer can use the simple geometric representations and manually choose a specific construction and these tools will draw the building elements in detail.

[1] Software ranging from simple "Home Design" software to sophisticated systems like Bentley's TRIFORMA, POWER ARCHITECT, AutoCAD's ARCHITECT, SOFTPLAN, etc.

The building elements used to generate the details in these tools are usually referred to as building assemblies. The use of assemblies in building design has proved to be useful. However, these state-of-the-art CADD tools described above suffer from several drawbacks:

1. Today's high-end design support software enables better exploitation of the increased computer capabilities. Especially, analyzing the behavior of the building and its elements has become possible through the introduction of many analysis and generation software ranging from shape grammar generators to thermal analysis [Schmit 1990]. However, the integration of different analysis software in the design process in general and in building detail selection in particular is very limited [Augenbroe et al. 1994].

   There are many reasons for not incorporating the different analysis software in building detail production. The main reason is that each software has its separate representation of the building design, i.e. the internal data structures used in each tool are different. The proliferation of the different representation is inevitable because each software performs a different kind of analysis that requires a different building representation, i.e. data structure. For example, daylighting software deal with the building as surfaces (and their surface reflectance) and opening while structural software deal with the building as structural elements (e.g. columns, beams and slabs). It is clear that each software requires a different data structure.

   Furthermore, there are a growing number of applications that perform the same analysis task in different manners or using different analysis methods [Assal 1996]. Such programs are being developed independently of each other and most applications use data structures and representations optimized for their operation. The varied representation causes compatibility problems. The translation of the building data from one software to the other becomes an increasingly difficult problem, as we will see in chapter 4. Several standards like IGES and STEP were introduced to facilitate conversions between different software. However, in the domain of Architecture, Engineering and, Construction (AEC), discrepancies in the product models of many of the current software still exist [DeVries 1996].

2. The construction of a truly intelligent CADD system requires the acquisition and management of a tremendous amount of knowledge and information[2] [Cugini et al. 1988]. The knowledge bases of the state-of-the-art CADD software do not allow for modification or expansion. This limits the knowledge base of these software tools to the hard coded (already programmed) knowledge. For example, adding new assembly construction types is difficult, since there is no standardized method to describe how the assemblies of this construction type will be generated. Furthermore, a user can not add rules about the construction types. Also in the state-of-the-art CADD software tools the user description of the design is confined to a limited number of graphical representations (e.g. a wall can only be represented by a line). This limits the flexibility of design description by the designer (architect).

   The lack of a standard knowledge representation method/interface limits the extension of the knowledge bases of the state-of-the-art CADD tools and even hinders automated knowledge acquisition.

3. Finally and more importantly, a major drawback in the state-of-the-art CADD software is that these software tools lack a methodology for selecting the appropriate construction for building assemblies. Because these tools do not incorporate design and/or construction knowledge, the validity of the generated details is dependant on user input only. The lack of a methodology for selecting the 'best'[3] assemblies for a building design, can often cause design problems. For example, in a state-of-the-art CADD tools it is possible to select an assembly construction with an inappropriate fire rating or select an assembly construction that is not the best assembly for the particular design case (or difficult to use with another assembly, e.g. CMU walls on a wooden frame floor).

---

[2] In this dissertation the information is defined as "the knowledge of ideas, facts and/or processes [Schenck et al. 1994]. Information can be communicated, that is it may be transferred between two or more partners.

[3] The 'best' assembly is the assembly that has the highest performance for a set of criteria defined by the designer.

Due to all the reasons above, the selection and generation of the building assemblies remains a manual process since the assemblies are not linked to any design knowledge. This means that, the state-of-the-art CADD tools are limited to automating drafting tasks.

1.1. SCOPE STATEMENT

The goal of the research in this dissertation is to automate the design development stages, by developing an automated assembly selection and generation procedure. The automatic assembly selection procedure is incorporated in a prototype tool that extends the ability of the current state-of-the-art CADD tools. The prototype tools takes as input a description of the schematic design, and performs an automatic selection of the building assemblies' constructions, based on a set of constraints and criteria. The prototype tool also helps the designer to generate building details. Firstly, a building product model is developed in a formal information modeling language. The product model facilitates information exchange and is implemented as a database (Chapter 5). The database is used to store both the design description (entered graphically by the designer) and the different construction assemblies. Also different criteria and constraints on the use of the construction assemblies can be defined. Examples of the criteria and constraints pertaining to building assemblies are given. Secondly, an assembly selection procedure is described (Chapter 6). The selection procedure handles the selection based on the different criteria. A graphical assembly generation procedure is also defined (Chapter 6) that helps the designer in generating the assembly details. A prototype system is developed for validation of the concept (Chapter 7).

1.2. RESEARCH PROBLEM

A methodology for automating the selection and generation of building assembly is currently in need. The new methodology should consider the limitations of the existing state-of-the-art CADD tools. Using this methodology design and construction knowledge about building assemblies could be described and stored in a computer, so that the building assemblies can be automatically selected and a detailed 3D solid model generated. Details can then be extracted from the generated 3D solid model. The intelligent automation of assembly selection will address several problems [Nassar et al. 1998b].

1. Lack of Knowledge. For example, the designer might have insufficient information about many of the possible construction assemblies available for selection or incomplete information on a particular assembly construction.

2. Lack of Experience. There is no uniform system that will help in the dissemination of previous experiences and knowledge about the various building assemblies. This experience comes mostly in the form of constraints on the use of certain assemblies.

3. Lost time due to repetitive processes in design. For example, certain selection and generation processes are repetitive from one project to another. The automation of these tasks would save a great deal of time and reduce errors.

4. Coordination and rational decision making for better overall performance. Many of the building construction assemblies can have a good performance with respect to one criterion but perform poorly with respect to another (for example exterior insulation finishing systems (EIFS) performs well thermally but perform poorly with respect to moisture migration). With the large number of assemblies in a building and criteria to be considered, this can become a difficult task to perform manually. An automated selection system can help rationalize the selection and present the best compromise solutions.

As can be seen, the lack of a systematic way of describing and storing design and construction knowledge about building assemblies in the current CADD tools hinders the automation of assembly construction selection and generation. Next we briefly introduce intelligent building assemblies as an approach for automating building detail generation.

1.3. INTELLIGENT BUILDING ASSEMBLIES

One can think of an intelligent building assembly as a functional unit of a building with a distinct graphic representation. (e.g. an architectural vocabulary element like a wall, roof, window etc…). The assembly can be represented in a computer at two levels of detail as shown in Figure 2.2.

The assembly can be represented initially by a simple geometric primitive. For example, a box or a plane can represent a wall assembly. The assembly's graphical representation[4] is linked to a central knowledge base, which contains the data of the assembly graphical representation. The knowledge base contains a set of assembly constructions and a set of constraints on the use of these assembly constructions. Using the knowledge base and a developed selection procedure, a construction assembly can be automatically chosen for the simple graphical representation. Then, 3D solid model details can be generated for the selected assembly construction.

Intelligent building assemblies are similar to libraries of "dynamic details". These details know about any important constraints on their use and take them into account each time they are used in a building.

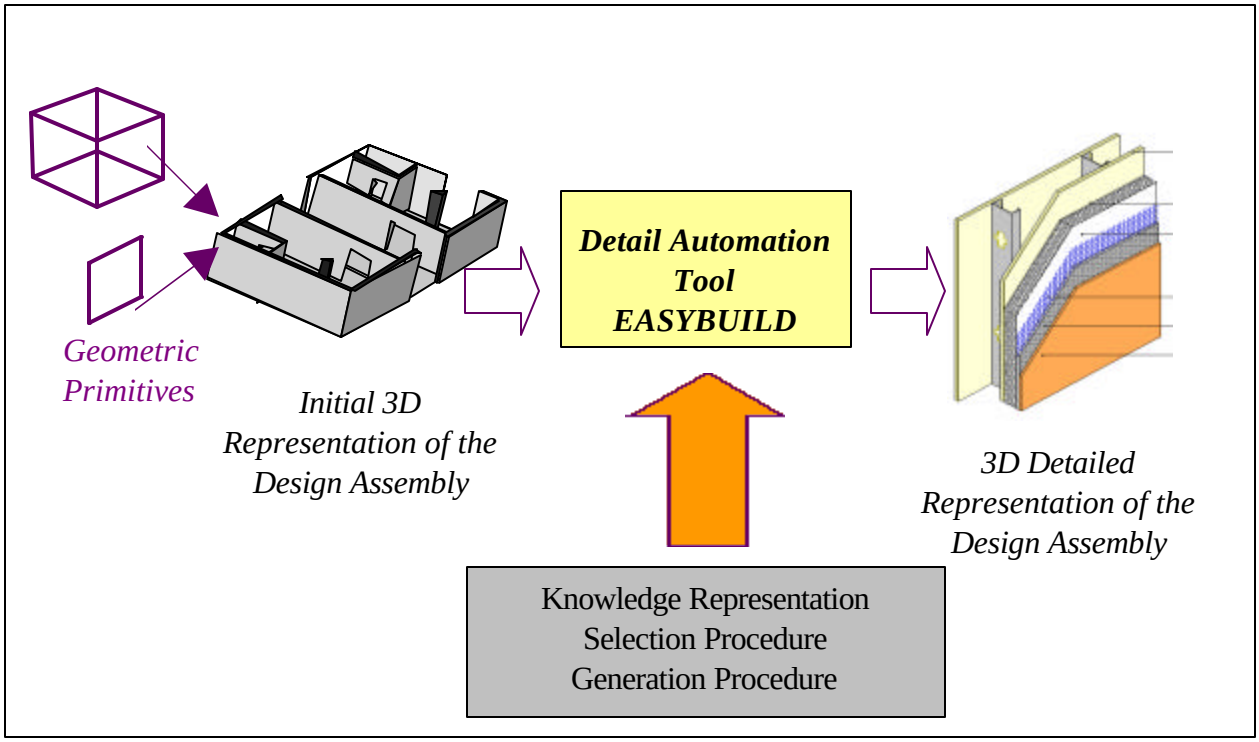


Figure 1.2. The Wall Assembly at the two levels of detail

[4] as will be described the assembly graphical representation is called a CADREP

An important point to note here is that intelligence can be added to any decomposition unit of the building. One can add intelligence to spaces, rooms or building materials. However, the decomposition of the building into assemblies allows more flexibility to the designer for describing the design and limits the variables of the design problem. The variables are limited to the choice of assembly construction and the design of the assembly (i.e. size and choice of the materials used in the assembly). The building topology (geometric design like shape and dimensions) becomes exogenous variables or constraints as will be described in chapter two.

1.4. RESEARCH OBJECTIVE

The main objective of this research is to develop a new way for selecting and generating building assemblies. Intelligent building assemblies that can be automatically selected and generated are put forward as a novel way for producing building details.

The proposed methodology is based on a defined building product model (Figure 1.3.). Therefore, the research presented in this dissertation seeks to define a building product model that can support the definition, storage and modification of building assembly constructions and their use constraints. The defined building product model breaks down the building into entities, relationships and semantic integrity rules[5] (i.e. rules that apply to all the instances of the entities). In this dissertation the building product model is defined formally in EXPRESS/EXPRESS-G (a standard information modeling method). The formal definition of the product model facilitates the interoperability with other analysis software as will be described. Also the building product model provides a standardized internal data structures for other assembly selection and generation software.

Secondly, the formal building product model is implemented as a database. The database implementation provides users with a knowledge definition interface developed to capture design and construction knowledge. The database implementation allows for specifying rules about the entities in the building product model database as well as adding/deleting entities. The formal

[5] See [Date 1995] for a comprehensive definition of semantic integrity rules.

product model can also be used to define and change meta-knowledge like the relationships between the entities.

Thirdly, an assembly selection and general procedures are developed based on the entities in the building product model. The selection procedure provides for a way to select the most appropriate assembly construction based on a set of performance criteria and their relative importance weights defined by the designer. The generation procedure allows for semi-automated generation of a detailed solid model from the abstract schematic description of the building. The generation procedure is based on describing parametrical procedures of how certain assemblies can be generated graphically.

The steps required in this research are shown in Figure 1.3. The overall developed methodology assists the designer during the steps of selection and generation of building assembly details. By following the design and information modeling methodology of the described system, intelligence can be added to building assemblies.

Specifically, the following tasks are required:

[1] A study of the building design process to answer the questions: Does the design process lend itself to the idea of automatic assembly selection and generation? How do intelligent building assemblies fit into the current design paradigm? At which stage during the design process can the automated selection be useful.

[2] A review of the research carried out related to automating the selection and generation of building assembly constructions.

[3] In order to develop the building product model a review of current building product models and modeling techniques is required.

[4] The development of a building product model is one the tasks in adding intelligence to assemblies because of the difficulty in representing the diverse knowledge and information on the constraints on use of assembly constructions. The review of product models carried out earlier forms the background for this task. The product model has to implemented in a way to

allow for constraint definition and automatic selection. The product model is used to develop the database of the building's assemblies and the various assembly constructions that could be selected. Therefore the development of a product model in a formal and unambiguous way is required.

[5] Development of a building assembly selection procedure is the fifth task. The assembly selection methodology should consider the multi-criteria aspect of the building assembly selection problem.

[6] The development of a graphical assembly generation procedure is also required. The graphical assembly generation procedure is responsible for helping the designer in drawing the 3D solid model of the components of the selected assemblies.

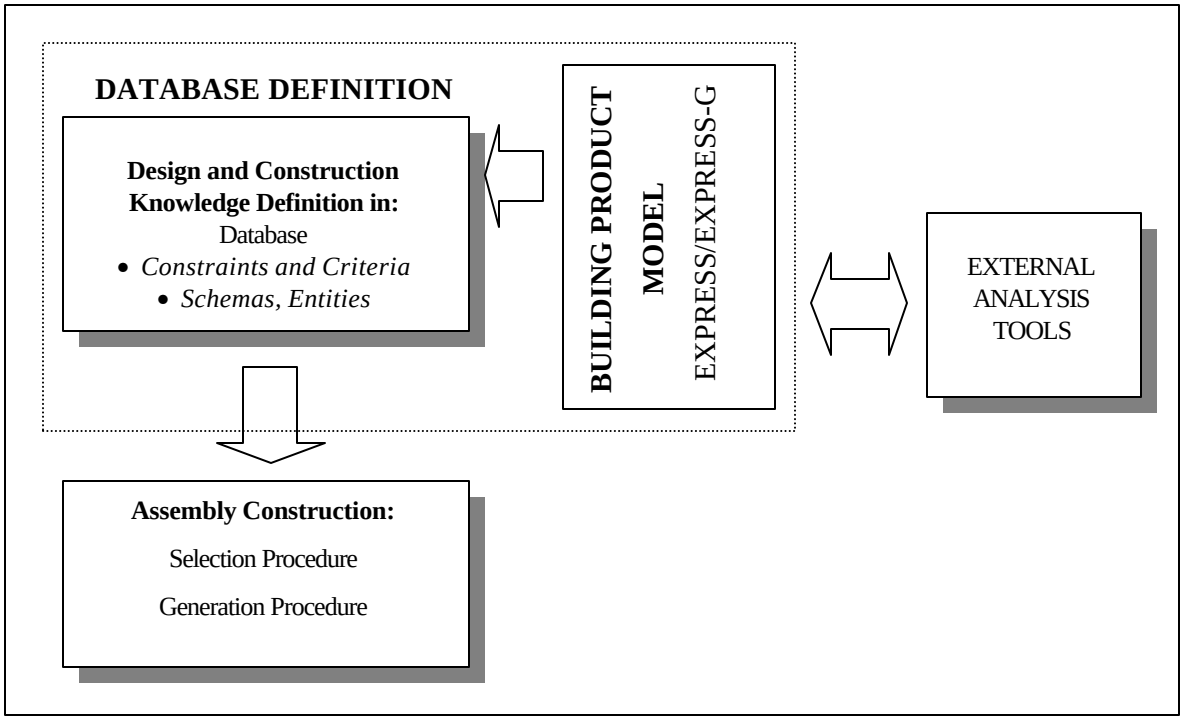


Figure 1.3. Structure of the Proposed Research

[7] Developing a prototype system as a proof-of-concept carries out validation of the idea of generating the assembly detail automatically (EASYBUILD). The development issues and the

available technologies will be discussed. An example for a specific building design will be provided.

1.5. THESIS ORGANIZATION

The dissertation is divided into eight chapters. In chapter one, a brief outline is provided for the different directions of research presented in this dissertation. The research problem is described. The objectives, specific tasks and, the limitation of the research are presented.

Chapter two provides insight on the design process and automation of assembly selection and generation. The different types and stages of design are identified. The design stage at which automation of detail can be initiated is described. The discussion is provided through a design example.

Chapter three presents a review of some methods that can be used in the automatic selection and generation of building assemblies. Examples are given from previous research efforts. In specific research efforts utilizing neural networks, case-base reasoning, expert systems and constraint modeling are described.

Chapter four introduces building product modeling. The objectives of building product modeling are identified. The different information modeling techniques used in building modeling are described. The various building models in the literature are described according to the information modeling technique they utilized. The formal information modeling technique used in this dissertation (EXPRESS) is introduced.

Chapter five introduces the developed building product model and the database. The different entities of the building are identified in a conceptual model. A complete description of each of the entities is provided along with the attributes, relationships between them and formal specification. The different semantic and modeling propositions are described in the discussion. The supporting database design is presented, along with a description of how the database can be used for automatic selection and data exchange.

Chapter six describes the assembly selection procedure and the assembly generation procedure. The assembly selection procedure is based on assigning weights to important criteria and performing a selection based on the defined objective. Examples of the selection criteria are given. The assembly generation procedure is based on defining a set of parametric operations that allow for the generation of the building detailed solid model. The parametric operations are described and an example is provided.

In chapter seven the proof-of-concept prototype EASYBUILD is described. The various modules of EASYBUILD are identified along with the different inputs and outputs of each module. The technologies utilized in each module are described. An example of a flat-plate-concrete-construction residential building is provided for validation.

Chapter eight summarizes the research presented in this dissertation. The contribution of the research presented here is identified. The directions for future research and extensions to the ideas in this dissertation are identified. Finally a vision for the design and construction practices and, the benefits associated with this vision are described.

1.6. LIMITATIONS

This dissertation introduces the idea of automatic assembly selection and generation for building design. In order to realize the automation of building details specific tasks are required. This dissertation deals with the development of the building product model, assembly selection and generation procedures, to support assembly design and construction intelligence. Although examples of domain knowledge and processes are given, the specific domain knowledge and processes are not the focus of this dissertation. A prototype system (EASYBUILD) is developed as a proof of concept. The system supports only one construction type (concrete flat-plate construction).

CHAPTER 2. AUTOMATION AND BUILDING DESIGN

This chapter provides insight on the design process and automation of building assembly selection and generation. The different types and stages of design are identified. Intelligent building assemblies are introduced as a tool for automating part of the building design process. The design stage at which automation can be initiated is described. The discussion is provided through a design example.

Design is satisfying constraints and meeting objectives
Design is problem solving
Design is decision making
Design is reasoning under uncertainty
Design is search
Design is planning
Design is an interactive process
Design is a parallel process
Design is an evolutionary process
Design is a mapping from functional space to a physical space
Design is like a game
Design is creative and inexplicable.

Figure 2.1. The Different Views of Design [Adapted from Miles et al. 1994]

Design in general, and in architecture specifically is one of the most diverse activities of humans. It has been approached in many ways and has been formulated using several principles. It is very hard to define what design actually is. For example Figure 2.1. shows some of the views on design. In order to assess the applicability of the idea of intelligent assemblies and see when automation can be useful, we need to take a look at the stages of the architectural design and the types of architectural design activity.

2.1. DESIGN STAGES

In the literature on building design, there are two ways to decompose building design into different stages. The first is to look at the building design activity from the point of view of design theory[6]. The second is to look at design from the point of view of building delivery.

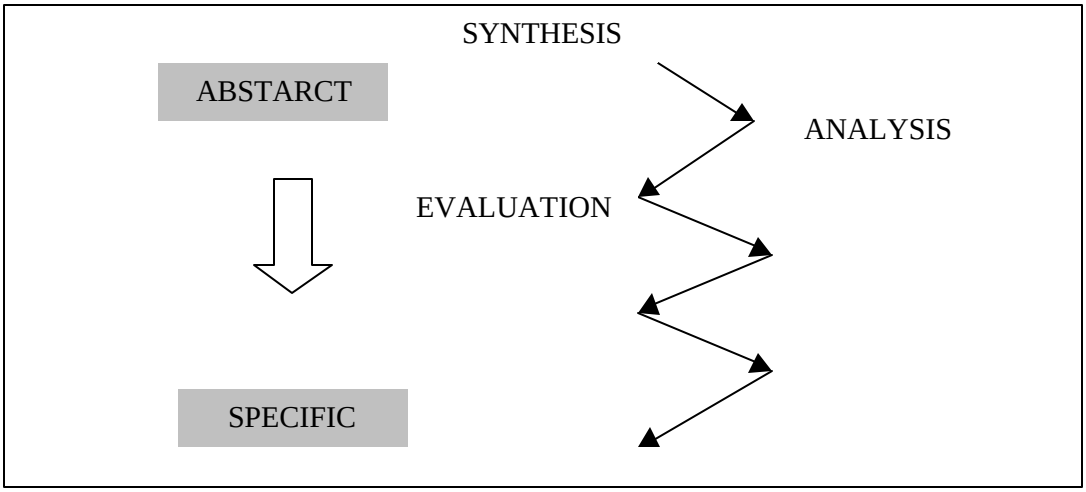


Figure 2.2. The Design Cyclic Process

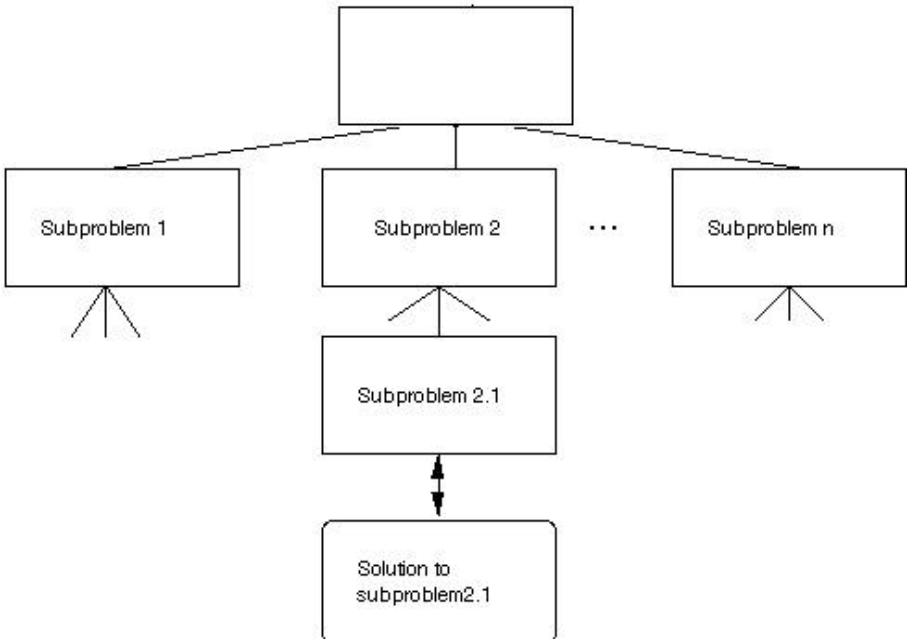
First Approach: Most of the researchers of design theory agree that design consists of three identifiable phases; [Radford et al. 1988] problem analysis, design evaluation and design synthesis (Figure 2.2.). The design problem and its context are analyzed to determine the problems and possibilities of the design. An initial design is then created in the design synthesis stage. This design is evaluated according to a set of criteria. The process continues until an evaluation of the design proves satisfactory. The design activity is therefore a cyclical process proceeding from abstract to specific.

Design synthesis, which is primarily a manual process, is the core of the "Analysis/Synthesis/Evaluation" approach. Maher identified three kinds of design synthesis; decomposition, case-based reasoning, and transformation [Maher 1990]) Figure 2.3. The design

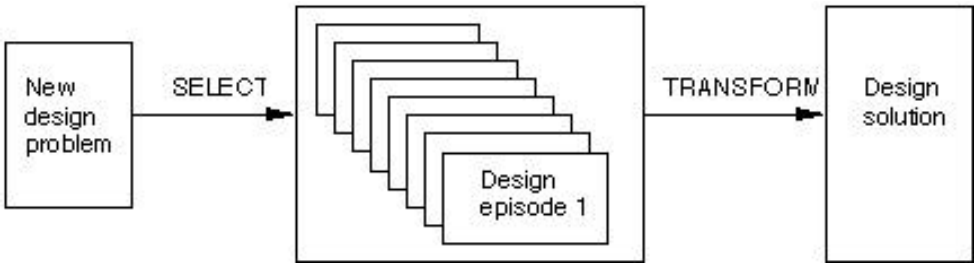
[6] This deals with the process of "how designers design" and the stages of this process.

can be synthesized by decomposing the problem into a set of sub-problems and then tackling each sub-problem (Figure 2.3.a). Another alternative is to synthesize the design by reasoning with previous cases (Figure 2.3.b). The design can also be synthesized by sequential transformation of an initial concept (Figure 2.3.c).

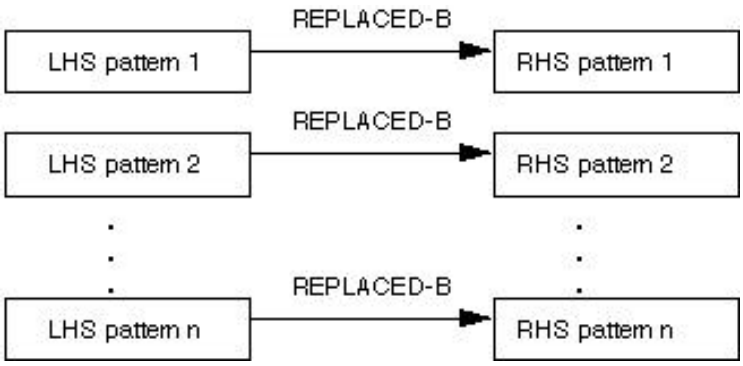
Since the stages of design in the "Analysis/Synthesis/Evaluation" approach are cyclic, and can be formulated in different ways (as we described with design synthesis) it is difficult to identify and discuss specific design stages using this approach.



a) The Decomposition Model



b) Case-Based Reasoning Model



c) Transformation Model

Figure 2.3. Models of Design Synthesis [Maher 1990]

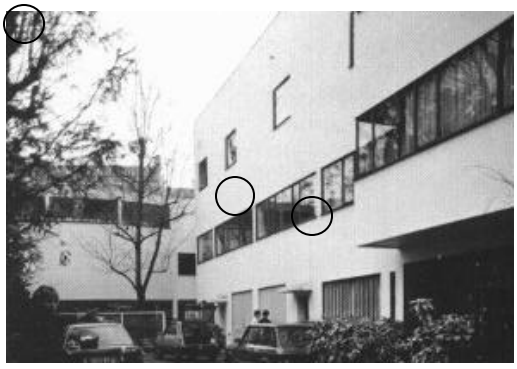
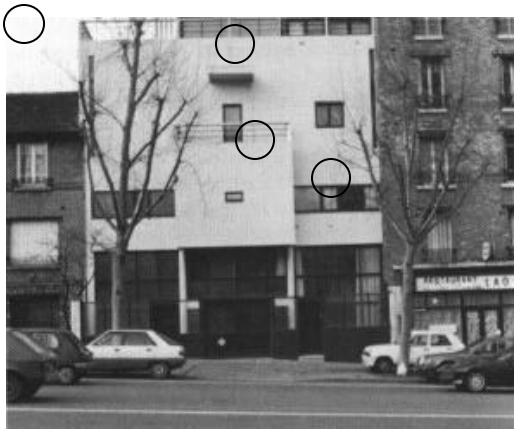
Second Approach: Due to the fact that most projects do have a beginning and end, and that they are defined by contracts for planning and design, there have to be stages for the building design delivery of projects. These stages have been described in several academic and professional publications [CSI 1994], [Laseau 1980]. Laseau [Laseau 1980]proposes the following stage; (feasability study and programming), schematic design, preliminary design, design development, contract documents, shop drawings, (and construction). However, these stage are usually compacted to conceptual (or schematic design), design development, and detailed (or construction documents) stages. The latter three stages are more widely used in the AEC industry. These three stages are very useful functional description of the design *stages*, however we also need a description of the various *types* of design.

2.2. DESIGN TYPES

Researchers have proposed three types of design activity; parametric design, innovative design and, creative design [Schmit 1990]. Parametric design (or sometimes-called routine design) is characterized by the existence of a prototype for the design. This prototype will have several parameters (e.g. shape, number of rooms, number of floors, etc…). We can change those parameters to reach a new design (hence this *type* of design can also be called "*prototype refinement*"). The prototype is not fundamentally altered during this process [Schmit 1990]. For

example, a modular house design that undergoes minor changes of dimensions to produce a new house would be considered a prototype refinement of the original design. Figure 2.4. shows four different buildings by Le Corbusier (a famous architect of the modern era) [Ferleger et al. 1987]. The buildings are an example of a certain design pattern [Alexander et al. 1977]. This pattern was changed parametrically to produce the various building designs. The design for these buildings is an example of parametric design.

The second type, innovative design, deals with the adaptation and combination of two or more design prototypes. Each of those prototypes posses some of the properties required for the final design. Sometimes also innovative design is attainable with modification of the prototype [Schmit 1990] (For example a house design that share common features from two previously designed houses).

	A This is an example of a Construction Type. This is a concrete construction type, which is a typical example of Le Corbusier residential design style. It is characterized by flat slabs, straight columns. The freed the façade and the interior walls from any structural constraints. This construction Type has specific assemblies that it uses.
	B Examples of assemblies are Exterior Walls, which are not bearing and are present in all of the four examples here. Since the designs belong to the same pattern, notice that the wall assembly is constant i.e. is used in the same way in all designs. The Wall assembly can have sub- assemblies.

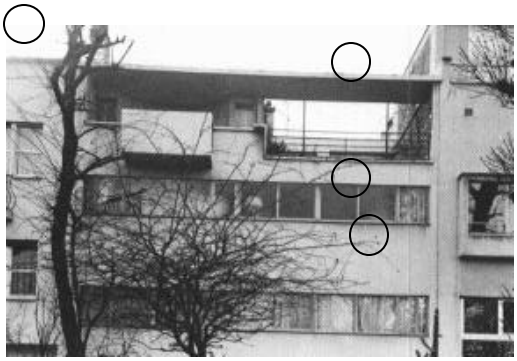
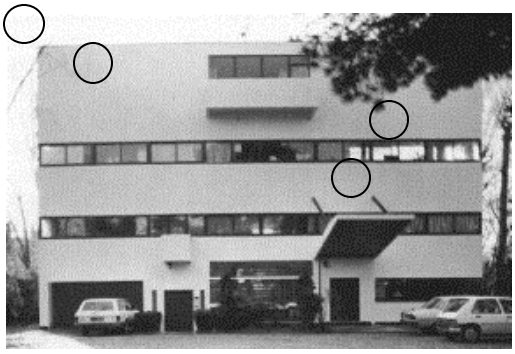
	C The windows here are an example of such sub-assemblies. The windows can only exist in the wall assembly and have the same form and design from one building to another.
	D Another example of an assembly here is the canopy. Canopies are used in all four buildings. A canopy is another example of a construction type assembly. It is used to describe the design concept of the building (it is a slab sub-assembly because it can not exist without the slab).

Figure 2.4. An example of parametric design

The third type of design, creative design, is rare. In most cases of creative design there is a design prototype creation. Another interesting fact about creative design is that the created design itself can sometimes influence the original design goal.

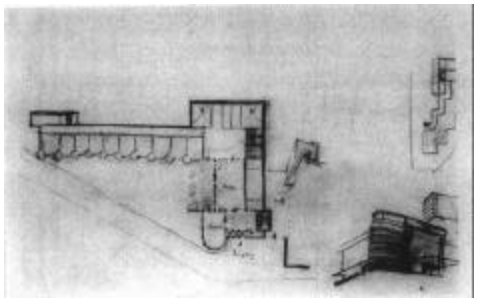
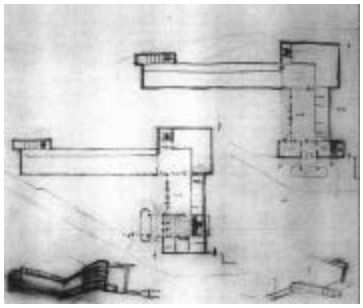
Now that we have described the stages and types of design, let us consider how automation can fit in the design stages (i.e. when can automation of assembly selection become helpful during design). Also we will consider how automation fits in with the different design types. We will use a building design example as a demonstration.

2.3. ASSEMBLY SELECTION AUTOMATION AND DESIGN

Figure 2.5. shows the "Haus Sonnenfang" building in Berlin designed by Walter Gropius. The building design is shown at different stages (defined earlier as schematic design, design

development and construction documents). A manual of practice (The CSI Manual of Practice [CSI 1994]) states that at the schematic stage usually the designer has sketches, renderings, and conceptual plans and elevations. The designer during the schematic design phase (also referred to as the A/E) also has preliminary project description and cost projections. The first two drawings in Figure 2.5. (2.5.a, 2.5.b) might qualify for that stage. During design development the designer prepares plans, sections, design criteria, and prepares outline specifications. (Figure 2.5.c, 2.5.d). During the construction documents stage detailed drawings and specifications are prepared (Figure 2.5.e).

Also during the design development stage the various building assembly constructions are selected. The design schematic design becomes a kind of a constraint. This is because the assemblies that result in the best performance for the prepared schematic design are selected. The various criteria that affect the selection are enumerated and often a set of assemblies that trade-off some criteria over others is selected. There might exist some other constraints on the use of the assembly construction that hinders it from being selected.

	a. Schematic Design These are the preliminary sketches of the project. The design at this stage is not stable yet and can change to accommodate a new concept or idea. However, usually the main concept is there. In this case for example the L shape form (of course there might be others). This does not mean that this main concept can not change.
	b. Schematic Design The design at this stage is becoming more stable and becoming less *likely* to change. There is a shift now to look at a few design concept details within the framework of the main concept (like the soft corners on

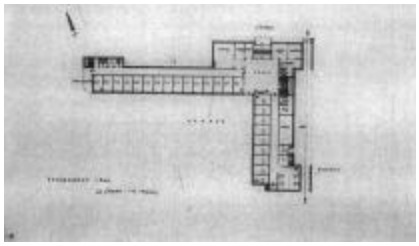
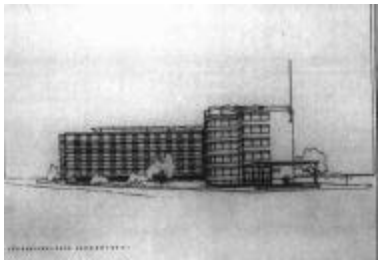
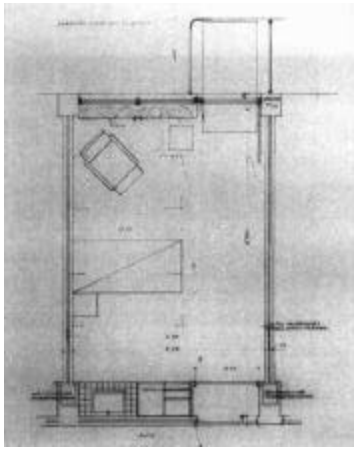
	staircases in this case). The building is continuously evaluated.
	c. Design Development Possible Input Stage. At this stage space/activity allocations are worked out and specific form features are designed (e.g. the building footprint and projections).
	d. Design Development Possible Input Stage. This is the input stage where we can get the design from the architect and start working on it. The design is almost stable and we now know the form of the building, the spaces/activities and some feature details (like the solids on balconies in this case) Other specific requirements can exist like a certain type of finishing material.
	e. Output Stage This is the stage we are trying to get to at the end using automation. Since an important part of the design concept can be at that level of detail, the designer has to specify how to get from the last step to this one and be able to specify certain design requirements (e.g. a glass curtain wall assembly). Automation should only be targeted to technical and repetitive tasks (e.g. sizing the columns, placing the window mullions).

Figure 2.5. An example of design stages

At this level of design evolution the design is described at a level of abstraction where several design tasks are already resolved. These tasks are:

- Architectural massing and morphology (the basic building form and geometry, like height, width, projections, rooflines and skylines) are worked out.
- Activity-space relationships and programming issues are resolved.
- The spaces are defined and their functions and spatial relationships are known (this is known as the topology of the design).
- Certain constraints or designer requirements exist. For example, the designer might want a stone veneer on one wall, or a certain tint of glass for a specific window.

These design tasks can be classified as creative design tasks. These tasks are also mainly heuristic and involve a lot of design synthesis (by any of design synthesis models described above). We have said that design synthesis is mainly a manual procedure.

Notice that the building design at this input stage is described in a medium level of abstraction (i.e. detail of the design description). This means that the building design is described in a less generic way than a conceptual design and a less specific way than a detailed design. For example we know we are going to use a "*Cast In Place Concrete Post And Beam*" and not just a "concrete system". However we do not know the sizes and the reinforcement we are going to use for our assemblies. The description of the building design in terms of assemblies allows us to specify the results of all the above design tasks. Therefore, the use of a description of the design in terms of different assemblies, is very suitable.

- Considering that we have a design described in a particular level of abstraction, then for certain design cases we can automate the selection of the best type of assemblies based on user requirements and criteria. For example choose the best type of external wall assemblies to reduce annual heat load, or the best type of roof assembly for durability, etc… The criteria can be stored in a database and evaluated for each new design concept automatically. Also a

set of constraints on the use of the certain assembly constructions can also be evaluated automatically to eliminate the inapplicable assembly constructions that do not satisfy certain conditions for the particular design. Also automation can be used to generate a 3D solid model of the building in order to extract the building details [Nassar et al. 1999a].

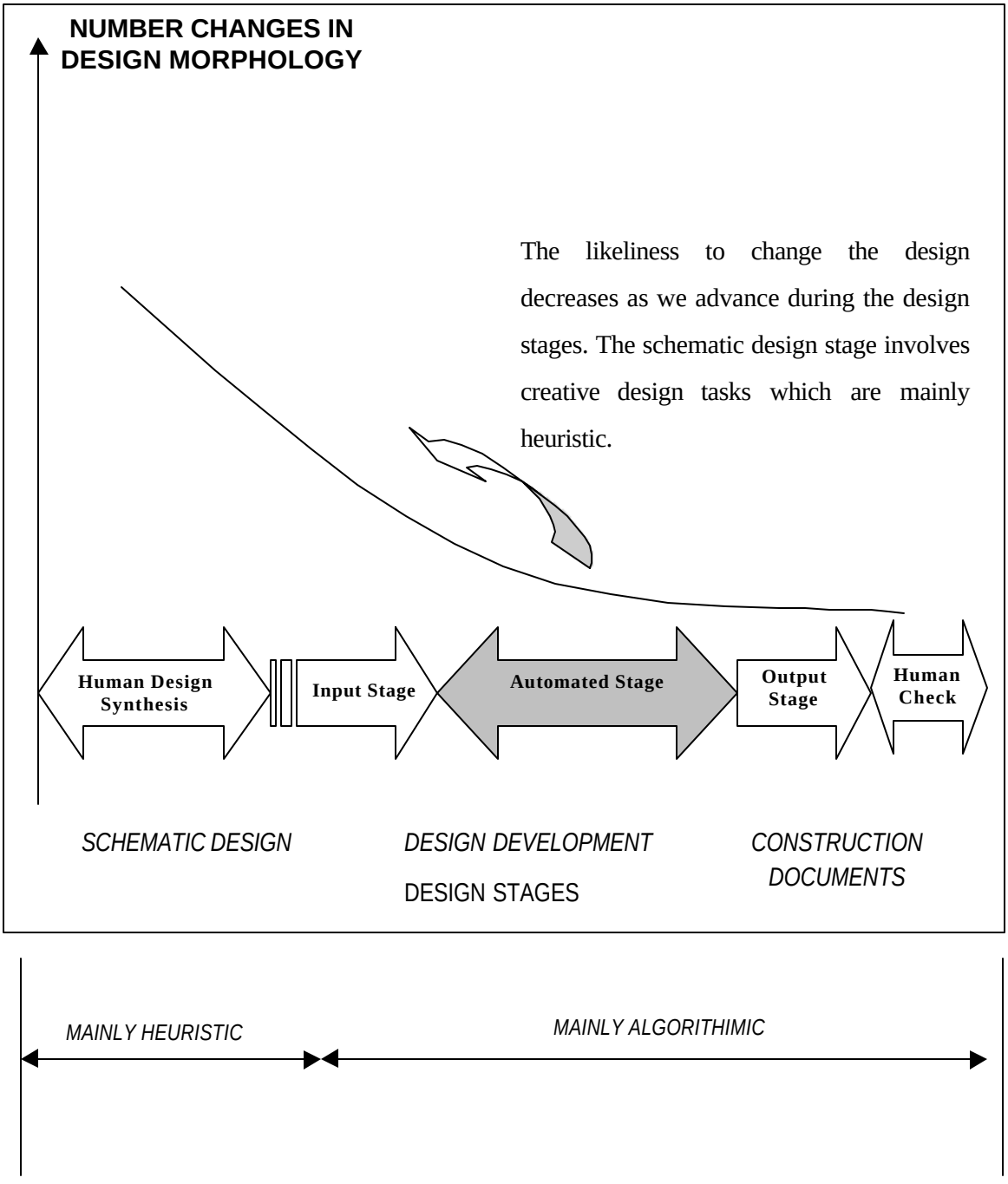


Figure 2.6. The Design Process

In most cases, these tasks can be classified as either routine or parametric design tasks stages [Nassar et al. 1999b]. These tasks are classified as parametric or routine because generally there is no synthesis is required. This makes these tasks suitable for automation. Therefore a design support tool that automates the design development stage to produce the construction documents, is feasible, because of the routine (or parametric) nature of the required design tasks at this stage. Creative design tasks are harder to model and automate.

Figure 2.6. shows the design process as we see it. This is only one model of design as we have described, that is good for practical consideration. After the schematic design is prepared the designer starts design development. At this stage the design is not yet very stable (i.e. many changes can be made to the design morphology, i.e. building form and shape, like height, aspect ratio, projections etc…) and can be modified to accommodate new ideas that might arise. Once the designer has undergone a major part of design development, and the creative design tasks described above are carried out, the design becomes more stable. The automated assembly selection can be done during design development by evaluating the constraints on use and the selection criteria.

Human checking and verification is required at this stage before construction. Verification is very important for two reasons. Firstly, it allows us to check the output from the automation. Secondly, this checking might in turn provide new insight to the design i.e. we can go back and make changes at the design development stage (e.g. change our mechanical system or our wall assemblies) or at the schematic design. For example, the designer may change a basic design variable (e.g. building height) and then proceed again with design development. Changing basic design variables would be the case in creative design and although this can be useful sometimes, it is not always practical or helpful.

2.4. SYNOPSIS

So we can conclude that a description of the design in terms of different assemblies, is very suitable for representing the design at the level of abstraction where most of the creative tasks are carried out and very little design synthesis about the form and shape of the building is required. The representation of the building in terms of assemblies, can then be used to perform routine and parametric design tasks, like assembly selection, to reach the final detailed design.

Intelligent building assemblies can be used as a design tool to help the designer advance through the design development and construction drawings design. Also, intelligent assemblies can be used to in evaluating constraints and criteria for the use of various assembly constructions. Today, the lack of such a tool and manual selection of assemblies can cause the selection of assemblies that are not the best for a particular design. Also automated assembly construction selection can provide insight which allows the designer to quickly experiment with different building assemblies. Also costly omissions and errors could be avoided, in addition to saved design time.

It is important to mention that the designer will have certain constraints and requirements on the building design. Therefore our assemblies must be able to accommodate user requirements for the final detailed building. In the next chapter we start reviewing the literature on assembly selection and generation procedures.

CHAPTER 3. ASSEMBLY SELECTION AND GENERATION METHODS

This chapter provides an overview of different methods that can be used in automatic selection and generation of building assemblies. Different previous research conducted to aid designers at the building assembly design level will be presented. This review is important before we describe the proposed method for the selection and generation of building assemblies, which will be described in chapter six. We will start with research on building assembly selection and then introduce assembly generation.

3.1. BUILDING ASSEMBLY SELECTION

The investigation of the research directions led to identifying the following three areas of research;

- Neural networks
- Case-based reasoning
- Expert systems

For each of these research directions the underling theory will be briefly introduced. Next a description of the actual research efforts will be presented and the limits and benefits of each direction will be identified.

3.1.1. Neural Networks

Here a brief look at neural networks is given to explain their use in selection of assemblies. A complete description can be found in [Garza et al. 1996], [Silvia et al. 1997], [Coyne et al. 1992], [Coyne et al. 1993]. In neural networks, knowledge is stored as weights and threshold values within a network [Coyne et al. 1990]. Nodes in the network are called 'units'. There are input units, hidden units and, output units, all connected by directed arcs in some configuration (Figure 3.1.).

Input units receive values from the 'outside world'. Hidden units receive values from the input units and transmit them to the output units that produce the results. The variables of a neural network are weights on the arcs, threshold values on the units, and input values of the units. It is simplest to think of the units as receiving and producing binary inputs and outputs (0 or 1). The output value of a unit is equal to the sum of the product of the weights of the arcs directed towards the unit and the output value of the connected unit. A unit will fire (producing a value of 1) if the net value is greater than the threshold, or else a value of 0 is produced. This operation propagates throughout the network to produce a set of output values in response to a set of input values.

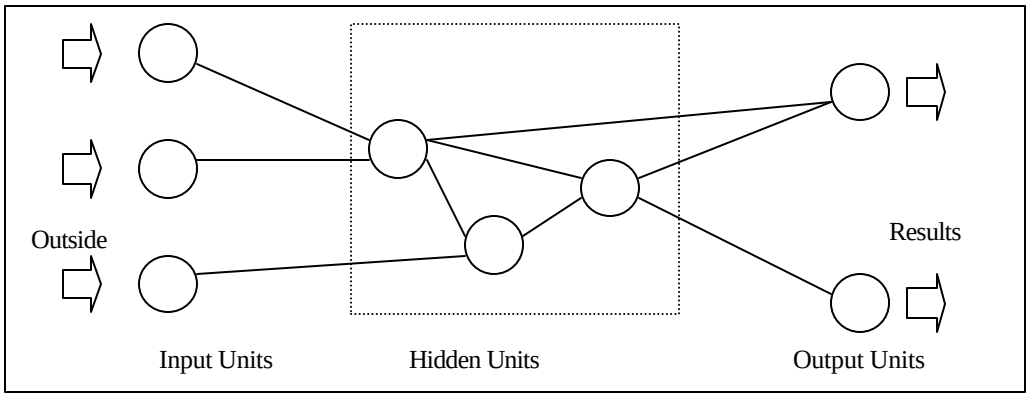


Figure 3.1. A simplified Neural Network

There are a number of different neural networks described in literature. Most of these networks share the same principles. The principles of neural network can be most easily explained using the

simple network in Figure 3.2. In this network, each input unit is connected to the output units with a directed arc of weight 1. The output unit has a threshold value *q=0.*

If we present this network (Figure 3.2.a) with a set of inputs, then a particular output would be generated. The output is calculated by summing the product of each input unit with each weight and comparing the result with the threshold (e.g. for Figure 3.2.b *1x1+1x0+1x1=2>0* therefore the output is 1).

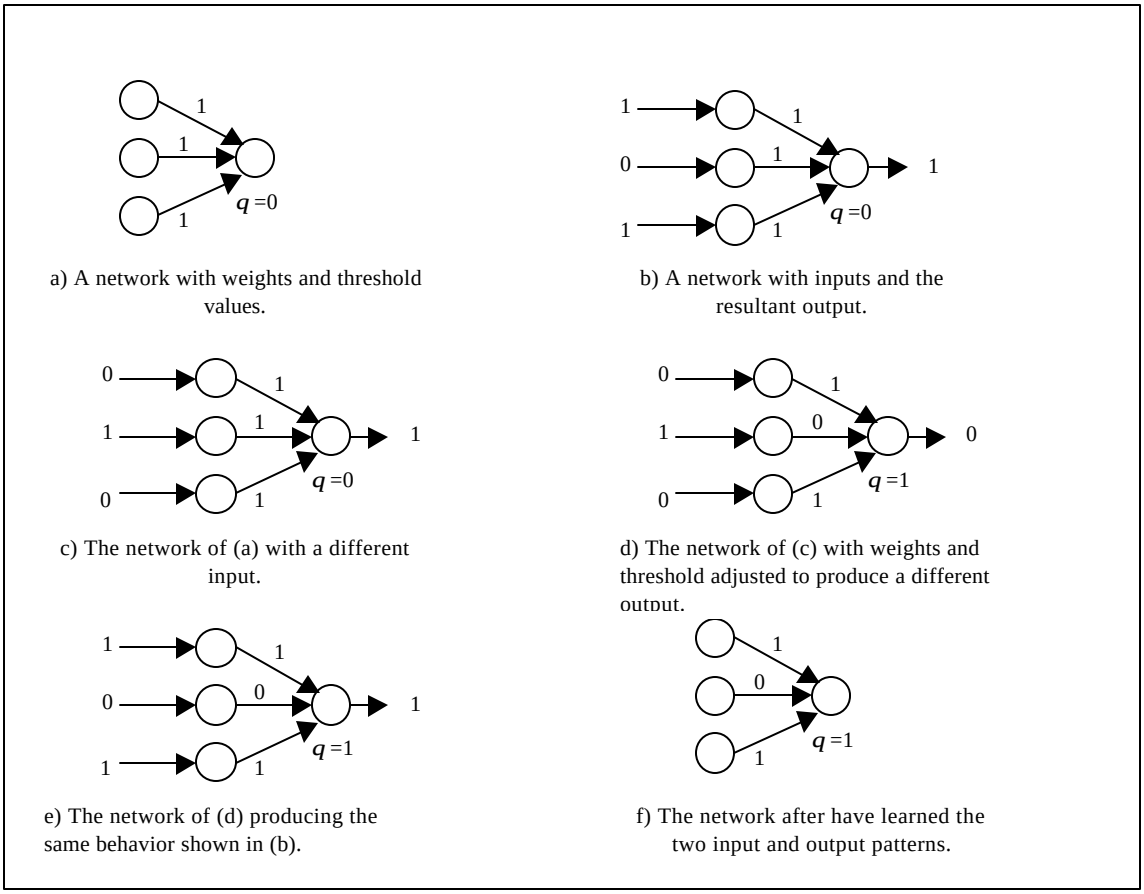


Figure 3.2. Neural networks adaptation (adapted from [Coyne and Newton 1990])

Suppose for example that we wish the network to generate an output of 0 for an input pattern of {0,1,0} (Figure 3.2.c). This could resemble a network that chooses whether an assembly

construction could be used or not based on a set of inputs that can resemble design situation. So for example the output 0 could mean that the assembly is not applicable and an output of 1 would mean that the assembly is applicable. The inputs could resemble design situation information like whether the building is in a wet climate or not (of course, this is a simple example for demonstration, but many inputs can also be used to resemble different other factors).

We need to modify the weights and threshold of the network as shown in Figure 3.2.d Note that the network also produces 0 for the original input pattern (Figure 3.2.e). The final network with the modified weights and threshold is shown in Figure 3.2.f. The algorithm for modifying the weights and threshold in this kind of a binary network is shown in Figure 3.3. Other types of neural networks will use a least square method to minimize the difference between the produced and the required outputs.

1. Calculate the value of the output unit from the given input pattern.
2. Compare this predicted value with the teaching output presented to the network
3. Inspect each of the input units in turn
If the input value =1 and predicted output = 1 and teaching output = 0
Then subtract 1 from the weight of the arc emanating from that unit and add 1 to the threshold of the output unit.
If the input value =1 and predicted output = 0 and teaching output = 1
Then add 1 to the weight of the arc emanating from that unit and subtract 1 from the threshold of the output unit.

Figure 3.3. Algorithm for modifying weights and thresholds

Modification of the weights is seen as teaching the network. The teaching operation involves cycling through the set of given input-output patterns and modifying the weights and threshold accordingly. However when any input node has the same value in more that one input-output pattern there is no assurance that a system of weights and threshold that satisfy all input-output pairs can be found.

Therefore an extension to the network has to be made. The difference between the net input and the threshold to an output unit actually gives an indication of the probability of the unit to fire. There is a convenient function for distributing the probability between 0 to 1. Calculated from the net inputs and the output threshold. The *logistic function* maps the difference "D" between the

input and the threshold to a unit onto the probability value "p" depending on a constant "T" (T is the slope of the function curve and is also called the temperature)[7]. The higher the temperature the more random the predicted values at output units and the greater the means of escaping from a system of weights and thresholds, that are sub-optimal).

The function is,

$$p = \left[1 + \exp\left(-\frac{D}{T}\right)\right]^{-1}$$

Using the logistic function we can calculate the likelihood of the predicted value being 0 or 1 and generate 0 or 1 at random according to this probability. Surprisingly, this randomness over a large number of learning cycles produces a more accurate system of weights and threshold.

Neural networks have been used in representing design information and finding implicit links in that information. In his earlier research [Coyne and Newton 1990] explored the notion that certain ideas promote the recollection of other related ideas. In his research Coyne explored the notion of association of hypermedia cards. The author postulates that it is easier to think in terms of linking cards than linking schemata, frames, or scripts.

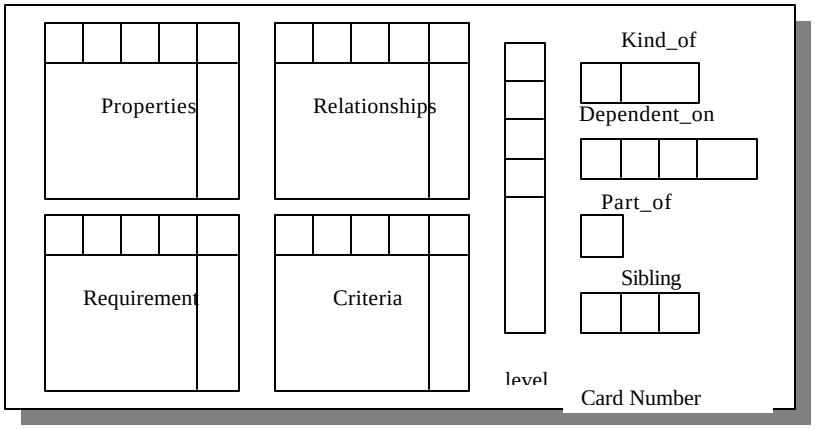


Figure 3.4. An example of a hyper-card

[7] The analogy is made between this process and thermodynamics principles.

Cards can contain almost any kind of information. A useful basis for linking cards might be similarity. For example, Coyne describes cards with building elements like lintels and podiums, which are similar, in that they both appear as horizontal elements in elevation. (In our research the cards could also contain different building assemblies). While browsing through cards it would be useful to be able to tell the system: 'show me a card that is similar to this card.' The system would need to know what is meant by 'similar' in this particular context. One way of telling the system is to point to other cards that are similar, then tell the system to find another one that has common features. Deriving the knowledge by which objects may be regarded as similar is essentially a classification problem. Automated classification systems are generally based on the idea of analyzing a wide range of examples demonstrating a particular concept, that is, examples that belong to a particular class. The system then detects and generalizes on features of the descriptions.

The relationships between the card were of primary importance in Coyne's research rather than the content of the cards. There are five kinds of relationships between the cards: *Kind_of*, *Dependant_on* , *Parts_of* , *Sibling_* and, *level*. Coyne presents four examples that demonstrate the application of the neural networks in generating associations between the cards.

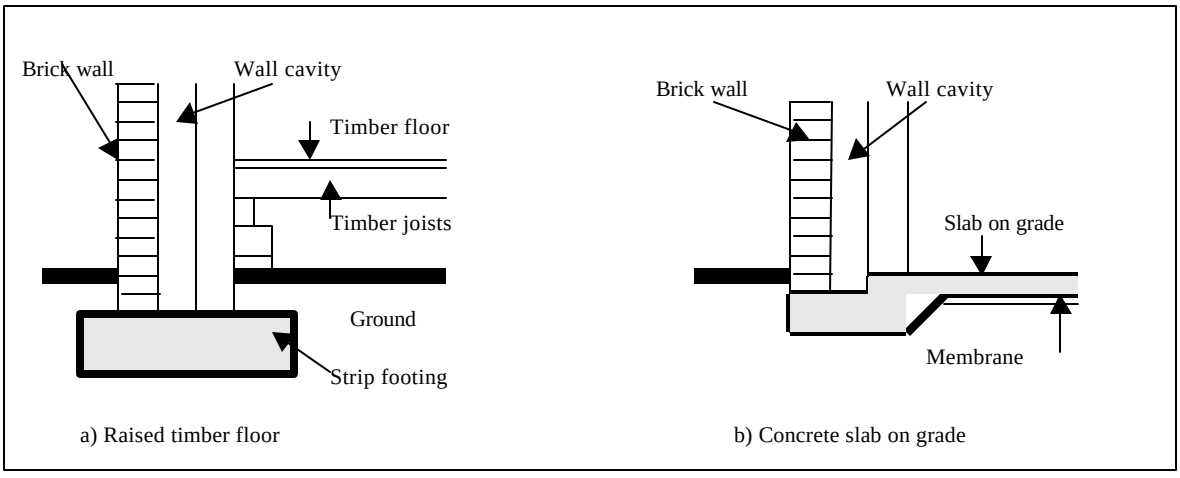


Figure 3.5. The sub-floor construction used in Coyne's research, adapted from [Coyne 1991]

The first example Coyne presented how a neural network can be taught a particular classification rule. Teaching is done by presenting a series of input patterns and their corresponding output pattern. In the second example, Coyne explores how the performance of the system changes significantly with the number of learning cycles. In the previous two examples, the linkage rules were explicitly defined by means of input and output patterns. The third example demonstrates that an implicit classification rule can be similarly learned. The last example presents a similar approach but with multiple criteria for the linkage rules.

Later, Coyne [Coyne 1991] presented an example of the use of neural networks relating to domestic sub-floor construction in buildings (Figure 3.5.). A network was presented with a number of examples (a training set) in the form of abstracted sub-floor details. The examples also included some descriptions and properties of the buildings to which the details belong. The examples tended to fall within certain unstated categories, such as 'raised timber floor', 'slab on ground', or 'strip footings and piers'. A typical associative training method was adopted. The system sets up weights between every pair of descriptors (these descriptors are the various parts of the sub-floor detail and the properties of the buildings to which they belong). These weights reflect the degree to which descriptors occur together in individual examples. Where a pair of descriptors do not occur together an inhibitory weight is set up between them. The system also establishes threshold values that provide a measure of the overall support for each unit.

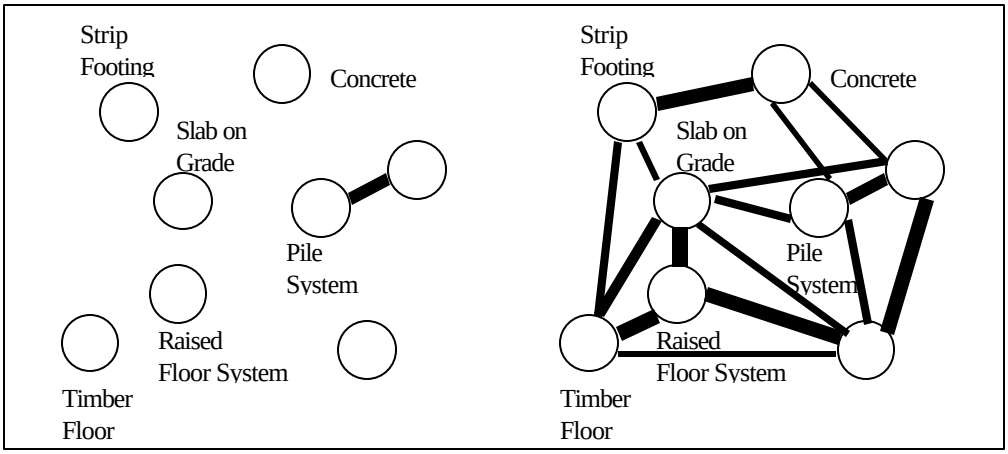


Figure 3.6. A network without and with weighted connections, adapted from [Coyne 1991]

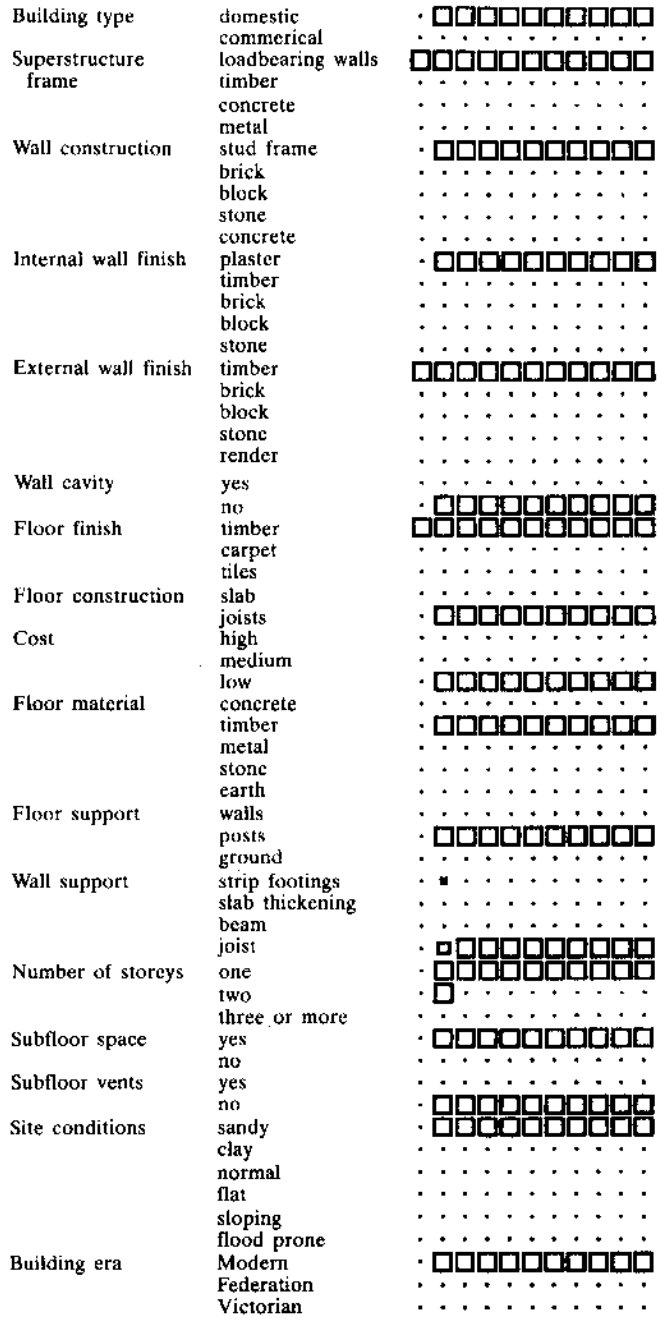


Figure 3.7. Successive states pertaining to a timber floor building with posts supporting the floor and joists supporting walls. Each column represents a different example. A square indicates a feature is present in the example. A dot indicates that it is absent. The size of the squares indicates the strength of the activation value for each unit [Coyne 1991]

A network that has been adequately 'trained was used to complete a partial pattern of descriptors. This was called the simulation phase. It does this by cycling through all the descriptors and giving them an activation value depending on the degree of support received from all other units relative to the threshold value. Over a series of iterations the system settles on a set of descriptors that are mutually supporting. So, for instance beginning with the descriptors "concrete floor' and 'domestic building type', the system settles on a description of a 'slab on ground' sub-floor detail. Figure 3.6. shows an example of the output of such a system before and after being exposed to the training iterations.

A very interesting property of the system from the viewpoint of design was discovered in this research. In one of the trials after the system has learned from successive states (as shown in Figure 3.7.), two units that did not occur together in any of the training sets were presented to it; 'timber floor' and 'concrete slab'. The resultant detail suggested a new entity. This presents an exciting system that supports emergence of new artifacts.

This research presented very useful ideas. The research shows that neural network can be used to find implicit links between assemblies and components of assemblies. However, the neural network approach suffers from certain limitations. Firstly the output of a neural network is not exact and changes from a design case to another. For example, if a neural network is presented with a set of inputs twice the output could be different for each input set[8]. Secondly, it is difficult in neural networks to add exact knowledge like fixed rules, which are important when selecting and designing assemblies. These types of rules can be represented easier in expert systems which will be described later. Thirdly, although it is possible to teach a neural network assembly performances, the evaluation is usually implicit (i.e. inherit in the learning set).

In the next section we will review the second research direction found in the literature, case-based reasoning. Case based reasoning can also be used to select assemblies based on their use in a set of previous case histories.

[8] Depending for example on the logistic function used, its variables (T, D) and the sets of weights.

3.1.2. Case Based Reasoning

One of the characteristics of design is that designers rely extensively on past experience in order to create new designs [Garza and Maher 1996]. Being able to reuse this experience requires recall at the appropriate times. In case-based reasoning systems, experience is captured and organized as a set of historical cases stored in a case base. Similar cases are then recalled from this case base, either to resolve problems or provide recommendations for the problem at hand [Yau et al. 1998]. Each case has several slots[9]. Each slot carries a particular value describing the specific case.

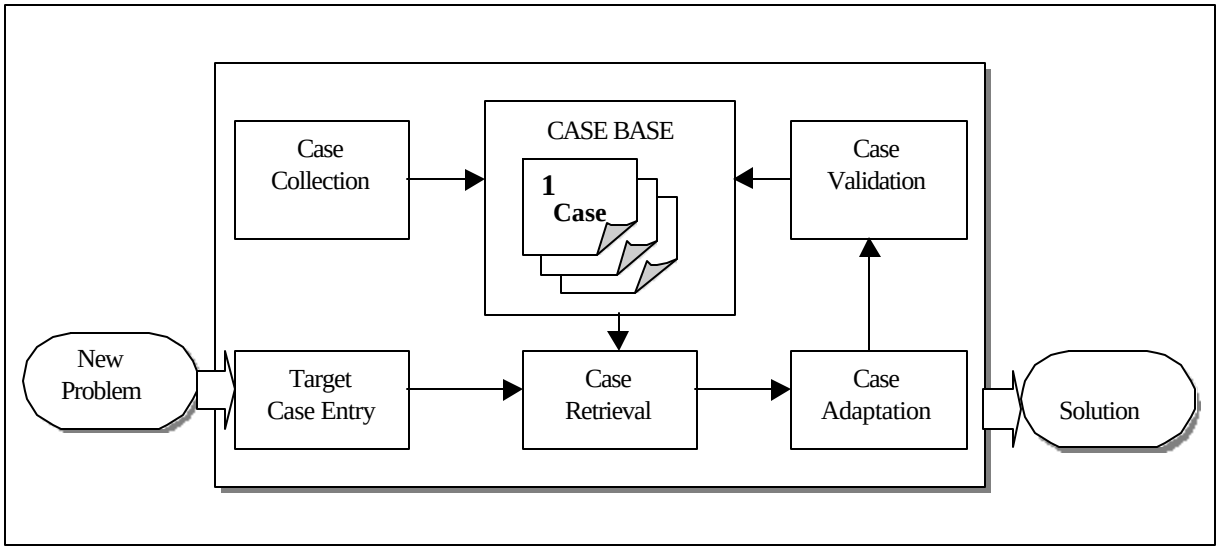


Figure 3.8. The Case-Based Reasoning Procedure [Yau and Yang 1998]

The overall methodology of case-based reasoning is shown in Figure 3.8. (detailed discussions on case-based reasoning can be found in [Garza and Maher 1996]). Cases are collected and defined in the case base structure. The case base stores previous cases as well as newly validated ones. The new problem is defined via the target case entry. Next case retrieval is performed to retrieve the most suitable case. Case adaptation generates a solution for the new problem by comparing the new problem with the retrieved case using a predefined adaptation procedure. Finally the case

[9] Fields in database terminology

is validated. Validation verifies that the solution of the new problem and merges the problem and the solution to the case base.

Perhaps the most important aspect of the case-based approach is the retrieval process. This is usually done on the basis of evaluating the similarity[10] of the new case with all the cases in the case base. The similarity index can be used to accomplish this. The similarity index SI is given by;

$$\mathrm{SI} = \frac{\sum_{i=1}^{n} (W_i \times SS_i)}{\sum_{i=1}^{n} (W_i)} \times 100$$

where n is the number of slots, i is the slot number[11] and W represents the relative weight (importance of the slot). SS_i is the similarity score. SS_i is determined on the basis of the slots' values of the target case and the case in the case base. The closer the value of the two slots (the target case and the one in the case base) the higher the SS_i. SI is normalized on a scale of 0 to 100 for easy comparison. The above equation indicates that each case has a similarity index with respect to each new problem. A higher similarity index indicates that a specific case more closely resembles the new problem case. Next we will describe two examples of case based reasoning systems in design.

CASTLES is a case-based retaining wall selection system in which the case base consists of 254 previous retaining wall cases. CASTLE is constructed in ESTEEM^{TM}, a commercial windows-based shell for case-based reasoning running in Microsoft Windows 3.1 environment. An illustrative example of the case base is seen in Figure 3.9. The various slots and their values are shown for a particular case.

---

[10] This is the 'degree of closeness' between the problem case and the case base.

[11] As mentioned before slots describe the design cases. For example in a structural system retrieval system, each case can resemble a specific building design with slots for building height, location, soil type, etc…

Case Description		Solution Description	
Slot/Feature	Value	Slot/Feature	Value
Project Number	142	Slurry Wall	Yes
Address	Taipei	Steel Sheet Pile	No
Excavation Depth (m)	10.5	Steel Rail Pile	No
Field Area (m^2)	301	Steel Pipe Pile	No
Working Space	Not Enough	H_Section_Steel_Pile	No
Pollution Prevention	Yes	Driven Pile	No
Neighboring Settlement	Forbidden	Auger Boring Pile	No
Groundwater (m)	3	Prepakt Mortar pile	No
Soil Type	Sandy Clay	Retaining Column	No
Soil Strength Soft	No	Full Casing Pile	
Soil Strength Firm	Yes	Row_Pile	

Figure 3.9. An example of the CASTLE case base (adapted from [Yau and Yang 1998])

The slots' importance weights can be assigned manually by the user or by various algorithms. In CASTLE the slots' relative importance weights are generated using the Gradient Descent Method (GDM). In the GDM random cases are selected from the case base as the target cases, and the cases in the case base that are most similar to them are found based on a set of initial slot weights. These initial weights are modified according to how well the slots' values are matched. After examining several random cases, the resulting weight update vector (made up of weights for each slot) is normalized, scaled by a factor (called the delta factor) and added to the current weight vector. The delta factor is decreased by multiplying it by a defined step size update. The process continues until the delta factor reaches a defined lower limit.

The CASTLE system is able to retrieve from the case history the case that best matches the problem at hand. Although the CASTLE currently has 254 cases, the author mentions that as the system is used, more cases will be added to the case base. This will provide more reliable results but also makes the selection process more complicated.

Another case based system is DEMEX [Garza and Maher 1996], an interactive case-based tool for the domain of structural design of buildings. The purpose of DEMEX is to help designers use prior knowledge to solve new design problems by using this knowledge to improve the user's

understanding of the problems. In order to do this, the system automates several case-based (memory-based) processes, performing large search processes and identifying potentially critical and additional information, but allows the user to guide and direct the retrieval of relevant experiences.

The cases are represented in DEMEX as design prototypes as opposed to slots in CASTLE. Gero was the first to introduce design prototypes [Rosenman et al. 1994] as means of representing design solutions that can be reused. Design prototypes usually have three components, function, behavior and structure. For example,

Function

Support-building-type *office*
Support-grid-geometry *rectangular*

Behavior

Net-area-useable-space *15000 m^2*

Structure

Overall-height *70 m*
Location-of-core *eccentric*
Floor-system-type *beam-slab*

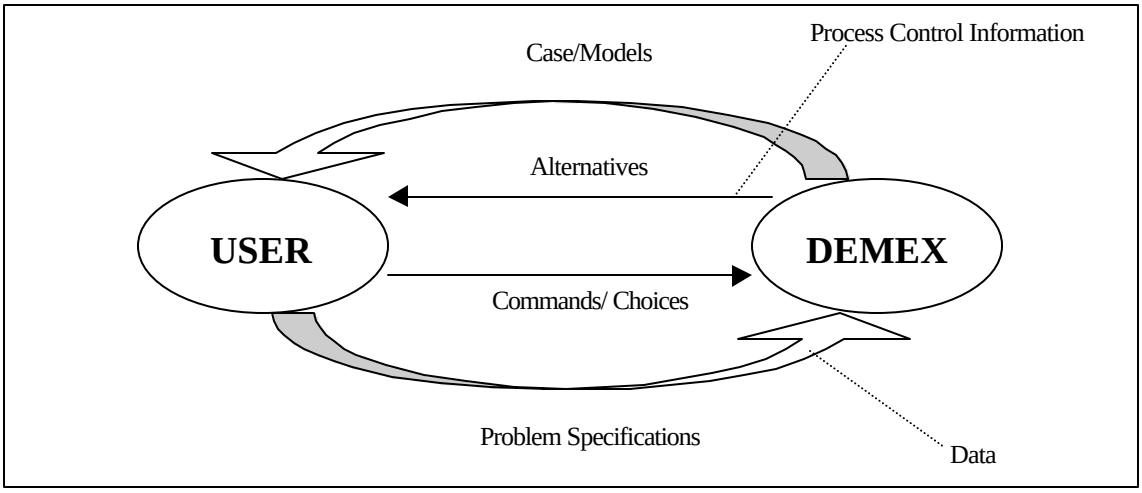


Figure 3.10. The DEMEX System [Garza and Maher 1996]

DEMEX employs retrieval algorithms to supply the user with similar cases in the case base [Garza and Maher 1996]. The user issues commands for retrieval and provides a problem specification (similar to CASTLE). DEMEX then provides alternative solutions from the case base (Figure 3.10.).

Case based reasoning is a useful approach for representing and retrieving previous design case histories. A drawback of this approach is that the retrieved case depends on the way similarity was formulated i.e. (the weights W_i). Different W_is will result in different retrieved cases. Also the choice of slots is very important. One must choose slots that actually influence the selection of the case and not idol slots. Generally, case-based reasoning does not allow for explicit definition of performance criteria.

Also, similar to neural networks, case-based reasoning does not allow for the definition of explicit rules on assembly selections, which is important for automated assembly selection. These kinds or rules can be defined in expert systems. In the next section the third approach studied for selecting building assemblies, expert systems, will be discussed.

3.1.3. Expert Systems

Expert systems (sometimes called knowledge-based systems) simulate the behavior of an expert in some narrow discipline [Hopgood 1993]. In expert systems the knowledge is stored separately from the reasoning strategy, often in the form of simple if/then rules, (IF <condition> THEN <conclusion>). These rules can be readily changed, which means that expert systems are easier to maintain than other kinds of computer programs.

There are two main reasoning mechanisms, forward and backward chaining, depending on the type of the problem (Figure 3.11. 3.12.). Forward chaining is the name given to a data-driven strategy, i.e. rules are selected and applied in response to current fact base. The fact base comprises all facts known by the system, whether derived by rules or supplied directly. For example we might have rules that say[12],

IF room function ?X is library THEN required floor assembly fire rating ?X is 3h.

[12] The ?X in the rules represent a variable.

IF require assembly fire rating ?X is 3h AND Construction Type ?X is concrete flat plate THEN assembly used ?X concrete_slab_and_carpet.

Now if we present this expert system with the following facts;

Room function is library

Construction Type is concrete flat plate

We would expect the following conclusion: *assembly used is concrete_slab_and_carpet.*

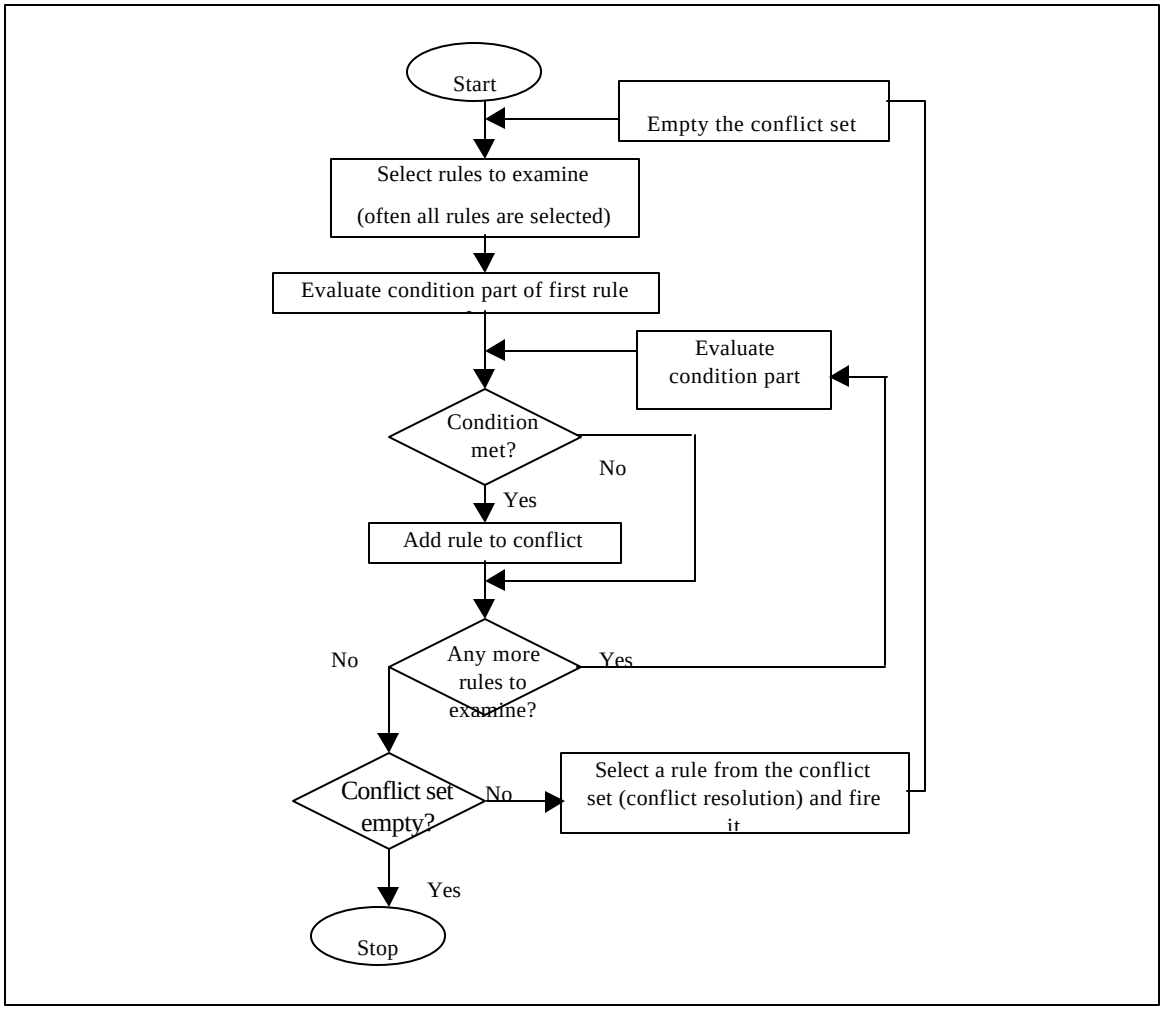


Figure 3.11. Forward Chaining (adapted from [Hopgood 1993])

The cyclic selection, examination and firing of rules in forward chaining are shown in Figure 3.11. Several variations to this cycle are possible.

Backward chaining assumes the existence of a goal that needs to be established or refuted. For example, in the previous example our goal might be to establish the fire rating of the assembly and we mat not be interested in any other deductions the system is capable of making. The process of backward chaining is shown in Figure 3.12.

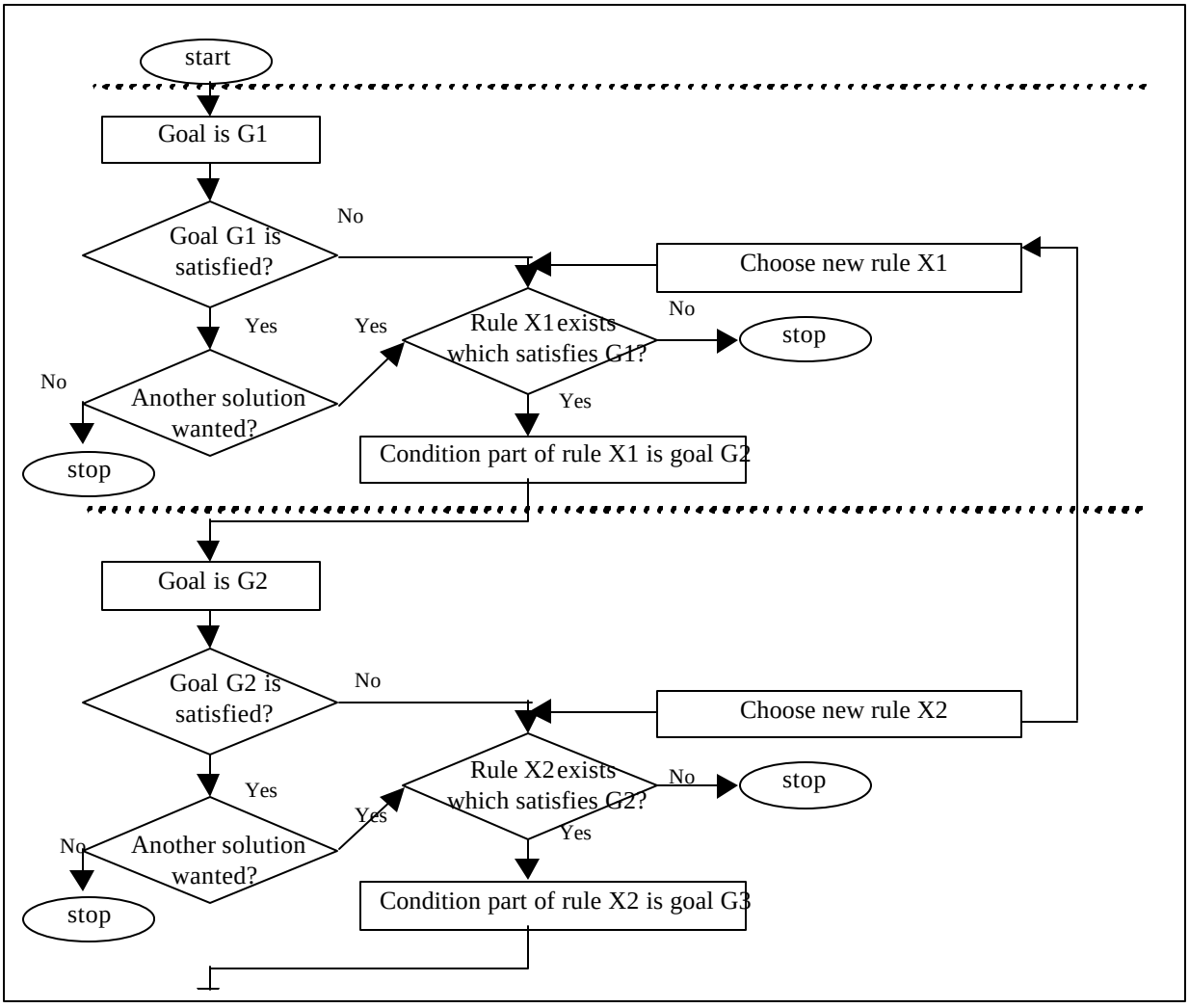


Figure 3.12. Backward Chaining (adapted from [Hopgood 1993])

Therefore we can see that expert systems attempt to represent domain knowledge within a computer system. Rules are good at representing casual knowledge, but there are many other types of knowledge: object knowledge, intuitive knowledge and so on. Representing knowledge about the shape and properties of a wall assembly, for example, would be tedious, if not

impossible, using if/then statements. A much better way is to represent objects as a series of attributes and methods, in object oriented terminology. In knowledge base terminology, we use frames. Expert systems with rules and frames have a strong expressive power, can deal with higher order constructs and inheritance structures (like classes).

Few expert systems for design and/or selection of particular assemblies or components have been developed. The developed systems tend to target a narrow domain of building design. Cornick [Cornick et al. 1987] describes two prototype expert systems, one dealing with the external building envelop and the other with the internal space division of buildings. BERT (The Brickwork ExpeRT) investigates the extent of integration between a knowledge base and a computer-aided drafting system. Knowledge about the movements of joints within brickwork cladding and the related spatial information was extracted from a human expert and built into BERT. The system, given a geometric description of a wall, remarks on the movement joint layout proposal and suggests possible improvements.

Another system is GERT (A Gypsum Metal Stud Partition Expert). GERT supports two major processes; component selection and an automated component designer. Given a specific requirement GERT would return the most appropriate product. Selection is mostly database retrieval but the factors that govern the cost of the components are rule based. Also GERT has the ability to generate a partition design based on specific rules.

WOODPRO is a lumber database and wood selection expert system. As a database WOODPRO contains 331 common types of wood and their properties. Based on a set of questions provided by the user about the design case, WOODPRO makes a selection of the most suitable type of wood to use. The selection is based on expert rules.

Sealant failure is one of the major causes of post-construction costs in new building projects. Incorrect joint design, incorrect product selection, or application deficiencies may cause sealant failure. SEALANT is an expert system that selects and designs sealant, based on the location, the type of joint, and the materials being joined. The joint is immediately sized, allowing for construction error, temperature and moisture movement and other deflections. SEALANT will also look up material movement characteristics and perform movement calculations.

Expert Systems used in construction can capture domain specific rules about assemblies. Although frame based expert systems can represent object knowledge, expert systems only allow the representation of declarative knowledge only. Procedural and algorithmic knowledge is difficult to represent in expert systems. It is difficult to represent an assembly selection procedure using expert system rules only.

Now that we have described the three research directions on assembly selection we will discuss assembly generation. The assembly generation research direction, described in the next section, tries to capture other types of rules (geometric rules) about building assemblies.

3.2. ASSEMBLY GENERATION

In the seventies Habraken [Gross 1996] described a design method where the various members of the design team responsible for the different assemblies/components, agree to work within certain position rules (position constraints) and, avoid interference conflicts that can be costly. By agreeing to locate each assembly/component in its own grid band or its own grid line, interference conflicts can be minimized and reduced to predictable occurrences where bands or lines cross. Then a stored library of predefined conflict resolution routines can be provided, indexed on the grid band intersection.

This design method requires constraint-based drawing editors (e.g. CoDraw, SKETCHPAD and Visio). These editors allow the designer to apply geometric constraints to drawing elements; then as the designer edits the drawing, the computer enforces the constraints. For example, once the designer establishes a relation among two elements (e.g. proportions of two edges or adjacency), the computer tries to keep the elements arranged to maintain the relation. By contrast, in conventional drawing editors the designer must keep track of all desired spatial relationships. Constraint based editors enable designers to instruct the computer about the desired arrangement of elements.

A few systems took constraint-based editors a step further to try to implement Habraken's design method [Gross 1996]. In these systems a *system designer* (this is the first type of user) programs

each component with a set of possible constraints that describe how it can be placed on the site and in conjunction with other components. For example, a beam with a set of alternative spatial relation constraints that describe how it may be placed in the site and assembled with columns. When the *end designer* (this is a second user that woks within the rules specified by the system designer), places a beam into the design, it may take only certain positions relative to other assembles. The positioning behavior of assemblies/components in these systems, is similar to 'snap-dragging' (dragged drawing elements snap to align edges and centers of the elements in the drawing) in familiar computer drafting packages. However, in such systems preferred positions are customized for the components rather than using generic edged and center alignment. The preferred positions correspond to assembly rules that the system designer has defined. Also, in such systems, once two components are joined their assembly condition is constrained until the end designer takes them apart.

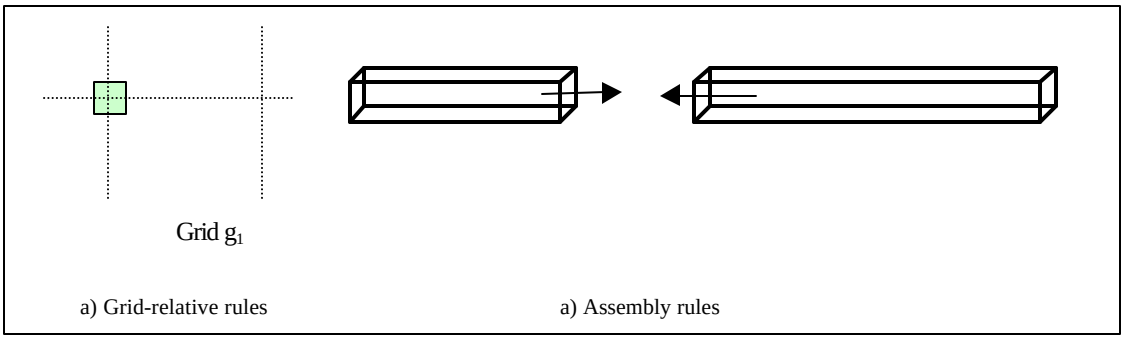


Figure 3.13. Grid-relative and assembly rules

There are many ways to express rules governing the positions of components, but two methods are common in architectural design [Gross 1996]:

1) Grid-relative rules; the placement of assemblies/components relative to a grid or a system of grids (Figure 3.13.a). For example the grid rule that centers a column on grid crossings converts raw component coordinate using a function based on the grid spacing in grid g_1;

$$\{x_{actual,}\, y_{actual}\} = \left(\mathrm{g-rule}_{\text{ crossing }}\, g_1\{x_{raw,}\, y_{raw}\}\right)$$

2) Assembly rules; the placement of assemblies/components relative to one another (Figure 3.13.b). For example, the rule for two components c_1 and c_2 assembled with abutting left and right sides comprises two complementary functions for computing the coordinates of c_1 and c_2;

$$c_1 : \{x_{actual,}\, y_{actual}\} = \left(\text{a} - \text{rule}_{\text{right - abut}}\ c_2\{x_{raw,}\, y_{raw}\}\right)$$

$$c_2 : \{x_{actual,}\, y_{actual}\} = \left(\text{a} - \text{rule}_{\text{left - abut}}\ c_1\{x_{raw,}\, y_{raw}\}\right)$$

The various functions that modify a component's position are composed, with grid-relative rules operating first, followed by assembly rules. A detailed description of these functions can be found in G. Kramer. Next we describe the two building design constraint-based systems in the literature, CKB [Gross 1996] and SEED [Flemming et al. 1993], [Flemming et al. 1992; Flemming et al. 1994].

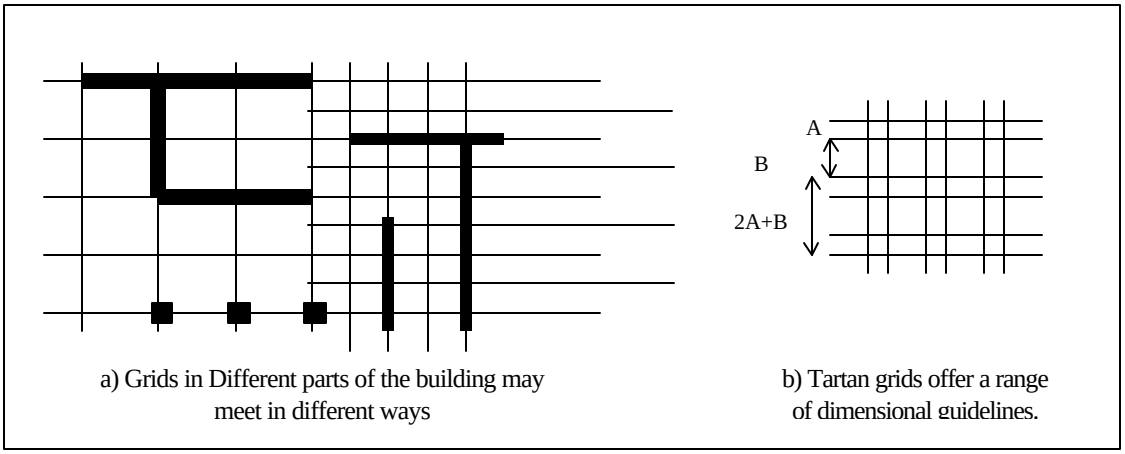


Figure 3.14. Grids in CKB

3.2.1. Construction Kit Builder

Gross [Gross 1996] explored the idea of making CAD editors more like lego, by building in to the editors knowledge about the layout and the assembly of building components. He developed the CKB (Construction Kit Builder) system. The CKB allows system designers to specify grid-relative rules like center-on-grid-crossing (e.g. for square columns), center-on-horizontal-lines, center-on-vertical-lines (e.g. for wall panels), center-on-grid-squares (e.g. for lighting fixtures).

Different kinds of grids can be defined (Figure 3.14.), like rectangular, tartan and non-rectangular grids. Also assembly rules can be defined to relate how two components fit together. Therefore, there are 4 classes in CKB, component, grid, assembly rule and grid relative rule, Table 3.1. CKB is a flat system; it does not allow for working with an assembly of parts. Instead only components or single assemblies are supported.

Component	Grid	Assembly Rule	Grid Relative Rule
Reference point	Horizontal spacing	Component-1	Horizontal type
Grid position rule	Vertical Spacing	Component-2	Vertical type
Grid	Horizontal offset	Horizontal relative position	Combination
Assembly rules	Vertical offset	Vertical relative position	Horizontal bands
Applicable Assembly rules			Vertical bands

Table 3.1. Key slots of the 4 classes in CKB

Systems similar to CKB have been developed as research projects. The Harness program for hospital design employs automatic positioning of structural members according to a modular system. MATURA [MATURA] is a system for dwelling in-fill and provides several examples of the uses of grids and position rules. The A4 [Gross 1996] system addresses similar issues to CKB especially spatial coordination of mechanical systems. The OXSYS [Hoskins 1973] system of the seventies produced details for the oxford construction method.

These systems operate with a predefined building system hard-coded in the program, rather than allowing the designer to specify the rules for locating the assemblies/components. CADD systems today, do not support assembly/component coordination in any straightforward manner. Furthermore, there are conceptual limits to the constraint-based approach such as limits on creativity and design flexibility. This makes it suited for parametric and innovative types of design described in chapter 2.

3.2.2. SEED

SEED is an acronym for Software Environment to support the Early phases in building Design [Flemming, Coyne et al. 1993]. The overall architecture of SEED is based on a division of the preliminary design process into phases, each of which addresses a specific task. SEED intends to support each phase by an individual support module based on a shared logic and architecture. The modules envisioned for the first SEED prototype support the following phases: architectural programming, schematic layout design, and schematic configuration design.

1.Architectural Programming. A problem is posed in this module by a statement of the client's overall goals, a description of the site, etc. The module supports the generation of an architectural program for a building that realizes the client's goal; this is a solution in the context of the module. A case that could be reused and adapted would be a program that has been developed in the past for a similar building type and site.

2.Schematic Layout Design. In this module, a problem is posed by an architectural program, and the solution generated is a schematic layout of the functional components of the program. A case would be a layout that has been developed in the past for a similar architectural program.

3.Schematic Configuration Design. The problem is posed by a schematic layout, and a solution develops this layout into a 3-dimensional configuration of spaces and building components.

A more detailed functional specification of the first three SEED modules from a high-level user perspective can be found in [Flemming, Baykan et al. 1992]. Here, we are concerned with the constraints in SEED. In developing SEED, constraints on design elements (e.g. spaces) are represented by objects whose class indicates the nature of the constraint and whose attributes give specific data. For example, a width constraint is represented by an object that instantiates class WidthConstraint and has two main attributes, LowBound and HighBound. The constraint classes that are currently implemented or under development are Single-Unit Constraints and Multiple-Unit constraints. Single-Unit Constraint constrain a single unit of the building in relation to itself, e.g width, height, etc… Multiple-Unit constraints constrain two or more units of the building together as can be seen in table 3.2.

Constraints that restrict non-geometric attributes like the illuminance level of a room are currently not implemented in SEED [Flemming, Baykan et al. 1992]. But note that an evaluation of this

constraint would have to take not only the attributes of the room for which it is specified into account, but also the attributes of the surrounding walls and windows. In SEED terminology [Flemming, Coyne et al. 1993], it would nevertheless be called a "single-unit-constraint" because its formulation involves the attribute of a single design unit; it would indeed be impossible to specify up-front which other design units are involved in satisfying this constraint because these units change with the design. The units involved have to be found when the requirement is being evaluated for a given design.

Single-Unit Constraints	**Multi-Unit Constraints**
These are constraints on the attributes of a single design unit. When geometric attributes are the subject of such a constraint, it typically specifies a lower or an upper bound, like a minimum and maximum area for a room. Making an upper bound equal to a lower bound can specify exact values. The attribute under consideration may be explicitly given in a design unit or may have to be derived from other attributes.	These are constraints whose formulations involve more than one and typically two design units.
1. WidthConstraint	*1. AdjacencyConstraint* This constraint specifies a required adjacency between two design units. It is interpreted in a strict geometric sense, namely that the two units have a shared boundary of a minimum length, called overlap.
2. AreaConstraint	*2. Direction Constriant* This constraint specifies that a design-unit must be in one of several desired directions from another design-unit. In combination with an adjacency-constraint, it can be used to specify a required adjacency in specific directions.
3. HeightConstraint	*3. Distance Constraint* This constraint specifies a minimum or maximum distance between two design-units.
4. AspectConstraint The lower and upper bound of an aspect constraint are related by the following	*4. Access Constraint* This constraint specifies that a design unit must be accessible from another design unit

equation: AspectConstraint.LowBound = 1/AspectConstraint.HighBound, where 1 <= AspectConstraint.HighBound. Width and aspect constraints have clear interpretations for rectangular design units. Their meaning for design units with other shapes has yet to be established.	without necessarily being adjacent to it. Its evaluation requires the establishment of a clear path with a minimum width at each point between allocated design units or through design units that are public areas. *5. Exterior Access Constraint* This constraint specifies that a design unit must border the exterior of the overall enclosing rectangle. Specific directions can be ruled out as satisfying this constraint. This does not mean that an adjacency in a ruled-out direction is forbidden. For example, the administrative wing of a firestation must border the external rectangle from the main access direction, which does not mean that it cannot border the exterior also from other directions. Forbidden adjacencies are currently not implemented! *6. Outside Access Constraint* This constraint differs from ExteriorAccessConstraint in that it is satisfied not only by access to the enclosing rectangle, but also by an adjacency to any other outdoor space. Specific directions can again be ruled out as satisfying this constraint.

Table 3.2. Constraints in SEED

Currently in SEED, these constraints are used to define relationships between components in building enclosures[13]. The types of building enclosures are shown in Figure 3.15. Geometric knowledge about building design can be added by defining the different constraints between the components of an enclosure.

[13] Enclosures in SEED are similar to assemblies but also include connections.

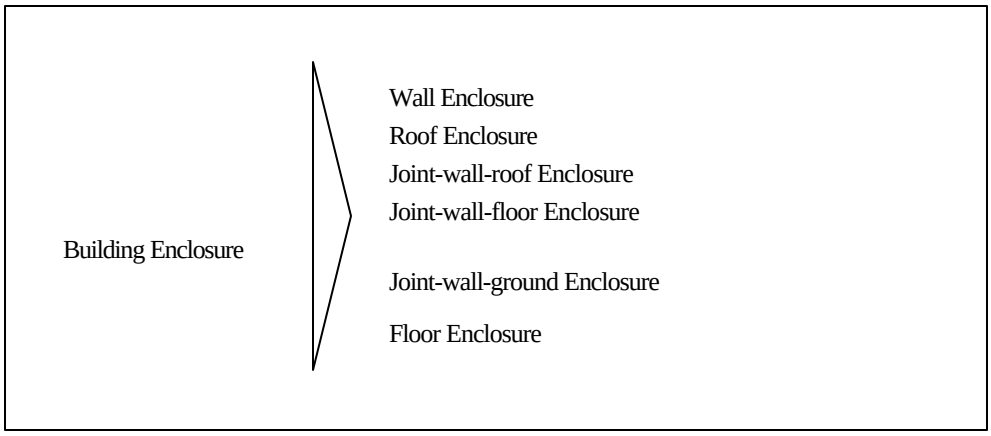


Figure 3.15. Building enclosures in SEED

Constraint based editors allow capturing geometric knowledge about the assembly. However these editors lack the linkage between the geometric constraints and other design constraints. For example the height of wall may be constrained by the local building code or the grid spacing (in the CKB system) can be constrained by the sizes of the structural elements. Most of the existing constraint based systems do not take this linkage into consideration.

3.3. SYNOPSIS

This chapter presented a review of the research carried out in support for automated assembly selection and generation.. In specific it was found that three research directions can be identified for automatic assembly selection: neural networks, case-base reasoning and, expert systems. Assembly generation research is concentrated on constraint based editors. The two constraint based editors in the literature were reviewed: CKB and SEED.

Neural networks were found to be good for discovering design emergence and generation of new design prototypes. Neural networks can be useful when the goal is to generate new assembly types. However, the output of a neural network is not exact and changes from a design case to another. It is difficult to add exact knowledge like fixed rules in neural networks. Case based reasoning can be used to find cases from a case history that most resemble the current design situation. It is assumed that the cases are all correct and the output is heavily dependent on the case base formulation. Similar to neural networks, case based reasoning can not be used to

represent exact knowledge like fixed rules. Therefore expert systems were reviewed. It was found that although frame based expert systems can represent object knowledge, expert systems allow the representation of declarative knowledge only. Procedural and algorithmic knowledge are difficult to represent in expert systems.

Next Constraint-based editors were reviewed for assembly generation. It was found that constraint based editors allow capturing geometric knowledge about the assembly. However these editors lack the linkage between the geometric constraints and other design constraints.

All of the described research directions had to be based on one kind of a building model or another (e.g. a frame-based model in expert systems, or a slot-filler model in case based systems). It is important to review building models and their importance. In the next chapter we will present the information modeling and building product models.

CHAPTER 4. INFORMATION MODELING AND BUILDING PRODUCT MODELS

In this chapter we introduce information modeling and building product models. This chapter provides the findings of the review carried out of current information and product modeling efforts in the architecture engineering and construction domain. The objectives of building product modeling are described. The different information modeling techniques used in building modeling are identified. The various building models are described according to the information modeling technique they utilized. Several examples are given.

In order to automate the assembly construction selection for the various assembly types in a building design we must have a database to store two types of information: firstly, the various assembly constructions and secondly, the actual building design and its related information. This database should store the available assembly construction options, and also their properties, criteria and constraints on their use. Also in order to be able to define the different constraints, and the selection procedure, the actual building design should be stored also in the database. The design of this database is dependent on how the various entities (e.g. the building, walls, assembly constructions, material components etc…) and their relationships are modeled. In order to develop this database the building must first be modeled by identifying the various entities and the relationships between them. In other words a building product model must first be developed using an information modeling technique. After the building is modeled the various constraints and the assembly selection and generation procedure can then be developed. This methodology was shown in Figure 1.1.

The developed building product model will also facilitate the data exchange with other tools as will be described later in this chapter. We start this chapter by a brief description of information and product modeling and then we identify the uses of the building product model. Next we start reviewing several information modeling techniques and give examples of some building product models developed using each technique.

4.1. INTRODUCTION

Much disagreement still exists about what a product model actually is and there are as many definitions of a product model as there are researchers working in that area [Miles and Moore 1994]. However, in this section we will try to present our own findings based on a review of the different kinds of product models in the literature, their use and motivation and, their modeling techniques. Product models are certain kind of models[14] called information models. Information Models (IM) are a description of 'something'. Formally, an IM can be defined as follows:

> *An information model is a formal description of types of ideas, facts and processes which together form a model of a portion of interest of the real world which provides an explicit set of interpretation rules [Schenck and Wilson 1994].*

Information models were originally used for database design (which is one of the uses of the product model in this dissertation), but have now proved useful for other tasks like knowledge-base design, representation of expert knowledge and product models. An information model that represents a physical object (a product), as opposed to abstract ideas, is called a product model. Product modeling is a technique to describe a product's data so that the data can be stored in the computer. Product data on the other hand, is the data that describes the function and physical characteristics of each unit of the product from its requirements at inception to its configuration at time of retirement [Leeuwen et al. 1996]. A definition of a product model is; *A product model is a structure for product data that a computer can understand*. This structure is similar for all instances of the product.

---

[14] Models facilitate understanding of objects through the description of salient features, e.g. drawings, cardboard models, mathematical models or, information models.

Therefore, a product model is basically a classification and breakdown of a product into classes or objects (sometimes called frames, entities, schema, or units), and a description of the different relationships between them. The objects are used to represent real world objects and concepts in the computer.

4.2. UESES OF BUILDING PRODUCT MODELS

So far we have said that product models are used to represent and store product data in the computer. The representation of product data in the computer can then be used for different purposes. We have found that building product models in the AEC industry have been used for two main purposes;

4.2.1. Developing Design Support Software and Databases

Product models are used to develop different design support software tools and also to develop databases to support design. An information modeling task is usually carried out before any new software tool is coded or any significant database is designed. One of the outputs of this information modeling task is the data structures to be used in the software, or the tables and relations in the database. These data structures can range from simple data structures (like linked lists) to classes and objects to a database design (which can be the different tables and relations in a relational database or objects in an object-oriented database), as can be seen in Figure 4.1. Figure 4.1. shows part of building product model developed in this dissertation. As shown in Figure 4.1. this product model can be implemented as a relational database and also as object oriented classes. More often, product modeling is widely used for developing data structures for databases. The kind of data structures developed depends on the information modeling method and the data model used. In AEC related software, the data structures usually represent the building and its variables. The excerpt of the product model is shown in Figure 4.1. The figure shows the formal model which is described using the EXPRESS modeling technique (which will be described later).

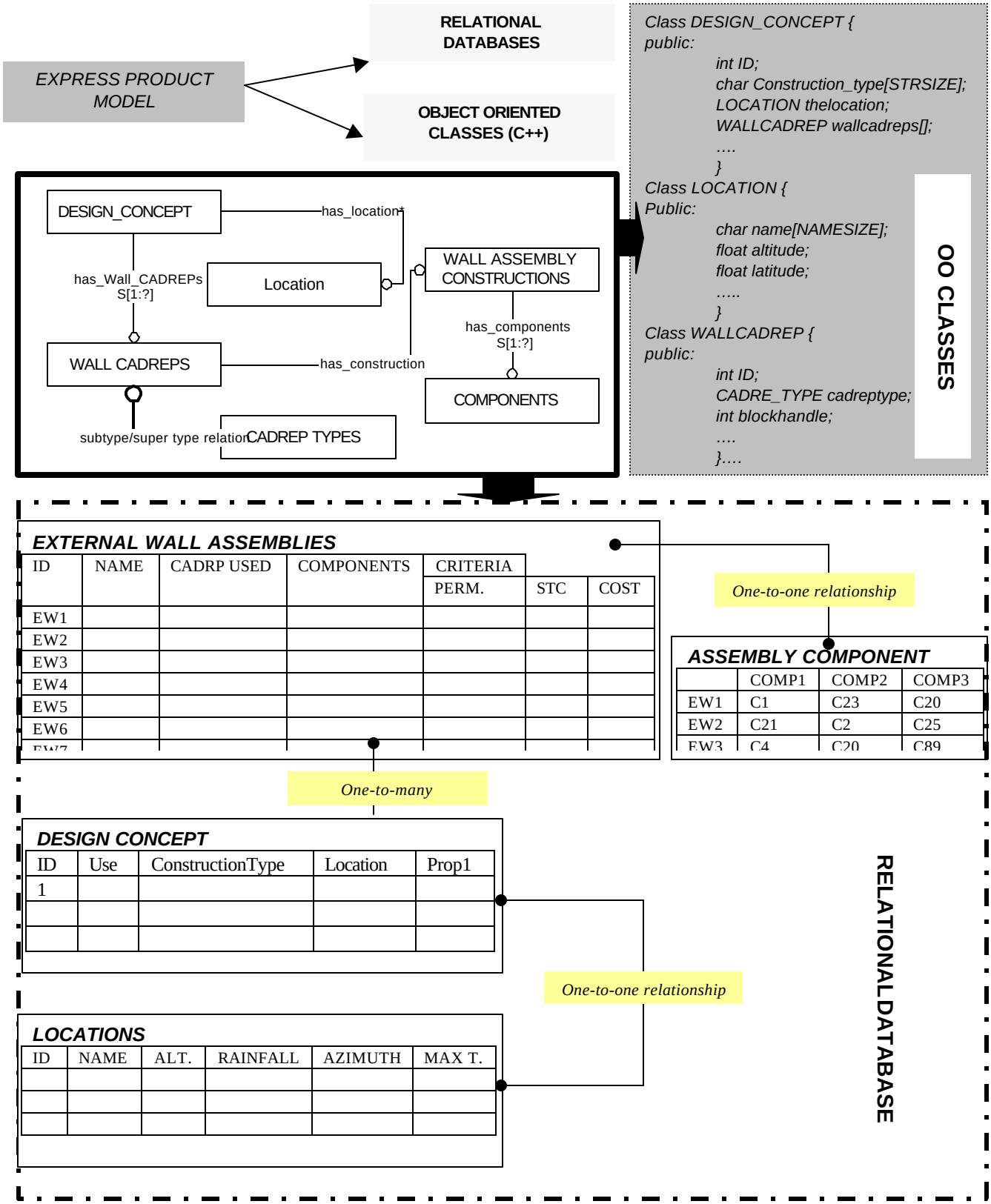


Figure 4.1. Use of the Product Model

It is important to note that the implementation (or compilation in information modeling language) of the formal product model (defined here in EXPRESS) as a database (or any other information base) can be automated (although in this dissertation the implementation was carried out manually). In section 4.2. we will review different formal information modeling techniques (which can be used to develop formal product models to be implemented as databases) and examples of building product models developed using them, but now let us review the second use of product models, data exchange.

4.2.2. Data Exchange

Product models are also most commonly used to facilitate data exchange between different software. The different building design support software needs to have their own representations of the building. For example, energy analysis software (like DOE or BEANS) need to represent the building as thermal zones, enclosures and equipment. On the other hand structural analysis software (like ANSYS or RISA) need to represent the building as structural elements, supports and loads. Due to these various representations communication between these various software becomes difficult. Most importantly, the lack of conventions in the structuring of data within CAD systems inhibits general interfaces and use of multiple software on a single CAD data file [Eastman 1989].

If the integration of the different software is made easier, many more software would be developed and integrated in a design system that can be used for architects and engineers. In an integrated system these representations need a way of transferring information back and forth between each other. We have found that are basically two techniques for accomplishing the integration between various software using product models [Nassar 1998a]: the traditional approach and the alternative approach.

The first approach is to develop a universal product model. Instead of developing product models for each of the software, product models have been used to define a unified representation of the building that facilitates data exchange. We call this the *traditional approach* (this has also been called the universal approach [Assal 1996]). In the traditional approach, all the building components and possible relationships between components are enumerated [Turner 1992]. The

second approach is to develop product models for each software and provide for means for transferring information from one software to another. We call this the *alternative approach*.

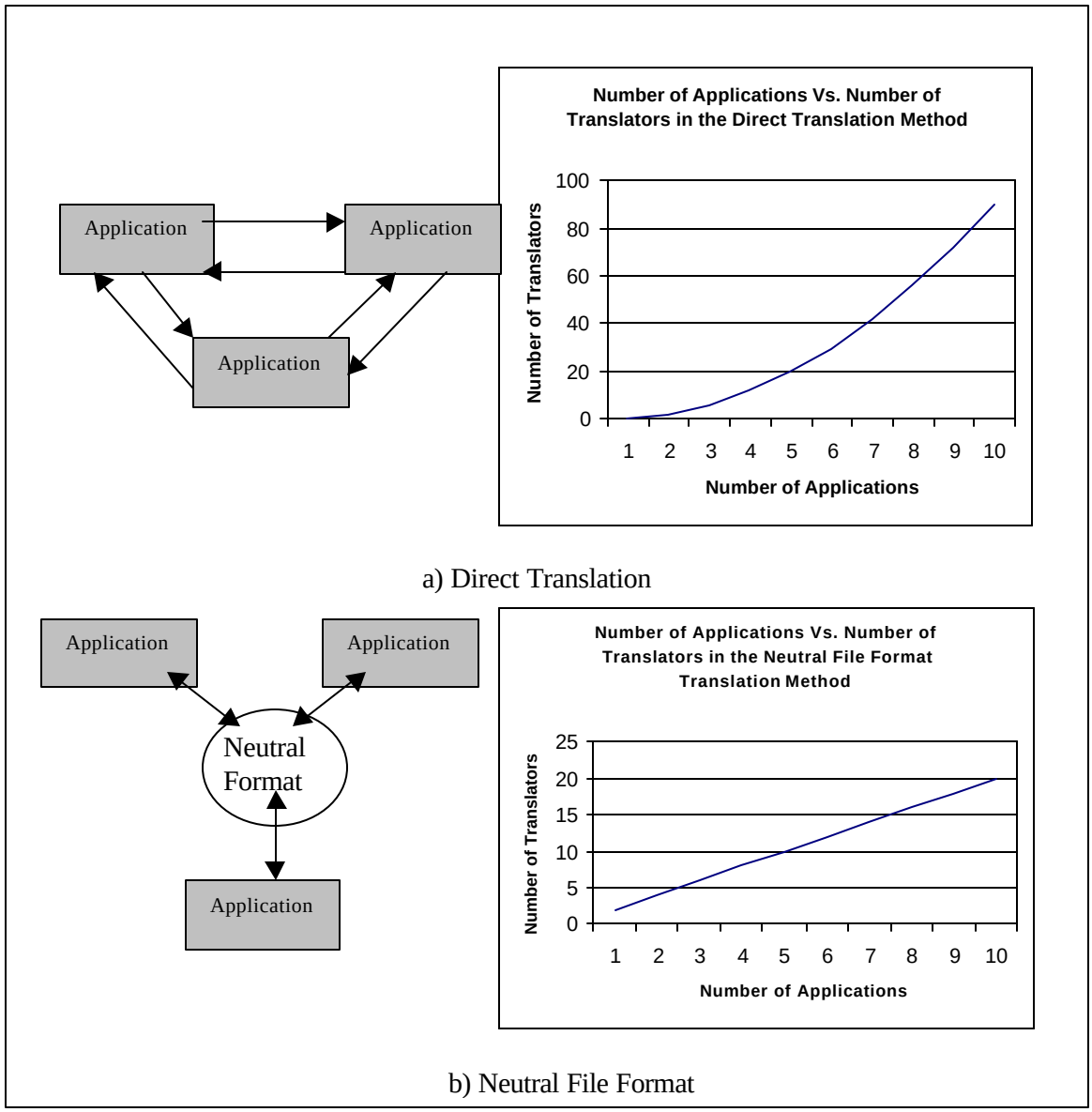


Figure 4.2. Translation Approaches

Several researchers have criticized the traditional approach as being a rigid solution [Jeng et al. 1997], [Assal 1996] and [Turner 1992]. The traditional approach requires that all objects and relationships be predefined in the product model. New methods and software emerge everyday, which require different representations that are not in the universal product model. The new

developed software most probably will require different formats for their internal operations. Due to this the traditional approach is considered to be limiting.

The second approach employs translation between the product model of one software and the other. This alternative approach can be accomplished by one of three methods: direct translation between two different software, the use of a neutral file format or by generating different database views.

In the direct translation method, translators have to be developed for each software developed[15]. As new software are developed new translators are required to exchange data with the other software already existing. If *n* is the number of applications that need to exchange data among them, then *n(n-1)* translators are required [Owen 1993]. As the number of applications increase, the number of translators required grows exponentially, as shown in Figure 4.2.

Figure 4.2. also shows the neutral file format method. In this method the software developers provide a way to write out the internal representation of their software in a neutral format. This neutral format can then be read by another software and the data structures of the neutral file format can be mapped to the software's internal data structures. An important advantage of this method, it that the number of translators required in this method is less than the direct pair-wise translation method. The number of translators required in this method is equal to *2n*, as shown in Figure 4.2. Examples of neutral file formats are Data eXchange File (DXF), Initial Graphics Exchange Specifications (IGES) and STEP AP203. For example, the DXF file format has become the de facto neutral file format for graphical applications. Figure 4.3. shows an excerpt of a DXF file for the shown drawing. This textual file can then be converted to a software-specific format, like AutoCad's DWG format.

[15] The difference between the neutral file format method and the traditional approach is that the neutral format is not the actual internal data structures of the software as in the traditional format [Assal 1996].

Another example of neutral file formats in construction is SDEF. The Standard Data Exchange Format is a non-proprietary format for the exchange of Critical Path Method (CPM) scheduling data between project team members. The SDEF has been implemented by the U.S. Army, Corps of Engineers and the major construction scheduling software vendors. A software (CHECKER) is developed to validate the syntax of the data structures produced by the various scheduling software vendors. The SDEF therefore, facilitates data exchange between various scheduling software (e.g. Primavera Project Planner™ and Microsoft Project™).

Currently there are two major research efforts being carried out to define a neutral file format for the building industry: the STEP (**S**tandard for **E**xchange of **P**roduct Model Data) initiative and the IAI's (**I**nternational **A**lliance for **I**nteroperability) IFC (Industry Foundation Classes). The IAI is developing standard neutral classes (IFCs) for application developers to use to help in the exchange of product data.

The STEP initiative on the other hand is currently working on a neutral file format for the AEC industry. Also, besides providing a neutral file format, has developed an information modeling technique that allows different application developers and researchers to develop product models. This information modeling technique uses a language called EXPRESS. By using EXPRESS product model developers can guarantee easier product data exchange, since mapping between the product model and the neutral file format becomes easier. The EXPRESS approach is used in this research in order to facilitate data exchange.

A third method used in the alternative approach is database view generation. Current database management systems (DBMS) can produce different views of the same data based on specific queries. For example, Figure 4.4. shows two views of the same data generated by two different queries on a commercial DBMS (Microsoft Access). The problems and suggested solutions with this approach has been discussed in detail by [Assal 1992], [Assal et al. 1995], and [Eastman et al. 1995].

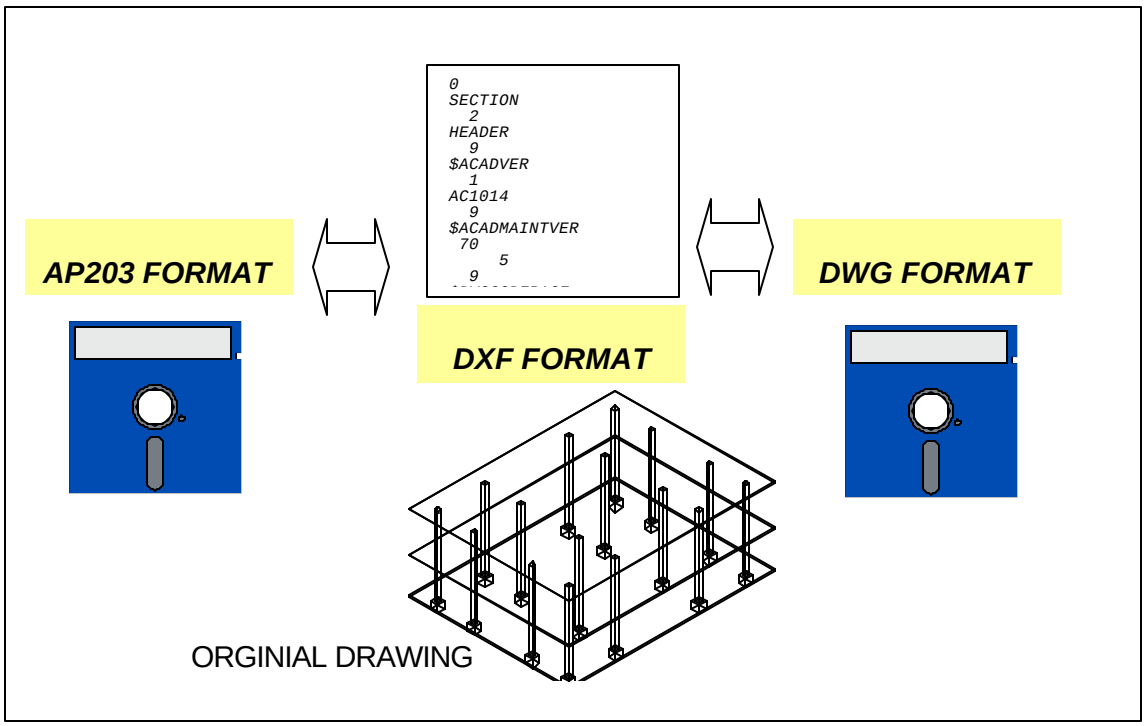


Figure 4.3. Neutral File Format Example

It is also important to mention that building product models have also been used for capturing Design Intent. Building product models have been used to represent the reasoning behind design [Garza et al. 1997]. Alcantra developed [Alcantra 1996] a product model that captures design intent during the design stage of the facility. The product model developed organizes design rationale data into a useful information source for designers, value engineers, cost estimators and, construction schedulers. Product models also provide an understanding of how the different parts of the building relate to each other, the processes and information required in AEC.

Whatever the use of the product model, the definition of product models that represent the building has proved to be a challenging task [Eastman 1994]. Buildings as a product are more complex and harder to model than a mechanical part or a circuit board as products. A building product model that supports design is challenging for many reasons. Designers think in terms of various objects.

One designer thinks in terms of space and activities, while another thinks in terms of solids and voids, etc…(the same designer can think in terms of different objects at various levels of design). Therefore, no one representation or breakdown structure can capture the way designers think and therefore a special product model that supports selection of assemblies is required.

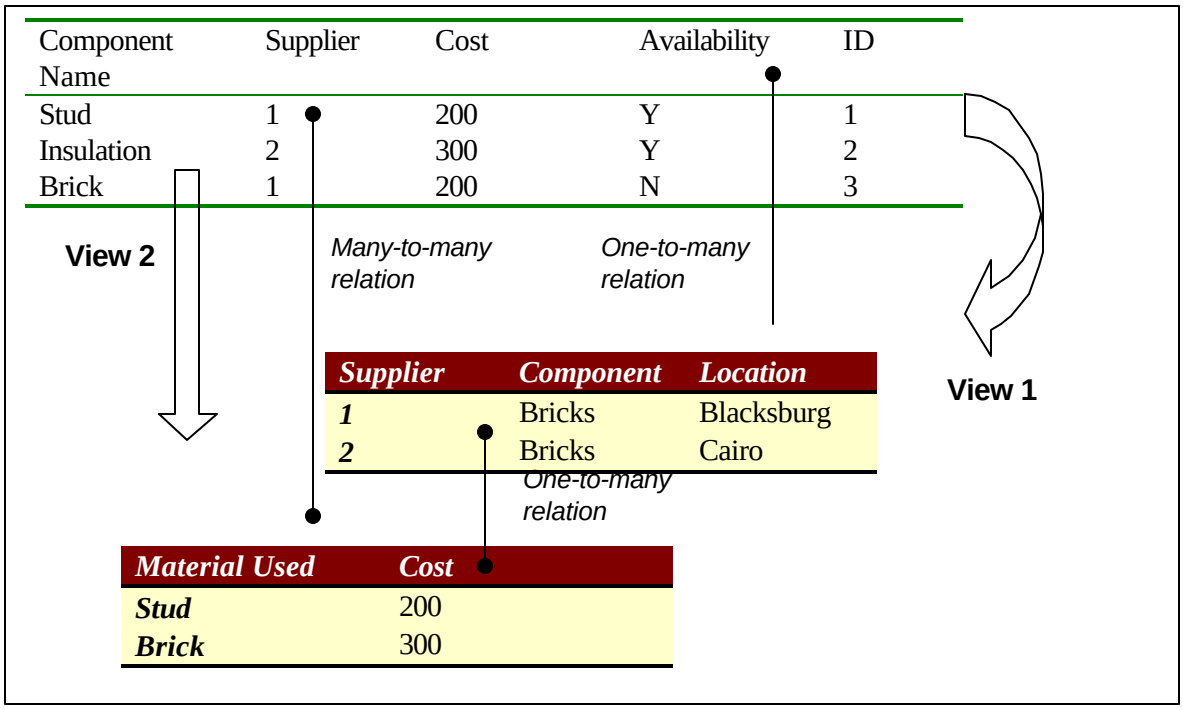


Figure 4.4. Database view generation

In the next section we will start by presenting different information modeling techniques that can be used to develop building product models. As we already said, the product model is required to develop the database that supports assembly selection (which will be described in the next chapter) and to facilitate data exchange. We will review six information modeling techniques used in AEC: NAIM, EXPRESS, IDEFx1, Feature-Based Modeling, Engineering Data Model (EDM) and Object Oriented Analysis. Also examples of different product models will be provided.

4.3. REVIEW OF MODELING TECHNIQUES AND BUILDING PRODUCT MODELS

The building product model developed in this dissertation (described in the next chapter) is used in two ways. Firstly, to define the objects (data structures) that could support the selection and generation of building assemblies. Secondly, to allow for data exchange with other software required in the selection and generation procedures. In this dissertation we adopt the neutral file method of the alternative approach for data exchange. The STEP initiative is suggested for providing the neutral file format to be used in the data exchange. This is discussed in more detail in the next chapter.

Therefore a review of the existing product models is required before developing the product model used in this dissertation. The difference between different building product models is how the models breakdown the building (i.e. the classes and relationships), the purpose of developing the model and, the modeling technique used. In the next sections we will present a review of the modeling techniques used in developing product models. Existing product models are also presented.

4.3.1. NIAM

NIAM is an acronym for "Nijssen's Information Analysis Methodology" after G. M. Nijssen, who was one of several people involved in the development of the method. More recently, since, it has been generalized to "Natural language Information Analysis Method". A more general name, "Object-role Modeling", or ORM is currently used. The philosophy behind the language is that it describes "*facts*", where a fact is a combination of entities, attributes, domains, and relationships [Hay 1997].

NIAM diagrams are a graphical notation for this modeling technique. The notation consists of symbols for objects, roles between objects, and object and role constraints [Turner 1998], [Sjogren et al. 1998]. The basic building blocks in all information modeling methods are the objects and the relations, as we will see in the next sections. Figure 4.5. shows the different types of objects used in NIAM. For example, lexical Objects (LOT) represent a set of values of an entity, such as

names and properties. A CLONE, on the other hand, is an object that occurs elsewhere on this or another NIAM model or another NIAM document (the black box and circle).

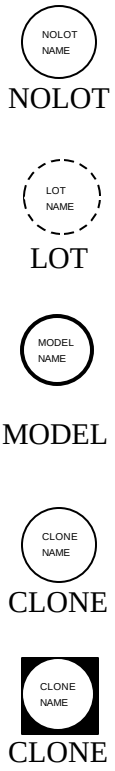


Figure 4.5. The NIAM Objects

Instead of relationships between two entities, NIAM presents the "roles" that entities, attributes and domains play in the organization's structure. Role names are read as A–R1–B and B–R2–A, or members of A "play" role R1 with members of B and members of B play role R2 with members of A [Turner 1998]. Roles act as a relation between the members of A and B:

R1: A -> B

R2: B -> A

The set of occurrences of role R1 is equal to the subset of the cartesian product of A and B for which the role A-R1-B is true. For example in Figure 4.6. A could be a *building system*, B could be a *building system component* and the role R1: *A -> B is "HAS" and R2: B -> A is "OF"*.

There are also several other constructs, besides objects and roles, in NIAM. For example there are several types of constraints. Cardinality constraints designate the quantities of objects and roles allowed in a role. Sub-types constraints are rules that restrict the division of objects into subsets. Other types of constraints are UNIQUENESS and TOTAL constraints. These constraints are used to define the semantics of the relationships between objects (e.g. one to one, many to one, etc…).

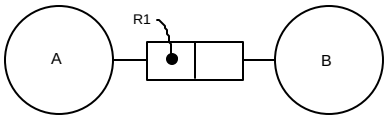


Figure 4.6. An Example of Roles in NIAM

Software exist that facilitate modeling using the NIAM method. These software can then generate data base descriptions (DDL) for a wide variety of Data Base Management Systems (DBMS). In addition, these software can generate a prototype application out of the data model for various 4th Generation Language (GL) development tools, sometime referred to as Lower CASE tools. More information on NIAM can be found in [Sjogren and Klausen 1998], [Nijssen et al. 1989].

Next we will present two examples of product models that were modeled using NIAM; the Building Systems Model and SIGMA-MOCIB.

4.3.1.1. Building Systems Model

The Building System Model was initially introduced by Turner [Turner 1990] as an ISO (International Standard Organization) effort to define a generic building reference product model. The system is the basic element in the Building System Model [Turner 1992]. A Building Project Object is the highest level element. A Building Project Object has a Building and Site as its two variables as shown in Figure 4.7. The Building and Site objects are then decomposed into systems and components. The components are then decomposed into the complete model. The complete model includes a detailed model of a generic system and a classification of the building and site systems.

A system is designed to satisfy needs and perform a function. A system has system components and properties. A system component may also be defined as a system with its own system components. In the Building Systems Model, six kinds of building systems were identified; structural, electrical, circulation, plumbing, heating, and lighting. Each of these systems can be divided into active, passive and associative systems [Ekholm 1994]. The systems can also be classified as space related, fabric related and service related systems.

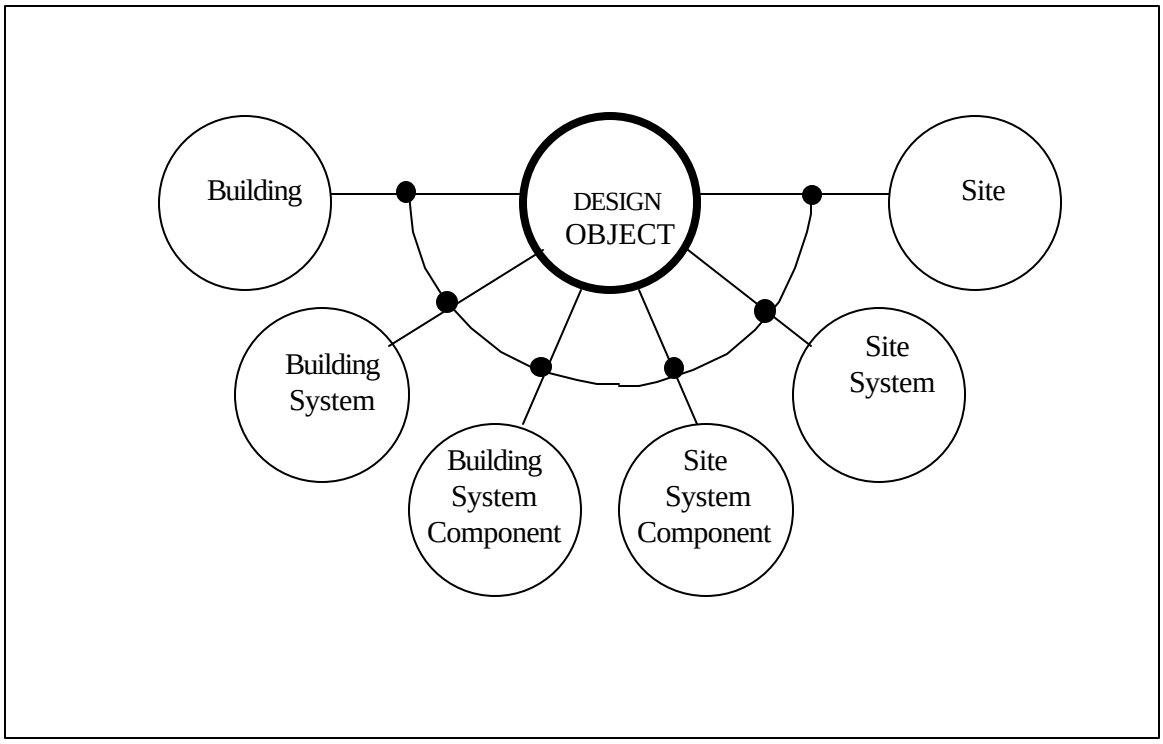


Figure 4.7. The Building Systems Model in NIAM notation [Turner 1990]

Furthermore, in the Building Systems Model, designed object can exist either as generic, specific or occurrence [Ekholm 1994]. Generic objects are objects whose exact variables are not specified. Specific objects are objects with most of their attributes known. Occurrences on the other hand are objects whose location and orientation have been determined, exactly or approximately. The Building Systems Model was criticized in two ways [Ekholm 1994]. The use of generic, specific and occurrence is limiting because during the design process, properties are not decided in the direction from generic to specific. Instead properties are decided in degrees of completeness. Also the systems hierarchy has been criticized as not being the proper

decomposition order. More details of the Building Systems Model can be found in the ISO report [Turner 1990].

4.3.1.2. SIGMA

SIGMA is a French project that aims to define a French reference model for the building trade and to initiate a normalization process [Tonarelli et al. 1997]. SIGMA is an integration of several French building product models (hence the name SIGM, the Greek letter symbolizing "the sum of"). SIGMA distinguishes three categories; SIGMA basic resources, SIGMA applications and the SIGMA core.

The basic resources are the most generic objects that are the building blocks for other objects. Basic resources are objects like geometry, date, actor and organizations and:

- Type objects: these are objects that can be referenced in the same project.
- Constraints: these are rules that project objects have to verify.
- Time Management: Successive task management objects to establish a work schedule.
- Geometry and Topology.

The second category defined in SIGMA is the application. This category aims to define application protocols that define how software developed for the SIGMA product model interact with the data. The SIGMA core groups, whose meaning is shared by all the intervening software, are the root of exchanges between the various software.

The SIGMA model seeks to define a generic reference model. From the point of view of data exchange, this is the traditional universal approach. The traditional approach suffers from being a rigid method for data exchange as described in section 2 above. In the next section we will look at the EXPRESS information modeling method and building product models using this method.

4.3.2. EXPRESS

EXPRESS is defined as an information modeling language to support the STEP initiative. The EXPRESS language is a formal data specification language with an object-oriented baseline,

based on Entity Relationship attribute model with generalization/ specialization and constraint specification. It is readable to humans and fully computer interpretable. Although EXPRESS was originally developed to describe information for product data, it is currently used for many other purposes outside STEP. EXPRESS is not a programming language. Instead it is an information modeling language that allows for procedures and partial method binding to entities. EXPRESS version II is currently under development. It is important to note that any model in EXPRESS can be shared [Wix 1997].

The fundamentals EXPRESS constructs are shown in table 4.1. A schema is a grouping/partitioning mechanism for an area of interest. All other EXPRESS declarations must occur within a schema. Declarations in one schema can be used in other schemas by means of interface statements. Interfacing allows for defining a schema and then using some or all of it to build others. The use of aliases is also provided for referencing.

Construct	**Meaning**
SCHEMA	- a wrapper for collections of related information
ENTITY	- a definition of an object - a real world concept
ATTRIBUTE	- the property for an object
TYPE	- a representation of value domains
RULE	- for handling of full and partial constraints
CONSTANT	- An unchangeable value
FUNCTION	-A procedural statement
PROCEDURE	- executable STATEMENT
USE FROM	- valid for ENTITY and TYPE only - treated as defined locally
REFERENCE FROM	valid for CONSTANT, ENTITY, FUNCTION, PROCEDURE, and TYPE. The object is not owned.

Table 4.1. EXPRESS Constructs

Data types represent domains of values. In EXPRESS there are many data types shown in table 4.2. The most important data type is an entity. An entity is a class of information which represents conceptual or real world physical objects which have common properties. An entity describes the data and a domain of values or specifies the relationship between the data structure.

Simple Data Types	
NUMBER	both INTEGER and REAL
REAL	all rational and irrational real number
INTEGER	Whole number
STRING	string of ISO 10646 characters
BINARY	string of bits 0 | 1
LOGICAL	one of FALSE | UNKNOWN | TRUE
BOOLEAN	one of FALSE | TRUE

(a)

Collection Data Types
ARRAY fixed size, indexed, multiple or unique occurrences. OPTIONAL, UNIQUE (sparsed) can be specified. ARRAY [1:10] OF UNIQUE INTEGER
LIST variable size, ordered, multiple or unique Occurrences LIST [1:?] OF UNIQUE REAL
SET variable size, un-ordered, unique occurrences SET [1:?] OF STRING (10) FIXED
BAG variable size, un-ordered, multiple occurrences BAG [0:?] OF NUMBER

(b)

Named Data Types
ENTITY (Described later)
DEFINED TYPE A type declaration creates a new type (a 'defined type') based on another type (the 'underlying type'). Named data types are used to increase the semantic of the underlying type.

(c)

Constructed Data Types
ENUMERATION data type Defines an ordered set of names Used to make data available as a set of static values
SELECT data type Makes defined types available as a grouped set of named data types

(d)

Generalized data type
Pseudo (or generalized) data types can only be used as the types of the formal parameters of functions and procedures
AGGREGATE - any kind of collection data type
GENERIC - every conceivable value

(e)

Table 4.2. EXPRESS Data Types

Subtypes and Supertypes provide Generalization/specialization in EXPRESS and are used to build a classification structure. A subtype inherits all of the properties of its supertype; attributes and constraints. The Subtype/supertype lattice is transitive. A subtype cannot be the supertype of itself, i.e. only acyclic graphs are legal. There are three inheritance methods in EXPRESS shown in Figure 4.8. Also, there are three subtype/supertype constraints shown in table 4.3.

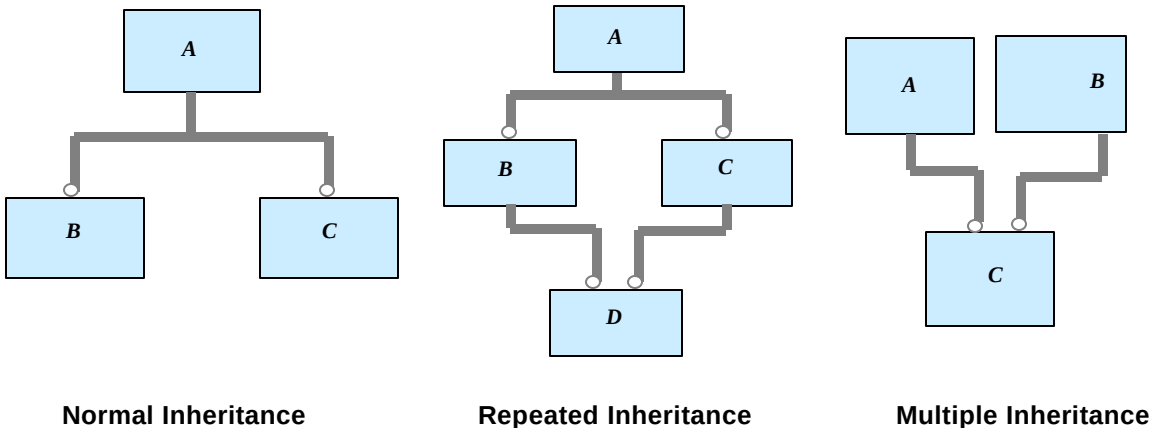


Figure 4.8. Different Kinds of Inheritance Structures in EXPRESS

Subtype/Supertype constraints	
ANDOR (default)	
Are not mutually exclusive, nor mutually inclusive 'all combinations'	
sub1, sub1+sub2, sub1+sub2+sub3, sub1+sub3, sub2, sub2+sub3, sub3, super	
ONE OF	
Subtypes are mutually exclusive	sub1, sub2, sub3, super
AND	
Subtypes are mutually inclusive	sub1, sub1+sub2+sub3, sub2, sub3, super

Table 4.3. Constraints in EXPRESS

Graphical symbols in EXPRESS-G correspond to declarations in the textual EXPRESS language (Figure 4.9.). However, EXPRESS-G is only a subset of EXPRESS; there are many EXPRESS declarations that cannot be represented in EXPRESS-G.

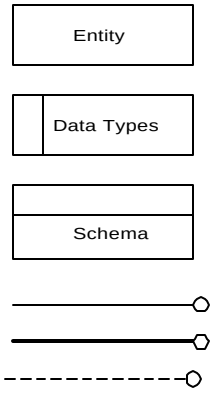
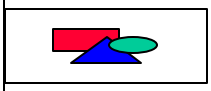


Figure 4.9. The EXPRESS-G, notation

In the next section will use present two examples of models developed using EXPRESS; the RATAS model and the Building Core Model.

4.3.2.1. RATAS

RATAS is a finish framework program for IT in construction. The acronym RATAS is derived from Finnish words for "Computer Aided Building Design" [Bjork 1989]. In order to facilitate sharing of information in building projects data is stored at a high semantic level as product models.

BUILDING	Author: Raine Talonpoika Organisation: VTT Building Technology Updated: 16.09.1996
	* A building is the wholeness which decomposes into spaces, structures, technical systems and furnitures.
Source of information (project (class)): TYR (Building) | FM-RATAS (Building) | FINNCORE (Building)	
Other references (project (class)): IFC (IfcBuilding)	
Decomposition level: Real_estate | Building | System | Subsystem | Part/space | Component	
Primary actor view: Client | Architecture | Structural | HVAC | Electricity/automation | Contractor | Supplier | User | FM	

```
ENTITY building
 SUBTYPE OF (ratas_object);
-- Properties --
     type                    : building_type_enum;
     total_volume              : volume_measure;
     total_area              : area_measure;
     effective_area            : area_measure;
     floor_area              : area_measure;
     number_of_floors            : number_measure;
     number_of_cellars           : number_measure;
     number_of_parking_places      : number_measure;
     number_of_stairs                    : number_measure;
-- Decomposition --
     spaces                  : SET[1:?] OF building_space;
     technical_systems           : SET[1:?] OF technical_system;
     building_parts                      : SET[1:?] OF building_part;
     components                          : SET[1:?] OF building_component;
-- Relationships --
     address                 : [1:1] address;
     owners                  : SET [1:?] OF owner;
     site                    : site;
END_ENTITY;
```

Figure 4.10. RATAS Definition Cards (adapted from [Hannus 1996])

The RATAS model, similar to other product models, describes a building using objects and relations between objects. Two types of relations are involved, the “part-of” relation and the “connected-to” relation. Objects are defined in EXPRESS in a format called definition cards as shown in Figure 4.10. Some areas are identified as future research areas in the RATAS research effort. For example, different types of connected-to relationships have to be investigated. Also, the description of shape and location is important and should be undertaken on an international level.

4.3.2.2. Building Construction Core Model

The BCCM (Building Construction Core Model) is developed as part of the STEP initiative; part 106 [ISO 1996]. The Building Construction Group was presented as an Application Protocol Planning Project. The model is a proposal for coordinated application protocol (AP) development. [Wix 1997]. It was recognized that the building construction industry is made up of a number of disciplines, each of which has their own application requirement.

The BCCM effort also identified that there is a set of information which needs to be exchanged between disciplines. This set of information, described as being of common interest, is less deep for any given application than would be the case for an AP but would have the capability to describe the means of exchanging data between application requirements. Such data were referred to as Core data and it is from this that the concept of a BCCM has developed.

A number of common objects make up the BCCM. These objects form the generic resources that several building-related applications can relate to. Figure 4.11. shows an example of a one such object, bc_element, in EXPRESS-G notation.

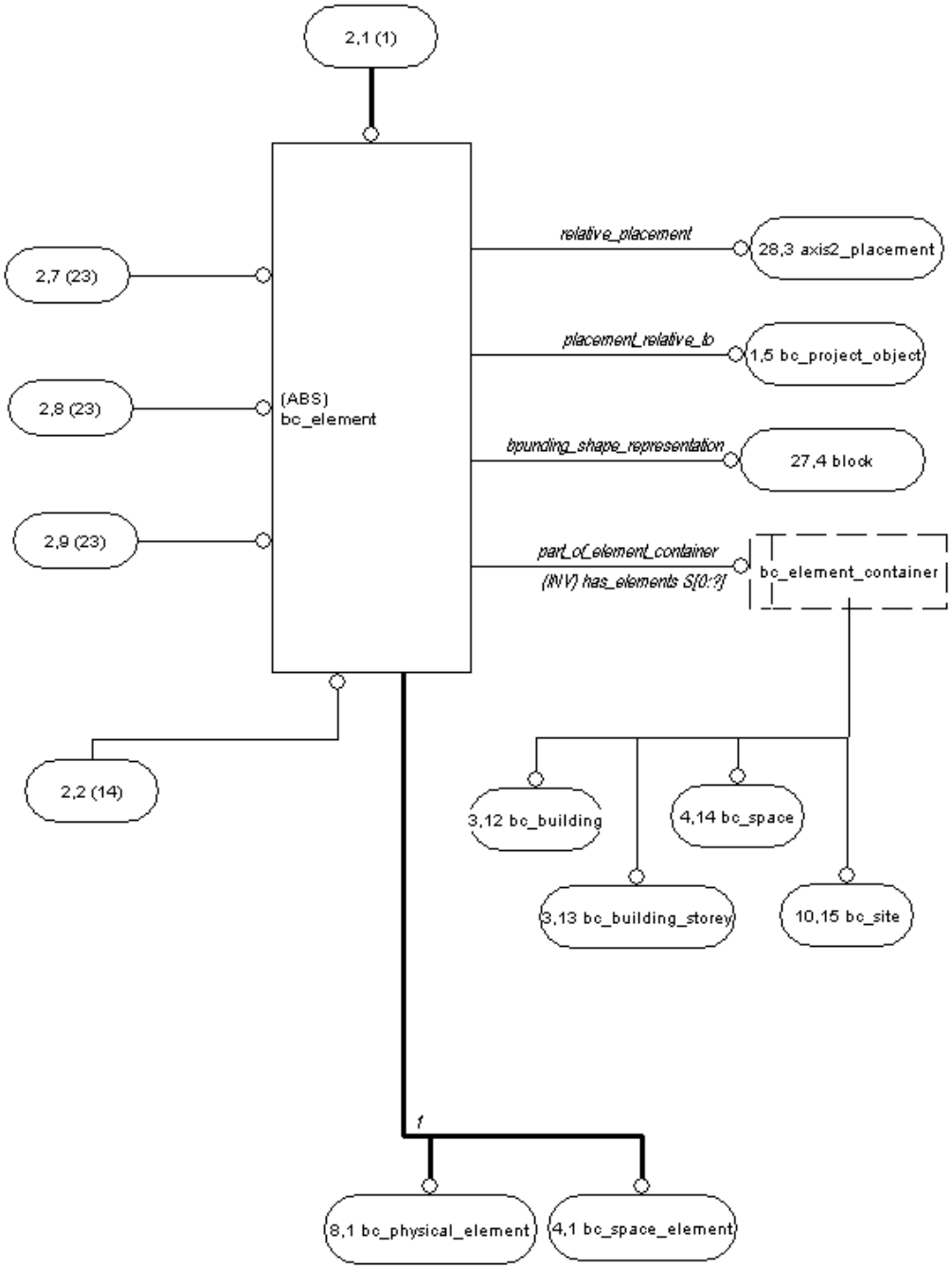


Figure 4.11. The bc_element object of the Building Core Model [ISO 1996]

4.3.3. IDEFx1

DEF1X is a method for designing relational databases with a syntax designed to support the semantic constructs necessary in developing a conceptual schema [Brown 1993]. A conceptual schema is a single integrated definition of the enterprise data that is unbiased toward any single application and independent of its access and physical storage. Because it is a design method, IDEF1X is not particularly suited to serve as an AS-IS analysis tool, although it is often used in that capacity as an alternative to IDEF1. IDEF1X is most useful for logical database design after the information requirements are known and the decision to implement a relational database has been made. Hence, the IDEF1X system perspective is focused on the actual data elements in a relational database. If the target system is not a relational system, for example, an object-oriented system, IDEF1X is not the best method. GARM is an example of a product model defined in IDEF1X.

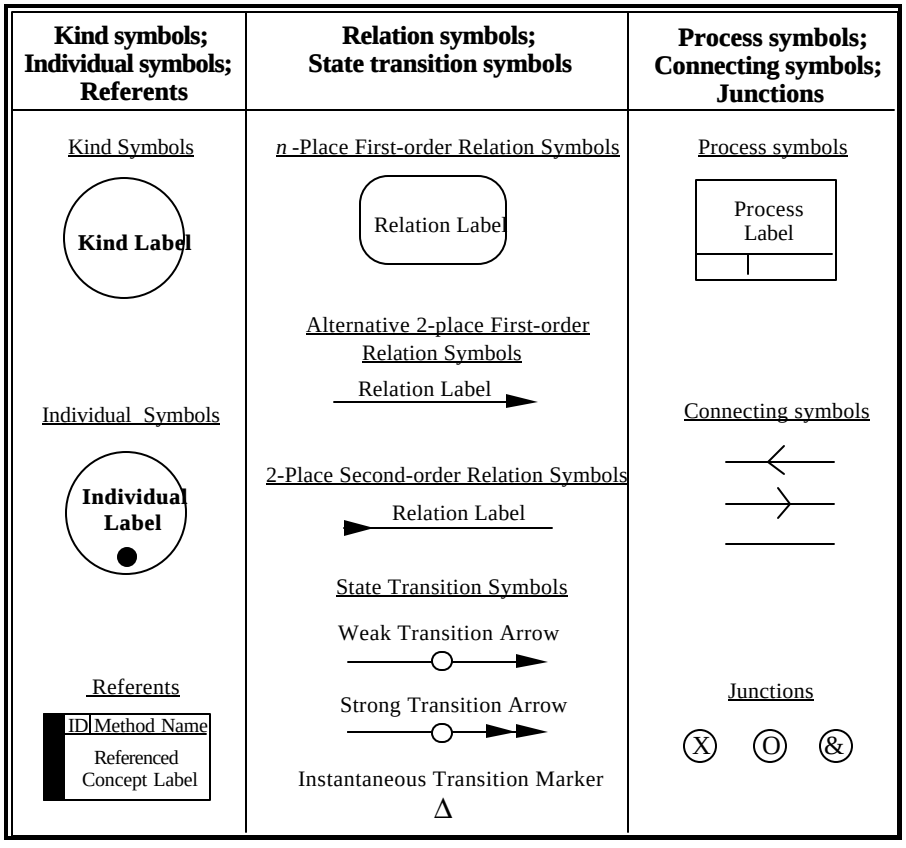


Figure 4.12. IDEF1X Constructs [Brown 1993]

4.3.3.1. GARM

GAAM (General AEC Reference Model) is one of the first building product models developed under the STEP initiative. GARM was developed as a generic model that would be used by all applications according to the traditional approach. The fundamental element in the GARM product model is the Production Definition Unit (PDU). The PDU is a generic design object, which has proven through subsequent research projects, to have application to many disciplines and building types (specialization types). An entire building, the structural system, a window or a single component like a brick can be a PDU. A PDU can also exist at three different levels: generic, specific, similar to the building systems model.

Because GARM was intended as a generic model, the PDU objects also exists at the different life cycle stages. A PDU an exist: as required, as designed, as planned, as altered, and as demolished. Decomposition relationships can exist for any PDU.

4.3.4. Feature Based Models

Feature-Based Modeling originates from the area of mechanical design. Shah [Shah et al. 1995] discusses this technique and its history. In the field of architecture the use of this technique is very recent, [Leeuwen et al. 1998]. Features in mechanical engineering describe characteristics of a product's part. Features can be defined as [Leeuwen et al. 1997]: *a collection of high-level information, possibly emerging during design, defining a set of characteristics or concepts with a semantic meaning to a particular view in the life-cycle of a building*. The important aspect here is that feature can be defined during the evolution of design.

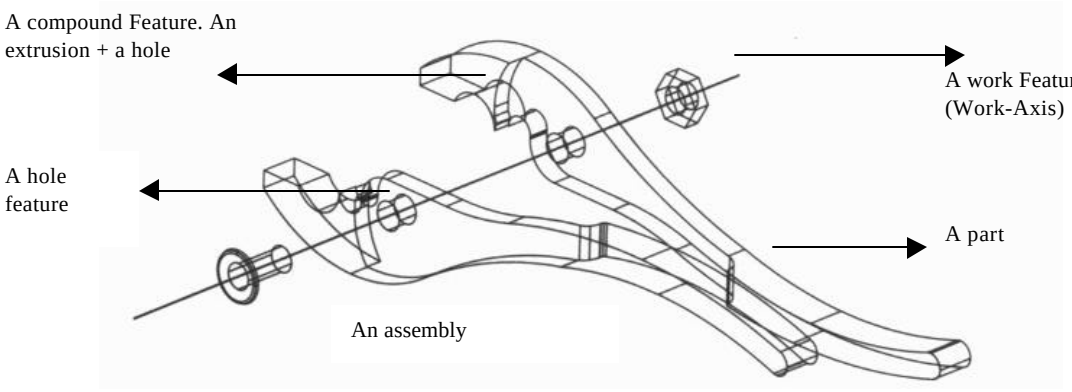


Figure 4.13. Examples of Features

Classification of features in mechanical design can fall into; Form Features, Precision Features, Material Features, assembly Features, Work Features and Constraint Features. Figure shows some feature examples of mechanical parts. These parts are combined to make an *assembly*.

Next we introduce an example of a feature-based product model, VIS-DIR.

4.3.4.1. VIS-DIR

The VIS-DIR (Distributed Interactive Simulation / Design Information Systems) project initiated by the group Building Information Technology of Eindhoven University of Technology, forms one of the potential areas of application of feature-based modeling [Leeuwen et al. 1995]. VIS-DIR uses virtual reality as a medium for simulating various processes related to buildings or their designs, and as an enhanced interface for design support systems. The technology used in VIS-DIR project needs to serve multiple disciplines and various aspects of building information. Flexibility and extensibility are also prime concerns. Therefor VIS-DIR requires an approach that allows for easy customization of data-definitions and software components. A feature-based model is developed to support the Virtual reality application of the VIS-DIR project. The model supports the various software components in VIS-DIR [Leeuwen, Wagter et al. 1995].

4.3.5. Engineering Data Model

Charles Eastman developed the Engineering Design Model (EDM), [Eastman et al. 1995], [Eastman 1994], [Eastman, Assal et al. 1995], [Assal 1992], over several research projects. The EDM information model has been developed to effectively characterize design knowledge and has been developed and tested with architectural concepts in mind [Eastman and Siabiris 1995].

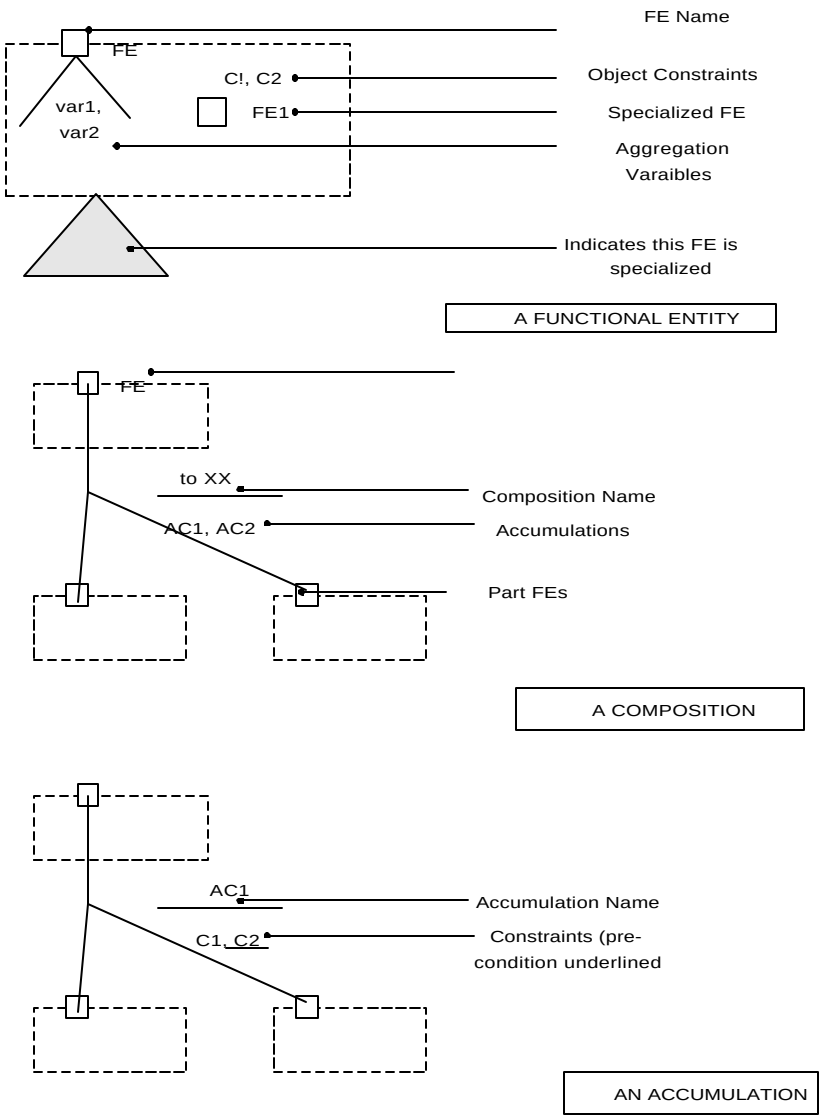


Figure 4.14. EDM Constructs (adapted from [Eastman, Assal et al. 1995])

EDM is based on a small number of structures to capture the semantics of design and engineering information. It uses sets and first order logic as a neutral meta-language for defining these structures, which are called forms. This high level meta-language is used to capture all the more specific representations used in design, such as geometry, color and textural images, drawings, analysis data and so forth. EDM represents both data and the relations between data like all the modeling techniques discussed so far. A relation in EDM is a rule that distinguishes a meaningful design from combinations of attributes that are meaningless. For example, a set of surfaces defining a volume enclosed solid, must satisfy closure, orientation and other rules.

Rules are defined for objects. These objects in EDM are called functional entities (FE). A functional entity consists of an aggregation and constraints, and also a set of other FEs. Compositions or accumulations can relate functional entities. EDM has a lexical format and a graphical format for specifying the different forms (FEs, accumulation and, compositions) as shown in Figure 4.14. The next generation, EDM2 is currently under development. We will present an example of a building model defined using EDM.

4.3.5.1. SPACE-ACTIVITY model

EDM has been used to define a generic building model [Eastman, Assal et al. 1995]. For the development of a robust and general purpose building model, EDM considers the geometric representation of a building to be a tight fitting set of solid models. Occupied spaces are represented a solids, as well as building components. A central design development issue in EDM, is defining a consistent set of such shape models that represents the shape of the building. In the model described in [Eastman and Siabiris 1995], Eastman, 1995 #22], the shape models describe the building spaces. The different parts of these shape models (e.g. the surface of a solid) describe different attributes of the building e.g. material finishes, properties and activities. More detail can be found in Eastman, 1995 #22].

4.3.6. Object Oriented Analysis

Several Object Oriented Analysis Methods (OOAM) exist like Shlaer and Mellor, Coad and Yourdon and, Booch [Graham 1994]. The objective of object oriented analyses methods, is to aid programmers in developing applications. OOAM have been introduced in the AEC industry by [Alshawi et al. 1996]. A case study is presented that uses the Coad and Yourdon's method to model a Bay Design system [Alshawi and Underwood 1996]. First the classes and objects are identified. Next, adding relationships between the identified objects and classes creates the major object oriented structures. The attributes are added and the services are defined. The services usually fall into six categories; create, connect, access and release, calculate and monitor. Create services initialize new objects, while connect services connect one object to another. Access services get and set attributes values of an object, while release services disconnect and delete an object. Monitor services monitor an external system or device. Each of these services is identified for the case study bay design system.

Another object oriented model is the Industry Foundation Classes (IFC). The International Alliance for Interoperability (IAI) is an action-oriented, non-profit organization established to define, publish, and promote specifications for IFC as a basis for worldwide AEC project information sharing throughout the project life cycle [IAI 1997]. IFCs define a single object oriented data model of buildings shared by all IFC-compliant applications. IFC project models define individual building for which compliant applications can exchange error-free information. IFCs are public for use by any member. However, software implementation of IFCs is proprietary to protect the data and technologies of member companies. Building elements are defined so that they have graphical intelligence. For example, if the wall is moved, elements contained in the wall (like doors and windows) move with it [Bazjanac 1998], [Yu et al. 1998]. With major AEC software development and industry forces behind it, the IAI will continue to develop IFCs.

4.4. A COMPARISON OF BUILDING PRODUCT MODELS

The building systems model, GRAM, RATAS, SIGMA/MOCIB and, the Building Core Model were developed as general reference models that follow the traditional approach for translation. Although, this approach was branded as wrong [Turner 1992], it is important to study the objects, relationships, breakdown structures and, the modeling technique used in these models.

Table 4.4. shows the different modeling techniques reviewed in this chapter. From our review we found that if the target system is not a relational system, for example, an object-oriented system, IDEF1X is not the best method. Also it was obvious that systems in Building Systems Model are parallel to PDU in GARM similar to feature types in the VIS-DIR model. Another similarity is that the basic resources in the SIGMA model are similar to generic resources of STEP and the IFCs.

Since one of the most important uses of the product model developed in this dissertation is data exchange, EXPRESS seems to be an appropriate choice. In this dissertation Express was used as the modeling technique for many reasons: EXPRESS is widely accepted and used. Any model in express can be shared [Wix 1997] since Step's initiative allows easier data exchange. Futhermore EXPRESS has a powerful object oriented Approach (multiple inheritance) and its powerful modeling definitions allows for easier model extension.

Modeling Technique Used	Example Models	Influences	Basic Modeling Element	Graphic Notation
NIAM	Building Systems Model SIGMA	Natural language Information Analysis	Object	Yes
EXPRESS	RATAS BCCM	ISO's STEP data exchange and standardization.	Entity	Yes
IDEFX1	GARM	Database and Process Modeling	Kind/ PDU	
Feature-Based	VIS-DIR	Mechanical Design	Feature	No
EDM	Space-Activity	Geometric Modeling	FE (Functional Entity)	Yes
OOM	Alshawy IFCs	Software Development	Class/object	Yes

Table 4.4. Modeling Techniques

4.5. SYNOPSIS

In this chapter we introduced building product modeling. The goals of product modeling were identified: firstly, formalize the development of data structures like Object Oriented classes or databases (which is used in this dissertation) and secondly facilitating data exchange. The data exchange use of product models was introduced and the two major approaches to data exchange using product modeling were discussed. Several information modeling techniques were also introduced. Several examples of product models developed using these information modeling techniques were also reviewed. It is concluded that the fundamental theory of most of these information modeling techniques is the same. The objects and relationships are the main constructs for most of the information modeling techniques, although their implementation varies. EXPRESS was chosen as the modeling technique to be used within this dissertation, to develop the database and the product model behind it. In the next chapter we begin describing the proposed database and the product model.

CHAPTER 5. THE DATABASE AND BUILDING PRODUCT MODEL

As was mentioned in chapter one, to automate the selection and generation of assembly construction at the design development stage, there are three main tasks: to represent the design and construction information in a computer-interrupted way, to develop an assembly selection procedure and, to develop an assembly generation procedure. The selection procedure will select the best assembly construction form a database, given certain criteria and constraints. The building product model allows us to develop the database design and represent constraints and criteria in a way that a computer can understand. In this chapter we describe the proposed relational database and the building product model behind it.

The database implementation provides a definition of the available assembly selections and their criteria and constraints. The database also stores the design concept description so that an automated selection procedure can be defined. The building product model, which the database is based on, consists of two main schemas, the DESIGN_CONCEPT and the ASSEMBLY. The entities of both schemas and the relationships between them will be presented. We will first start by looking at the steps in developing the database in section 5.2. Then we will look at the kinds of criteria and constraints that can be defined in the prototype tool EASYBUILD. Examples of selection criteria in EASYBUILD are presented. Examples of constraints are defined for a certain type of assemblies: external wall assemblies. Then we will describe the two schemas.

5.1. INTRODUCTION

The process of building design has remained unchanged for a long time. The introduction of computers in the design process has been mainly geared towards automating the drafting process. The amount of information to be calculated or entered by the user, to produce the final design drawings and details remains to be substantial. The computer can alleviate this burden, by automating some design, evaluation and selection tasks. Specifically, assembly selection automation can be best introduced at the design development stage. We have seen in chapter two that the design development stage lends itself best to automation, because at this stage most of the creative implicit design variables will have been worked out (for example the building shape and space allocation etc…). Furthermore, several of the design tasks to be performed at this the design development stage are usually characterized as parametric or prototype design tasks.

The goal of the research described here is to develop a new way to automate the selection and generation tasks at the design development stage. A design support tool that can automatically select and generate building assembly details automatically can be very helpful to designers in accomplishing the objective. In chapter two we have seen how such a tool can be integrated with the current design practices by using a description of the building design as different assemblies. Advances in CADD, design analysis and support software, and other computer software and hardware have enhanced the ability to achieve this goal.

However, the amount of information and design information that such a tool needs to have is very large and no one tool can accommodate all the knowledge required for that automation. Therefore the first stage of the research is to develop a formal building product model as seen in Figure 1.1. The definition of the product model (which is described in this chapter) tackles the problem of the vast amount of knowledge required, and is important for two reasons (as was discussed in the last chapter). Firstly, the product model provides the data structures (a relational database in this research) in the computer that can store and handle the building assembly constructions and their criteria and constraints knowledge. The product model allows for easier extension of the database. In this chapter only a few examples of the types of criteria and constraints that could be represented are given. Secondly, the product model facilitates data exchange. The lack of conventions in the structuring of data within CAD systems inhibits general interfaces and use of multiple applications (as was described in chapter four). In the last chapter we discussed the

neutral file format approach to data exchange and introduced the STEP initiative. By providing a product model in a formal language (EXPRESS) we can achieve easier mapping to the neutral file format and therefore simplify data exchange. The definition of the product model in EXPRESS makes it easy to exchange data with the other applications that are based on the STEP neutral file format. Easier mapping between the STEP neutral file format data structures and the EASYBUILD data structures is possible because both data structures are based on the same modeling language EXPRESS[16]. This mapping is currently performed manually, but can also be automated.

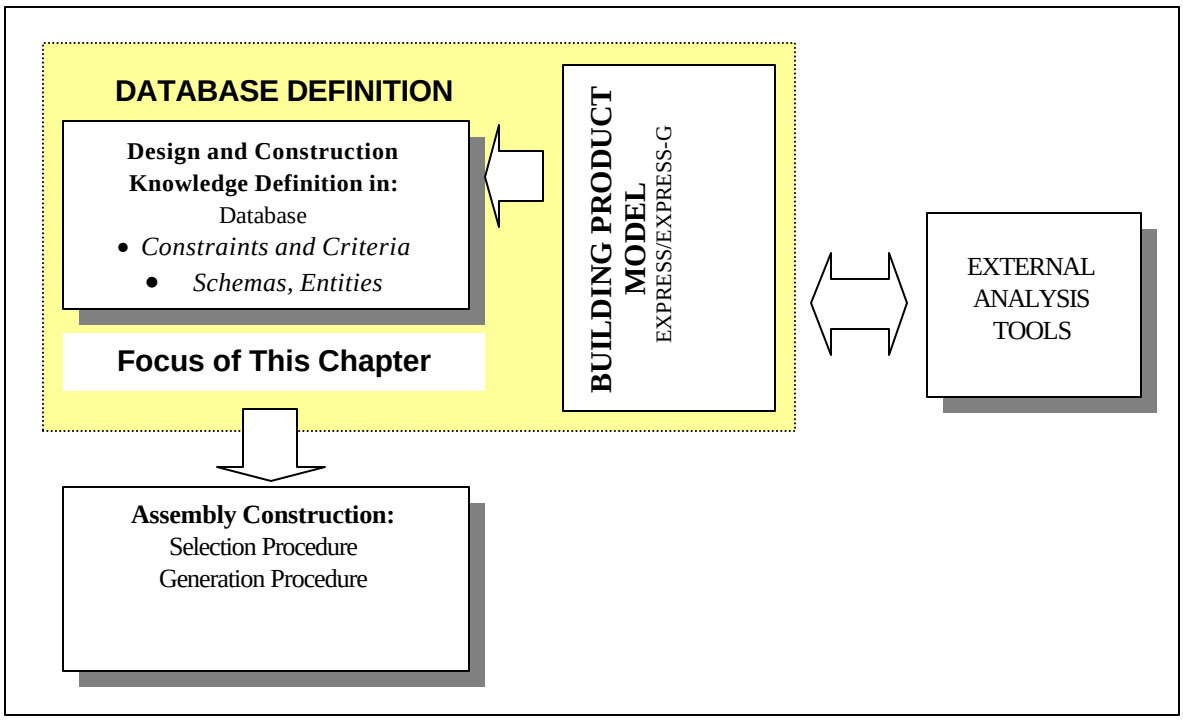


Figure 5.1. Focus of this chapter

The formal building product model is implemented as a relational database. The database implementation provides a definition of the available assembly selections and their criteria and constraints. The relational database also allows for storing the design concept description so that

[16] The STEP neutral file format is based on STEP's part 106. This part is currently under development and a first draft was issued in 1997.

an automated selection procedure can be defined. The formal product model, which the database is based on, can be used to define and change meta-knowledge like the relationships between the entities.

In this chapter we will present the building database and the building product. The assembly selection procedure and graphic generation procedures are described next chapter. We will start by describing the process of developing the database and the product model.

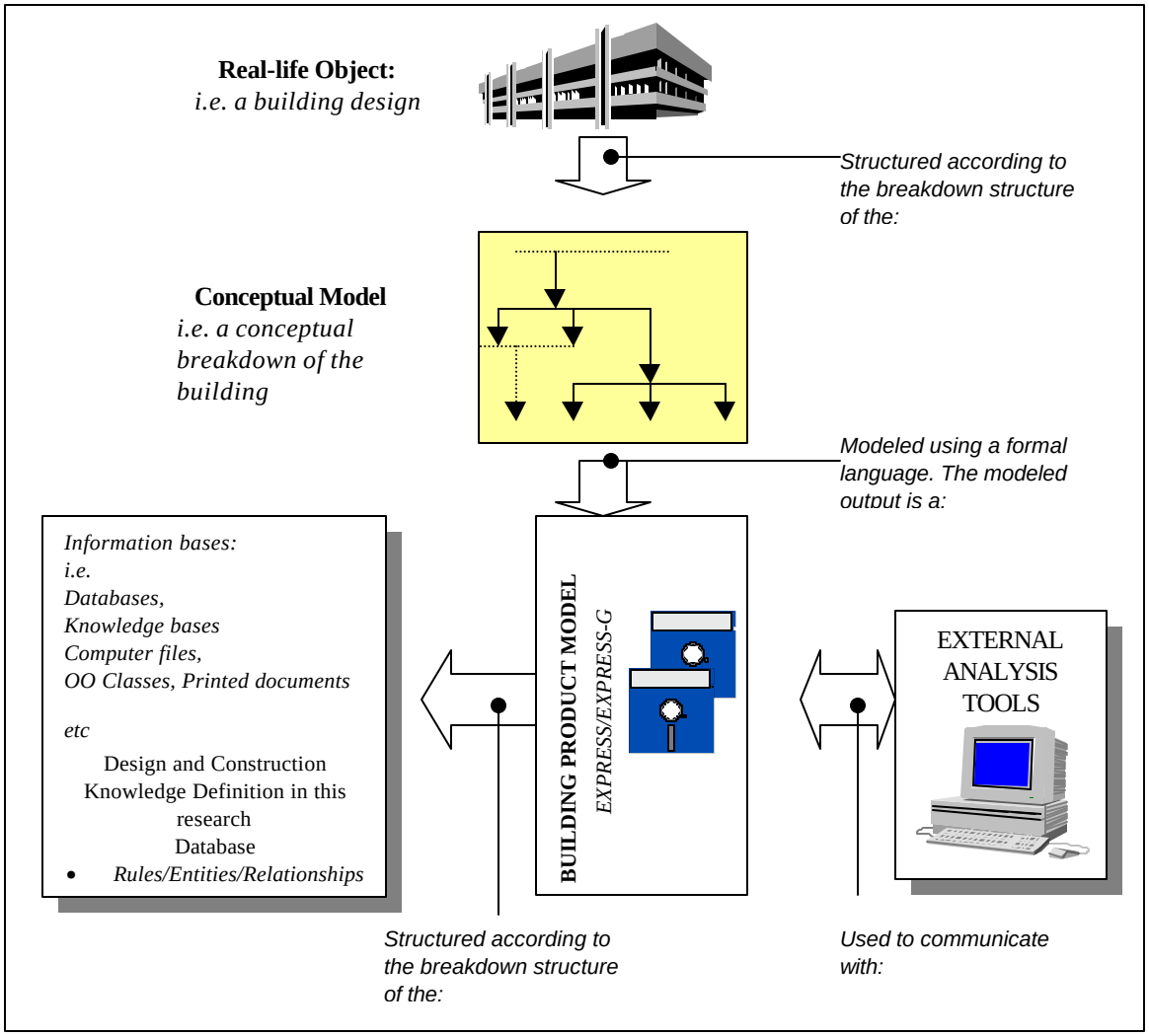


Figure 5.2. Layers of modeled objects

5.2. DEVELOPMENT OF THE DATABASE AND THE PRODUCT MODEL: LAYERS OF A MODELED OBJECT

In developing the relational database (and in the process also the product model) there are usually three levels of tasks required as shown in Figure 5.2. The real life object, (in our case the building) is structured according to a conceptual model. The conceptual model is a breakdown of the object into entities, their attributes and relationships. The breakdown structure of the building model described here supports the design and generation of building assemblies.

The conceptual model has to be described according to a formal method. The formal method provides generic constructs for describing conceptual models (for example entities or objects). By using the formal constructs it is possible to describe a wide range of concepts and relationships like inheritance, and decomposition. It is up to the modeler to develop the structure of the formal model. The reason for using formal methods is firstly to avoid ambiguity of the description. Secondly, using a formal method allows for a systematic and flexible way to extend the model by adding more entities or relationships. Thirdly, revisions of the model that would allow changing the actual structure of the model could be easily carried out. Fourthly, we can compile the model into several other information bases like databases (which is the information base used in this dissertation to support the selection procedure), computer files and printed documents. Finally the EXPRESS formal modeling language used here, facilitates data exchange.

In the next section we will describe the conceptual model developed. Then, the database and the formal product model are described in section . The reader can refer to [Schenck and Wilson 1994], [Date 1995] for definitions of the information modeling terminology used here.

5.3. THE CONCEPTUAL MODEL

Conceptually the building can be broken down in many ways. For example, the building can be broken down into the different spaces and then further broken down into the elements that make up the spaces, etc… The term broken down in this case implies the decomposition relationship. This is often called *has_a* relationship. There is no inheritance in this case, the component-types just include a part of the total attribute. For example, the spaces share the same *has_owner*

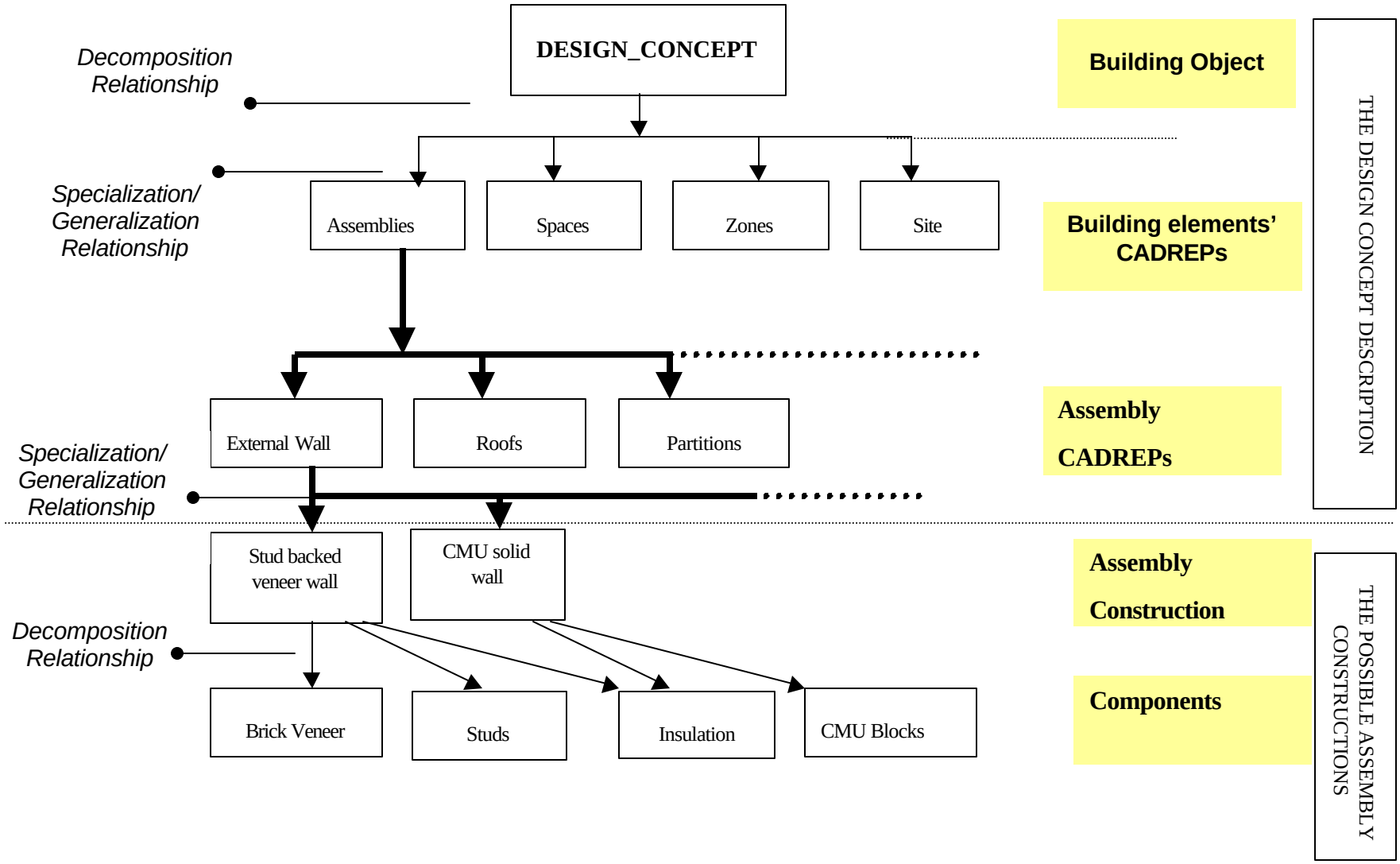
attribute of the building. However there are also specialization/generalization relationships that could be used in the building breakdown structure. For example a stud-backed-brick-veneer wall and a CMU wall are a subtype of the wall supertype. This is also called the Supertype/Subtype relationship and is often called *is_a* relationship. The subtype inherits all the attributes of the super type. For example, both the stud-backed-brick-veneer wall and a CMU wall will inherit the height attribute of the wall. The sub type might have its own attributes that are separate form the super type. For example the CMU wall might have a type_of_block attribute that the wall entity did not have. (Specialization relationships have also sometimes been used in information modeling).

Figure 5.3. shows the proposed breakdown of the building. This breakdown will support the selection and generation of building assemblies. The building here is broken down to five levels. The first level is the building object itself, design-concept. The DESIGN_CONCEPT is the volumetric description of the building from the schematic design stage. The DESIGN_CONCEPT is made up of the CAD representations of the various building elements. The building elements' CAD representations are called CADREPs. The CADREPs are the different decompositions of the DESIGN_CONCEPT. These CADREPs can either be assembly types, spaces, zones, or the site. The assembly CADREPs in turn are classified into several types. For example external walls, roofs and partitions are assembly types. These different assembly types inherit the attributes of the assembly type entity and therefore they have a specialization/generalization relationship with the assembly type entity.

The assembly types can be further specialized as the different assembly constructions, as shown in Figure 5.3. The different assembly constructions are further decomposed in the components that are used in the assembly construction. For example, a stud backed veneer wall can be decomposed into brick veneer, studs, insulation etc…

Therefore we can see that the five levels of the conceptual model relay two kinds of information. Firstly, the design concept and its CADREP decompositions, which are just the abstract graphical representations of the design and its assemblies without actual selected constructions. Secondly the various assembly construction options to be used for the various CADREPs and the components used in these assembly constructions. These are the actual assembly construction options that can be used.

Figure 5.3. The Building Conceptual Model



This conceptual model must now be presented as computer-interpretable information model i.e. a relational database. Therefore in the next section we will describe the database structure and the formal model required to support the database structure.

5.4. THE DATABASE AND THE PRODUCT MODEL

The conceptual model above was modeled in EXPRESS notation (a formal data specification language with an object-oriented baseline, described in the last chapter 4) as different schemas, entities and, relationships. The formal model can be found in appendix B. In order to understand the model, we will start in the next section (section 5.4.1.) by a brief description of the different modules of the EASYBUILD system. Then we will describe the database module of the EASYBUILD system.

The kinds of constraints and criteria defined in the database will then be presented in section 5.4.2. Example criteria implemented in the database are presented as well as the example constraints (for external wall assemblies) in section 5.4.2. This is followed by a description of the database's entities and schemas (their formal EXPRESS-G models and their actual database implementation) in section 5.4.3.

5.4.1. Brief Description of EASYBUILD

EASYBUILD is a prototype system based on the product model developed in this research. EASYBUILD is an application that allows the user to draw the building in terms of the assembly objects that define its form. Different kinds of information can be attached to the graphic assembly objects. The objects can be queried for the information latter on by other users. The system also has an assembly selection procedure (described in the next chapter) that helps the user select the best building assembly construction. EASYBUILD has an assembly detail generation procedure (also described in the next chapter) to help automate the graphical generation of the assembly 3D solid model detail. Furthermore, EASYBUILD allows for rules and constraints to be defined by an expert user to govern how the assemblies are used. The system

consists of three main modules; the DESIGN_CONCEPT definition module, the assembly generation definition module and, the database (with the criteria and constraints).

5.4.1.1. The DESIGN_CONCEPT definition module (DCDM)

The DCDM allows a user to model the building as building assemblies, e.g. walls, roofs etc… (as opposed to lines and circles that *represent* these assemblies). The user models his design automatically in three dimensions. EASYBUILD allows the user to specify different information about the building (e.g. location, building type, different requirements) and its assemblies (e.g. finish material requirements). EASYBUILD system also allows the user to specify certain requirements about the assembly construction to be selected, like finishing material or assembly types etc… The designer can select the assembly construction types for the different assemblies from the EASYBUILD database e.g. "*stud-backed brick veneer*" for wall assemblies or, "*single ply membrane roof on 4" concrete*" for roof assemblies. On the other hand these assembly constructions can be selected automatically by the selection procedure, based on a set of *criteria* and *constraints* defined in EASYBUILD.

5.4.1.2. The Assembly Generation Definition Module (AGDM)

Each assembly construction in the database is stored with the set of construction operations that are required to actually build the assembly. The assembly generation definition module in EASYBUILD is based on these construction operations. These operations are defined parametrically so that a graphical solid model can also be generated. The user uses these operations in the assembly generation module to construct the solid model the assembly. The generation procedure itself and the parametric operations are defined in the next chapter.

5.4.1.3. The Database

The EASYBUILD database (which is the focus of this chapter) is an implementation of the formal building product model. The database is used to store two separate pieces of information

(which are implemented as two different schemas, or database files, as we will describe in section):

- The design concept description (the DESIGN_COCNEPT schema)
- The possible assembly constructions (the ASSEMBLY schema)

Design concept description includes all the information of the building before the assembly constructions are selected for the different assembly types. In other words the various building assemblies are still just drawing objects (i.e. external wall, roof CADREPs, etc….). Assembly constructions on the other hand are the different possible constructions for the various assembly types. Using the selection procedure we select the best assembly constructions from the ASSEMBLY schema for the DESIGN_COCNEPT that best fit the set of performance criteria and satisfy a set of constraints. Therefore the database also includes:

- The various performance Criteria
- The various Constraints on assembly construction use

The EASYBUILD system is described in detail in chapter 7. Here we will only present the criteria and constraints that can be represented in the database (and examples of such criteria and constraints already defined in the EASYBUILD prototype system) as well as the entities and schemas that make up the database. First let us consider the criteria.

5.4.2. The EASYBUILD Database: Criteria and Constraints

The EASYBUILD database has an assembly selection procedure that selects the best assembly constructions based on a set user selected criteria (with importance weights of these criteria) and constraints. Therefore, in order to be able to automatically select the best assembly constructions the database must store two kinds of information: *criteria* and *constraints*. Criteria are the various performance measures for the various assembly constructions used for the various assembly types. Constraints specify rules on the use of certain assembly constructions. The criteria and constraints are then used in the selection procedure to select the best assembly constructions, as defined in the next chapter. Next we will look at the criteria and constraints that can be defined in the EASYBUILD database.

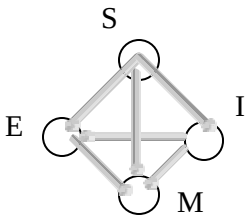
Building System Integration's Performances	DRIVE Performance Library
 Spatial Thermal Air Quality Acoustical Visual Building Integrity	Building Performances Acoustical Performance Economic Performance Fire Safety Performance Hygro-thermal Performance Suitability Performance Space Performances Acoustical Performance Economic Performance Fires Safety Performance Hygro-thermal Performance Suitability Performance Assembly Performances Mechanical Assembly Performances Electric Assembly Performances Economic Performances Architectural Assembly Performances Acoustical Performance Hygro-thermal Performance Fires Safety Performance Structural Assembly Performance Axial Element Performance Flexural Element Performance Frame Element Performance

Table 5.1. Different Building Criteria Classifications

5.4.2.1. Criteria

There are many ways to classify criteria for buildings. For example, the building performance criteria can be classified in different ways as shown in Table 5.1. However, for the purpose of selecting the best assembly constructions we found that the criteria can be classified to be either on the DESIGN_CONCEPT schema level or the ASSEMBLY schema level as shown in Figure 5.4.

5.4.2.1.1. Types of Criteria

1. *DESIGN_CONCEPT criteria* are those criteria that affect the building as a whole like annual cooling load or the cost per sq. ft.

2. *ASSEMBLY criteria* are those criteria that affect the single assembly entities like wall durability, or roof permeability, and do not require the whole building to be defined. DESIGN_CONCEPT criteria on the other hand take into consideration the interaction between the various assembly constructions and require all the assembly constructions to be selected before they are determined.

   For example, in EASYBUILD, the wall permeability is defined in the external wall table only and does not require other assemblies to be selected unlike the cooling load criteria which requires both the window and external wall assembly constructions to be selected (as shown in Figure 5.5.).

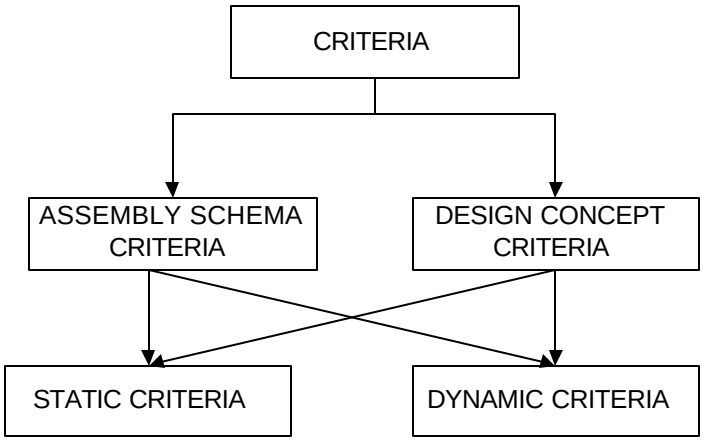


Figure 5.4. Criteria Types

Due to the fact that we have these two kinds of criteria the selection procedure (which is described in detail in the next chapter) becomes harder. Instead of just selecting the best assembly constructions that have the highest ASSEMBLY criteria (e.g. the wall construction with the highest permeability), we have to consider the inter-assembly effects. For example the wall construction with the highest permeability might be one that increases the annual cooling load.

The Criteria are also classified as either *static* or *dynamic*. *Static criteria* are included in the database as a simple data type and dynamic criteria are associated with a procedure that calculates the value of this criterion, as shown in Figure 5.5. Examples of static criteria included in

the database are simple properties of assemblies like permeability. On the other hand *dynamic criteria* that require calculation routines before their value can be entered in the database. Examples of such criteria are the building cooling load. EASYBUILD includes a set of criteria that are calculated internally (other external calculation procedures can also be added as will be explained in section).

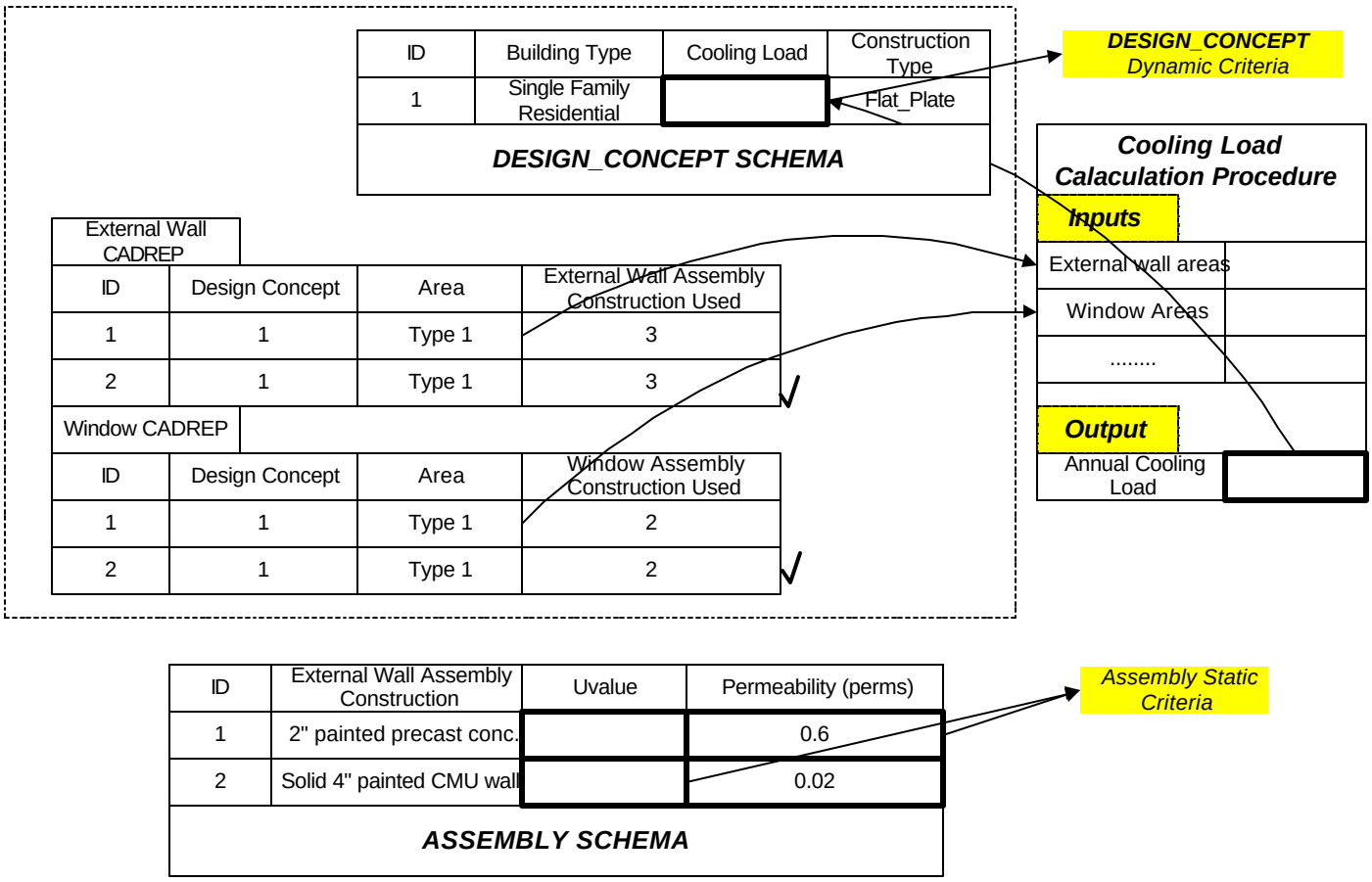


Figure 5.5. DESIGN_CONCEPT and ASSEMBLY criteria

5.4.2.1.2. Implemented Criteria Examples

Of course it is impossible to define all the performance criteria used to select building assembly constructions. Therefore in this dissertation only a few criteria are considered as examples. By structuring the criteria in the database it becomes easier to add new criteria as time goes on. Table 5..2. and 5.3. show the various criteria considered in the EASYBUILD system. Table 5.2. shows the example DESIGN_CONCEPT criteria, while Table 5..3. shows the example ASSEMBLY criteria for the various assembly types.

DESIGN_CONCEPT Criteria
❑ Initial Material Cost
❑ Annual Cooling load
❑ Square Foot Cost
❑ Area Weighted Fire Rating

Table 5.2. The DESIGN_CONCEPT various criteria considered in the EASYBUILD system

- The initial material cost is the total cost of materials used in the building. The initial material cost is calculated by Cost = $\sum_{1}^{n}\sum_{1}^{m} C_{nm}$ *where, n = number of assembly types, m = number of assemblies*. The cost data for the various assemblies is compiled from ^{RS}Means Assemblies Cost Data [RSMeans 1997].

- The square foot cost is another important criteria that affects assembly selection and is also considered in EASYBUILD.

- The annual cooling load is calculated using the ASHRAE cooling load temperature difference/cooling load factor/Solar Cooling Load (CLTD/CLF/SCL) method. This is a one-step procedure that is a simplification of a more elaborate method (transfer function method), and can be used for residential as well as other types of buildings. The formulae used in the CLTD/CLF/SCL method are shown in appendix A. For a complete description of the method refer to [ASHRAE 1996].

- Some DESIGN_CONCEPT criteria can be derived easily from ASSEMBLY criteria for specific design situations. The area weighted Fire Rating is another example of DESIGN_CONCEPT criteria that can be derived for design situations where fire rating is critical.

The various ASSEMBLY level schema criteria reflect examples of common performance measures often used in the selection of assemblies. The decision to incorporate these criteria in EASYBUILD was based on the availability of data, but other criteria can be added easily. The different assemblies and values for the criteria were collected from a variety of sources [Laseau

1980; Mahoney et al. 1986; Wakita et al. 1987; Herbert 1989; Allen 1990; Ballast 1990; CSI 1994; Allen et al. 1995] [ASHRAE 1996].

ASSEMBLY Criteria
External Walls
❑ Wall STC rating
❑ Interior Surface Material Sound absorption
❑ Wall Permeability/ Resistance
❑ U values
❑ Surface weight
❑ Number of layers
Roofs
❑ Roof Durability
❑ Maintenance
❑ Serviceability
❑ Compatibility (many corners in CADREP, high traffic, solvents and oils, building height, steel-deck)
❑ Impact Noise Reduction INR
❑ Permeability
❑ U-value
❑ Warranty
Slab on grade
❑ Compatibility (many corners in CADREP, load, soil type,)
❑ U-value
Floor
❑ Impact Insulation Class IIC
❑ STC
❑ Surface Resistance (grease, alkalies, stain, cigarette burns, indentation)
❑ Wet Cleanability
❑ Appearance retention
❑ Resilience
❑ Ease of Maintenance
❑ Durability
Internal walls
❑ STC
❑ Impact Insulation Class IIC
❑ Interior Surface Material Sound absorption
❑ Number of layers
❑ Surface Weight
Window
❑ Warranty
❑ Condensation Resistance Factors
❑ STC
❑ Visible light transmission %
❑ Opening Area %
❑ U-value
❑ Water Resistance Test, Structural Resistance Test , Air Infiltration Rate
Door
❑ Fire Rating
❑ Cost

Table 5.3. The various ASSEMBLY criteria considered in the EASYBUILD system

EASYBUID has some criteria options that are calculated or defined internally in the system, like annual cooling load, material and square foot costs and some assembly criteria like durability. Other criteria can be added to the system's database or by linking the database to external procedures that calculate criteria. Also the EASYBUILD database has examples of some constraints that restrict the use of certain assembly constructions. Next we will review the types of constraints and the example constraints defined in EASYBUILD.

5.4.2.2. Constraints

Constraints are rules that are defined on the attributes of the model. For example, a constraint could be added to indicate that the value of the height attribute of the DESIGN_CONCEPT could not be larger than a certain value based on the applicable building code. Constraints in EASYBUILD are used to eliminate the assembly constructions that are not applicable or can not be used for a particular design situation. Each assembly construction in the database has an *applicability* field. For example in the external wall construction database each assembly construction has the applicability field:

EXTERNAL WALL ASSEMBLY CONSTRUCTIONS				
ID	NAME	U-VALUE	STC RATING	***APPLICABILITY***
1	4" Painted Solid CMU wall			**0**
2	3" Painted Pre-cast conc. Wall			**1**

$$(APPLICABILITY)_i = \begin{Bmatrix} 0, \longrightarrow assembly\ construction\ i\ can\ not\ be\ used \\ 1\ \ , \longrightarrow assembly\ construction\ i\ can\ be\ used \end{Bmatrix}$$

The applicability field is used in the selection procedure to consider which assemblies can be selected. Each field entry has a boolean value (0 or 1) depending on whether that particular assembly construction (a record in the ASSEMBLY schema) can be used or not.

For each new project (i.e. design concept), the constraints are evaluated. The assembly constructions with an applicability field with a value of 1 (indicating that the assembly construction can be used) are short-listed (in an applicable assembly list) so that the best assembly among them can be selected (the best is defined by the criteria described in the last section).

EASYBUILD has examples of constraints. In order to be able to add more constraints, the definition and handling of constraints is formulated in a structured way. The constraints are specified as IF/THEN rules. Where the THEN part usually is the applicability field value (0 or 1). Each constraint is assigned a field with the constraint value as shown in Figure 5.6. The condition part (IF) of the rule is a relation between that constraint value and a property of the assembly construction (i.e. a field value).

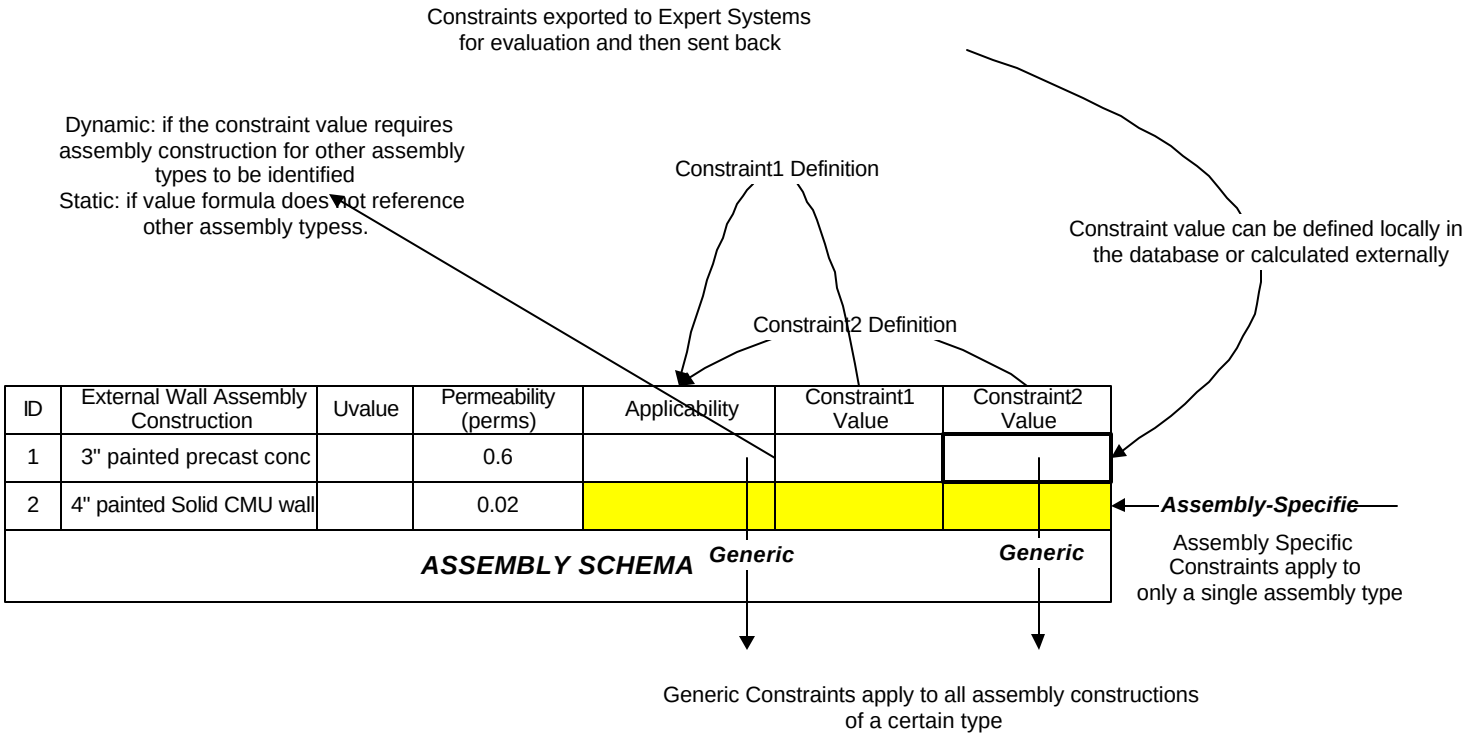


Figure 5.6. Constraint Handling

Therefore, the constraint definition format is:

IF *constraint value* <relation> *assembly property* THEN *Applicability* = 0,1.

For example in order to select the assemblies with the correct fire rating a constraint is added, with a constraint value field. The constraint value field contains the minimum fire rating for that particular design situation (based on a value derived from a table available in building codes). This value is then compared with the fire rating of the assembly construction to set the applicability field by using the following rule:

IF constraint_value1 > fire_rating THEN Applicability = 0

Of course, the result of this constraint will be different for each assembly construction in the database, but also for each new design concept since the constraint value (the minimum fire rating for external wall assemblies) also depends on the specific design concept.

Constraints can be defined in various database tables as constraints on the applicability field value or as external rules in a programming language. The <relation> part of the constraint can either be any of the traditional constraining relations (equality, range, list, etc…).

The definition of a constraint value field for each constraint allows for the value to be determined by external tools. For example in the rule above the fire rating constraint value was determined automatically from another table (the code table), but in other situations the constraint value can be determined using external procedures.

It is important to note that because we are trying to determine the applicability of a certain assembly construction the various constraints are combined by an *OR* operator (i,e, IF $Rule_1$ OR $Rule_2$ OR $Rule_n$ THEN Applicability = 0). Also the default value for the applicability field is 1. Constraints with circular references are not allowed. In the next section we will review the types of constraints that can be specified using this format.

5.4.2.2.1. Types of Constraints

An assembly construction can become inapplicable because of three main reasons: Code constraints, User Requirements constraints and Design and Construction constraints.

- Code constraints: Some assembly types might become inapplicable due to certain code limitation on their properties. For example, codes usually limit the fire ratings of certain assemblies to be more that a certain hourly rating for safety. Other code requirements limit the kinds of components that can be used in assemblies. For example the kind of mortar used in masonry assemblies is sometimes prescribed in certain codes depending on the location and type of wall.

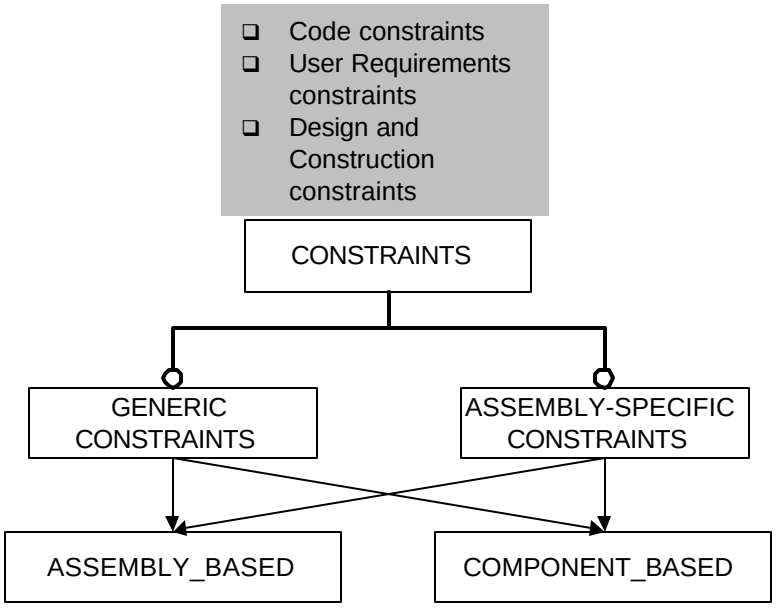


Figure 5.7. Constraints Types

- User Requirements constraints: Some assemblies also will become inapplicable to certain user requirements on the design. For example, designers often will require certain finishing materials for certain assemblies (e.g. external wall assemblies), due to aesthetic reasons. Assemblies that do not have the required finishing material are therefore excluded from the applicable assembly list. Other user requirements might limit the types of assemblies used. For example a user might constrain the type of assembly construction used for external walls to be of a masonry assembly construction.

- Design and Construction constraints: Often several assemblies will also become inapplicable due to certain design or construction requirements. For example certain external wall assemblies might become inapplicable if any surface or interstitial condensation is expected to

occur within the assembly. Certain assemblies might also become inapplicable due to the specific design situation. For example solid masonry walls are not recommended for locations with a high moisture or rainfall.

The constraints described above can be classified as *generic* or *assembly-specific* constraints as shown in Figure 5.7. Generic constraints are constraints that affect all the assembly constructions in the database table for a specific assembly type. On the other hand assembly specific constraints only affect certain assemblies in the database. In EASYBUILD example of constraints are defined for a specific assembly type: external wall assemblies. A number of generic and assembly-specific constraints are defined in EASYBUILD as shown in Table 5.4. So for example, a generic constraint defined in EASYBUILD for the external wall assemblies checks for the fire resistance ratings for the external wall assemblies to make sure that they meet code (BOCA code). On the other hand EASYBUILD also has examples of assembly specific constraints, e.g. constraints on masonry wall heights and spans only affect external wall assemblies that are of masonry type to check that the masonry construction can be used for wall CADREP's height and span defined by the user.

We found that both generic and assembly-specific constraints come in two conceptual subtypes: *assembly-based* or *component-based* constraints. If the constraint is based on a value of the assembly as a whole it is considered an assembly-based constraint. An example of an assembly-based constraint is the fire rating of the assembly (discussed above) because this is a property of the assembly as a whole. On the other hand if the constraint is on a property of an assembly component it is considered a component-based constraint. An example of a component-based constraint is the constraint on mortar type used in a masonry assembly based on the design situation.

Furthermore, constraints can be Persistent or Temporary. Persistent constraints are specified on the base template and therefor affect all the projects that are based on that template. Temporary constraints are constraints that are specified on the template instance for a specific project and therefor do not affect new projects.

5.4.2.2.2. Implemented Constraints

Table 5.4. shows examples of constraints defined in EASYBUILD. These are examples of the types of constraints described above which are currently implemented in the system. As an example of code constraints (the BOCA code), limits on the heights and spans of masonry assemblies are checked (also the fire resistance of external wall assemblies is checked, as we mentioned). Examples of user-requirement constraints are constraints on external wall assemblies finishing material, assembly types and cost can be defined by the user. As an example of design constraints, the condensation constraints check for surface and interstitial condensation in external wall assemblies using a procedure described in appendix A.

➢ ***Assembly-specific (Example used in EASYBUILD: masonry wall assemblies)***	***SUBTYPE***
❑ Surface Condensation	❑ Assembly-Based
❑ Interstitial condensation	❑ Assembly-Based
❑ Non-load bearing masonry walls heights and spans (based on code)	❑ Assembly-Based
❑ Masonry wall assembly's brick and mortar type constraints	❑ Component-Based
❑ Joint sizes	❑ Component-Based
❑ Brick Veneer tie spacing	❑ Component-Based

➢ ***Generic Constraints (Example used in EASYBUILD: External Wall Assemblies)***	***SUBTYPE***
❑ User Requirement on assembly construction types	❑ Assembly-Based
❑ Maximum Cost requirement	❑ Assembly-Based
❑ Fire ratings on external wall assemblies (based on code)	❑ Assembly-Based
❑ User Requirement on Finishing material	❑ Component-Based
❑ Finishing material types	❑ Component-Based

Table 5.4. The example constraints in EASYBUILD

It is important to note that constraint evaluation in EASYBUILD is carried out using first order logic only. Second Order Logic evaluation of constraints however could be achieved by linking the system with an external expert system tool. Other semantic integrity constraints like unique, inverse and cardinality constraints can be specified in the EXPRESS model (appendix B). Now that we have presented the constraints and criteria we will present various schemas and entities used in the database.

5.4.3. The EASYBUILD Database: Schemas and Entities

The EASYBUILD database is based on the EXPRESS product model (appendix B). The EXPRESS schemas are implemented as excel templates in the EASYBUILD database. Constraints on the applicable assembly construction are defined in the database. Instances are created by populating the templates and adding new records. For example, the wall assembly construction entity is implemented as a template. A stud-backed brick veneer wall assembly construction instance is created by using the template to fill in the attributes in the template and defining rules and procedures in the created instance. The created instance is stored as a record in the database.

Schemas in EXPRESS are a collection of entities that relate to each other in a certain way. In the developed model there are two schemas, DESIGN_CONCEPT, and ASSEMBLY. The schemas are implemented as two separate database files with certain relations between them. The DESIGN_CONCEPT schema in EXPRESS is used to store the data of the building design concept at an abstract level, i.e. before the assembly constructions are selected for the various assembly types. The DESIGN_CONCEPT schema includes entities like, Space, zone, site, location, CADREPs, Construction Types, and Building Types.

The ASSEMBLY schema represents the various assembly constructions that can be used for the various assembly types. The ASSEMBLY schema is implemented as the database of the available assembly construction options. It includes entities like External Wall constructions, Internal Wall constructions, Components, etc…

Each of the entities in the developed model has a set of attributes. The attributes can either derived or explicit. Derived attributes are attributes that are calculated automatically by the computer using the defined procedures, or constraints. Derived attributes can be graphical derived form the CAD model. For example areas, heights etc… Non Graphical derived attributes are attributes like column width (which is calculated based on the load attribute)..

Explicit attributes are those attributes that are user input. Explicit attributes are asked for when object is instantiated. Both the derived and explicit attributes can either be simple or relational attribute or other entities. Simple attributes are atomic attributes that are of simple data types e.g. name (string), ID (integer), etc… Relational attributes are pointers to other entities e.g. Perpendicular_wall, adjacent_space, etc…

A special type of explicit attribute is the requirement. A requirement is an explicit attribute that must be met. A requirement attribute is usually associated with a constraint to specify how this requirement is met. Requirement are therefor user specified and examples of different requirements are given for the external wall assemblies. Explicit attributes can also be optional (indicated by a doted line in the EXPRESS-G diagrams).

Next we will review the two schemas of the product model. For each schema, we will present the formal EXPRESS-G representation first and then the database implementation in terms of the tables and relations between them. The complete EXPRESS model is included in Appendix B.

5.4.3.1. DESIGN_CONCEPT Schema

The DESIGN_CONCEPT is the topmost entity that represents the building design. In most cases, the designer's idea is manifested in a volumetric description of the building using assemblies, construction type, and other requirements. The DESIGN_CONCEPT is the user-input representation of how the designer wants the building to "look like" as well as a set of design requirements like constraints on finishing material, maximum cost, etc.

Specifically the DESIGN_CONCEPT entity includes a definition for:

[1] The form of the building (morphology, things like height, width, width to length ratio, geometric formations like projections, solids and voids etc…). This is described by abstract representations of building assemblies (planes, or boxes) called CADREPs, as shown in Figure 5.8.

[2] A choice of construction type, like post and beam concrete, masonry wall bearing, platform framing, etc. This determines what assembly types can be used. For example, a flat plate construction type, does not have a beam assembly type.

[3] Space distribution and allocation to activities (The bubble diagram concept). This is also described by our configuration of building assemblies, as well as 2D space representation, called space CADREPs.

[4] Other attributes and requirements like building type, number of occupants and maximum cost allowed.

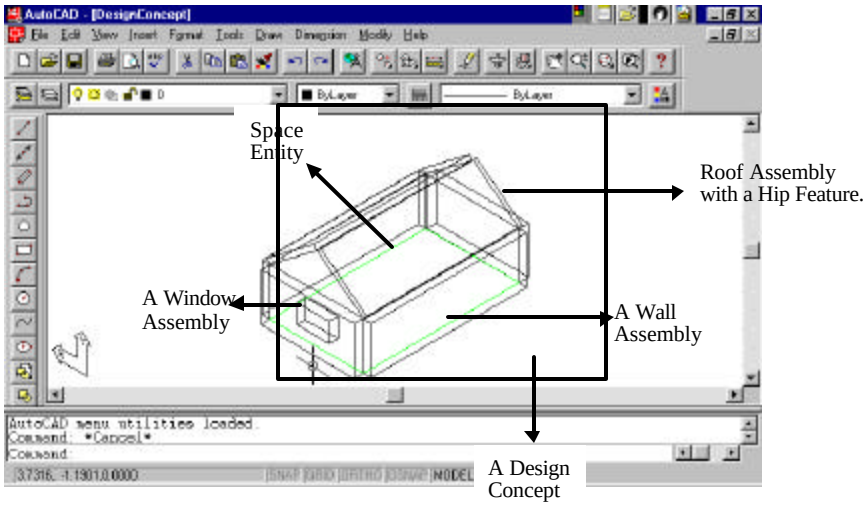


Figure 5.8. An Example of a DESIGN_CONCEPT

5.4.3.1.1. EXRESS-G Model

As described in the last chapter, EXPRESS-G is a graphical language whose symbols correspond to declarations in the textual EXPRESS language. In the next section will use EXPRESS-G as the language of the diagrams to develop the formal model. The EXPRESS-G model of the

DESIGN_CONCEPT entity is shown in Figure 5.9. We will review the entities, attributes, and constraints currently defined in the EASYBUILD model.

The explicit attributes defined in the DESIGN_CONCEPT schema are shown in Figure 5.9. In EASYBUILD, the construction type can be one of the following (although the prototype is defined for the two_way_flat_plate construction):

ID	Construction Type Name	Generic Type
1	Platform Framming	Wood and Masonry
2	Timber Framming	Wood and Masonry
3	Ordinary Construction	Wood and Masonry
4	Mill Construction	Wood and Masonry
5	Light Gauge Steel Framming	Steel
6	Single-Storey Rigid Steel Framming	Steel
7	Steel Frame-Hinged Connections	Steel
8	Steel Frame-Rigid Connections	Steel
9	One-Way Solid Slab	SiteCast Concrete
10	Posttensioned One-Way Solid Slab	SiteCast Concrete
11	One-Way Joist	SiteCast Concrete
12	Two Way Flate Plate	SiteCast Concrete
13	Posttensioned Two-Way Flat Plate	SiteCast Concrete
14	Two-Way Flat Slab	SiteCast Concrete
15	Posttensioned Two-Way Flat Slab	SiteCast Concrete
16	Solid Slab	PreCast Concrete
17	Hollow Core Slab	PreCast Concrete
18	Double Tee	PreCast Concrete
19	Single Tee	PreCast Concrete

The location entity is also an explicit attribute of the DESIGN_CONCEPT. The location has several attributes of its own and is shown in Figure 5.10. Variables like azimuth, altitude and rainfall are attributes of the location entity. Two instances of the location entity are currently defined in the EASYBUILD system (Blacksburg and Cairo). Other explicit attributes are the occupancy and, building type. The building type entity has a use group designation based on code. For example, a detached residential dwelling has a use designation of “R4” in the BOCA, “R” in the SBC and, a “C” in the NBCC.

The DESIGN_CONCEPT is decomposed into space, zone and site. The space entity is shown in Figure 5.11. The space entity has occupancy, loads and space type as its attributes. The space type defines the use of the space and also the special requirements like maximum occupancy and

area requirements. Design criteria can also be assigned at the space entity level. For example in EASYBUILD, a space entity has the indoor temperature as a criteria attribute.

The DESIGN_CONCEPT entity also has a set of derived attributes. For example, the material cost attribute is derived based on the assembly constructions used. Other derived attributes are the graphically derived attributes like building height and the building area.

There is a set of derived criteria attributes or performance factors that are measured at the DESIGN_CONCEPT level. As an example of DESIGN_CONCEPT derived criteria attributes, two criteria are defined in EASYBUILD for the DESIGN_CONCEPT: Annual Cooling load and Material cost (other criteria and requirements are defines for the various assemblies as will be explained later in this section).

The DESIGN_CONCEPT entity also has the CADREPs entities as its attributes (shown in EXPRESS-G in Figure 5.12.). A CADREP CADREP (short for CAD REPresentation) is a graphical representation of the assembly. A CADREP is an atomic element that can be scaled in three directions and rotated in 3D. In EASYBUILD CADREPs are defined as AutoCad block files that are inserted in the drawing and then scaled and rotated to describe the way the DESIGN_CONCEPT "looks like". Each of the assemblies types (walls, roof, etc…) that make up the DESIGN_CONCEPT and also the spaces are defined graphically in terms of CADREPs.

For example, in Figure 5.13. the CADREP for a wall is solid box with unit dimensions. By scaling the box we get the representation of the wall. Another example is the roof assembly. The CADREP for the roof assembly is a unit A-shaped prism. By scaling the height and width of the prism it is possible to define the roof representation.

The various assemblies drawn in Figure 5.13. are examples of CADREPs. CADREPs can be solid models or surface models, depending on how they are defined in the block file. Different designers will prefer certain representations for their assemblies. While some would like to use 3D solid models representations other will want to describe their building DESIGN_CONCEPT in wire-frame or surface models.

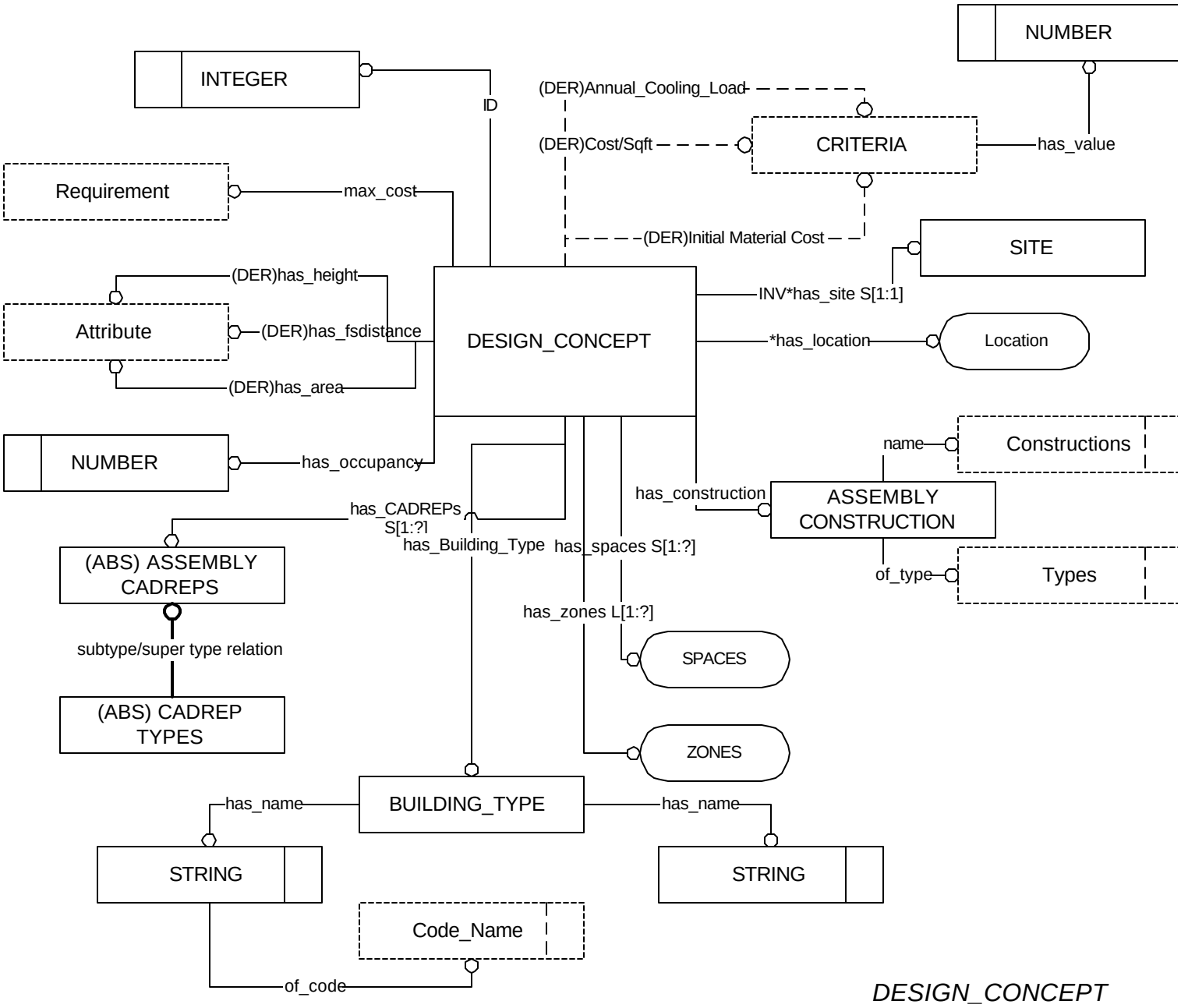


Figure 5.9. The DESIGN_CONCEPT Entity and its attributes in EXPRESS-G

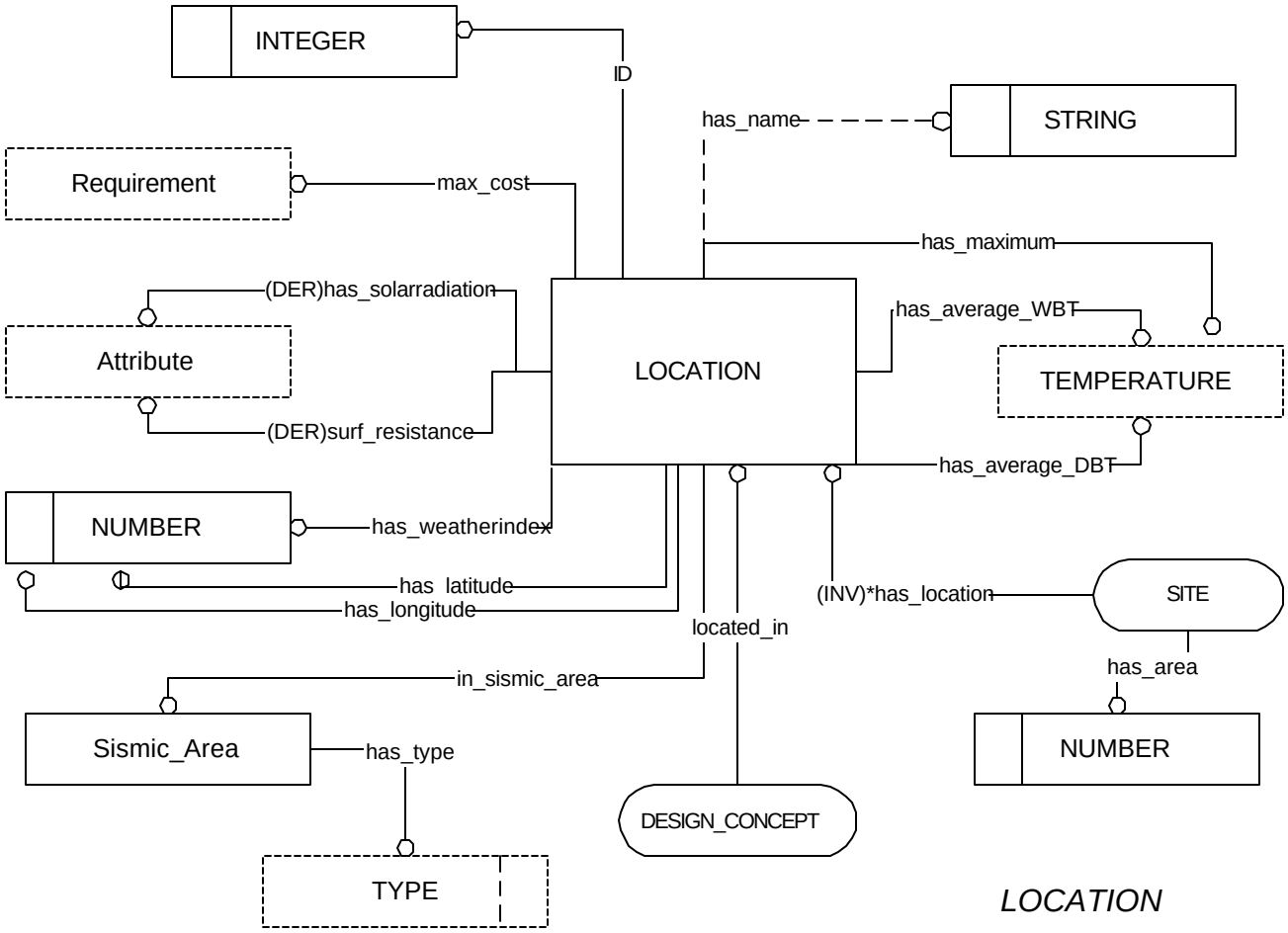


Figure 5.10. The location Entity

CADREPs are defined for the various assembly types (walls, roofs etc…) and also for spaces, zones, and sites. Different CADREPs can be defined using different blocks. For example a circular wall would have a CADREP that is a cylindrical section, as shown in Figure 5.13. By appropriately manipulating the CADREPs for various assembly types, it is possible to represent a design configuration. This gives flexibility for the designers to define different building designs.

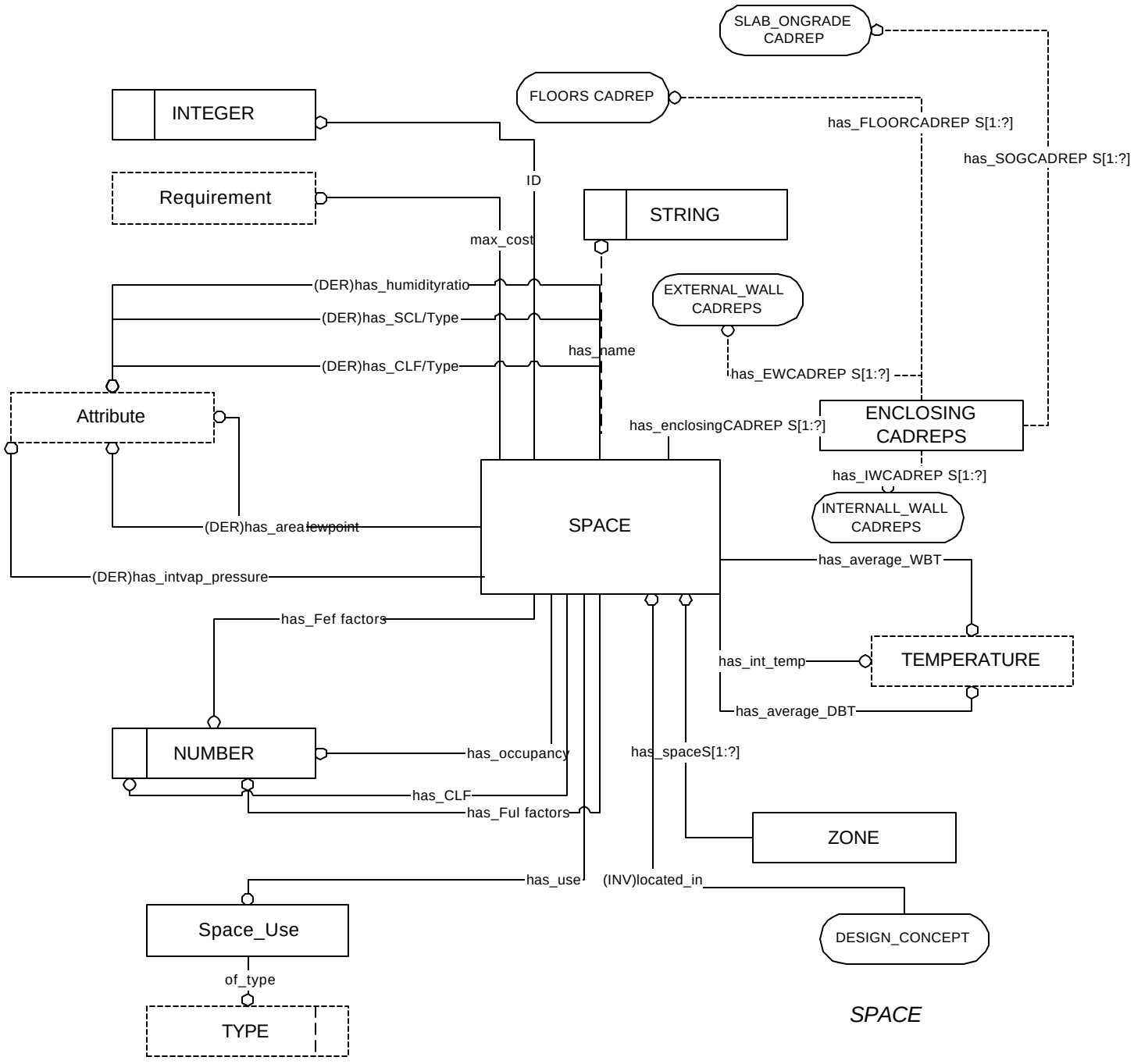


Figure 5.11. The Space Entity

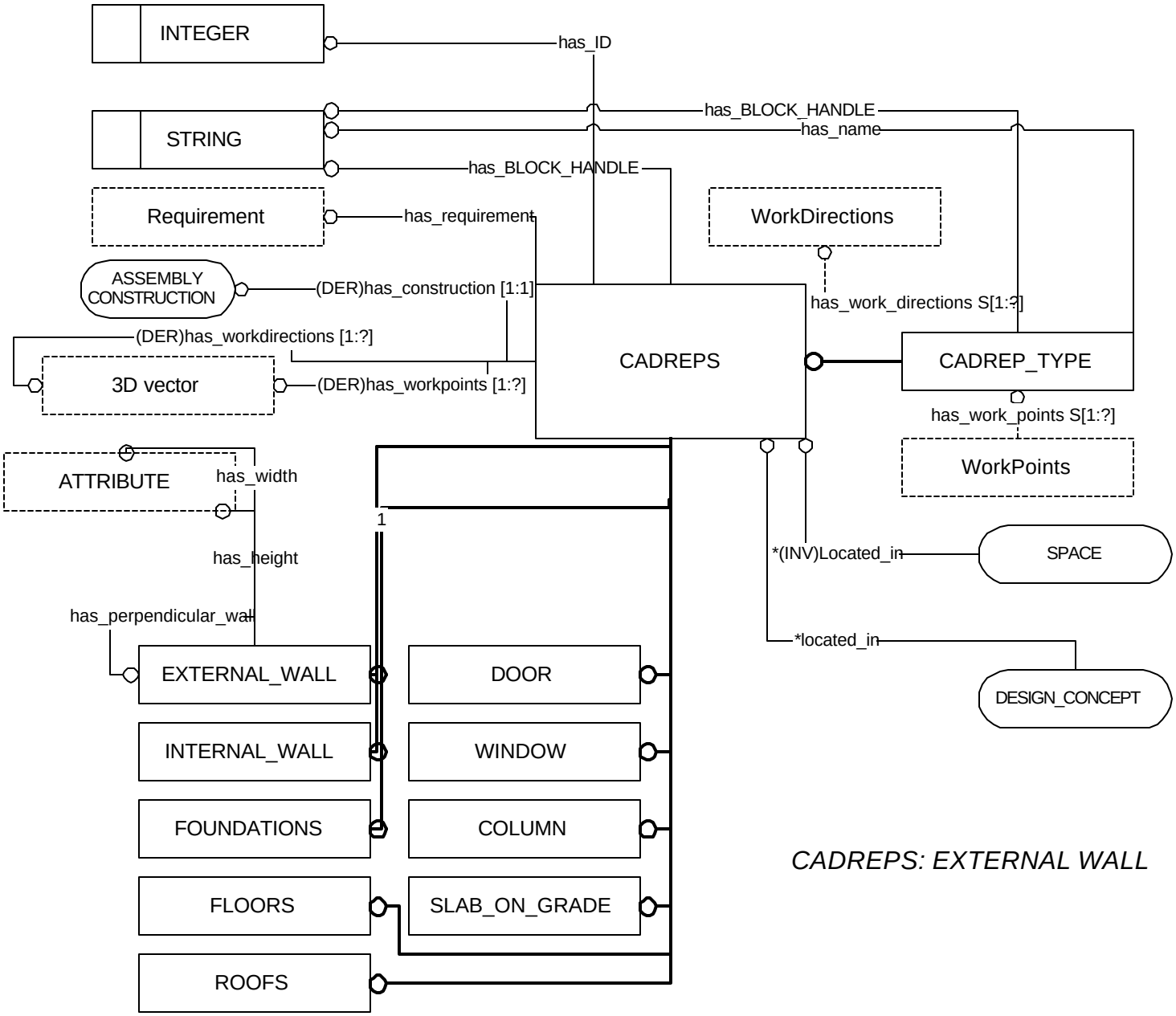


Figure 5.12. The CADREP entity

The geometric information associated with the CADREP geometry is extracted from the block file and stored in the AutoCad files as Xdata (Xdata are textual and geometric data that can be added to the drawing elements). For example a solid box CADREP is stored in the AutoCad block file as a solid model (a boundary representation solid structure in AutoCad). The height of the box can be extracted and saved as Xdata for the box and then in turn saved in the database.

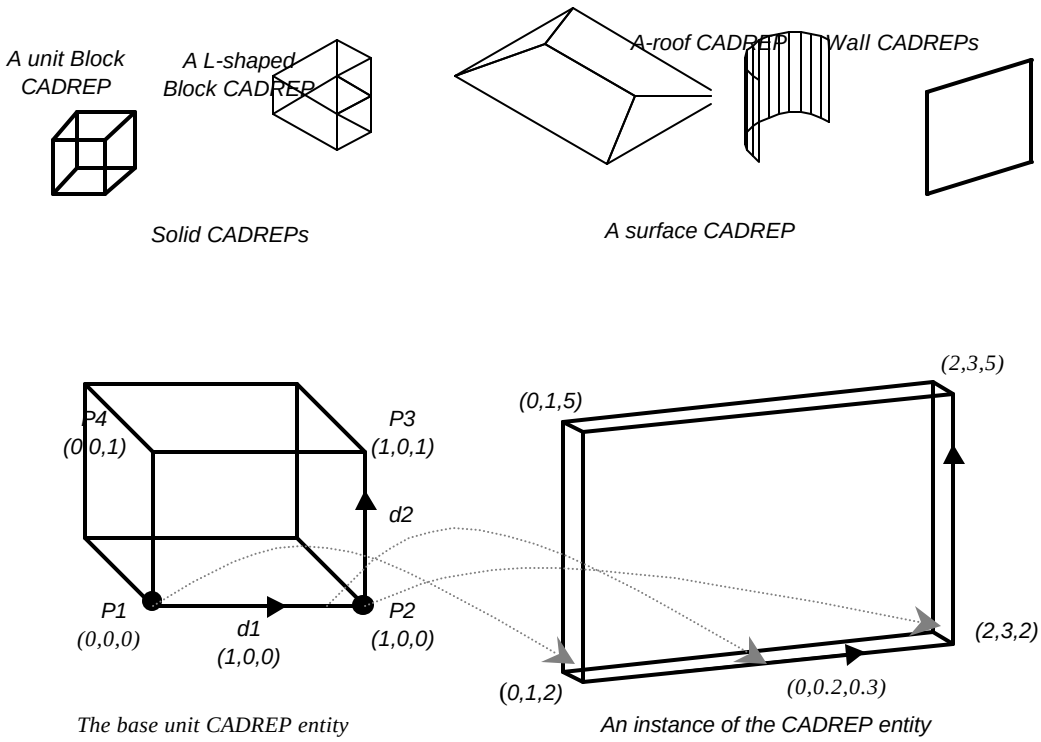


Figure 5.13. Examples of CADREPs

CADREPs also have work features as its attributes. The work features can be points or directions. The work features are geometrical elements that remain in their relative position and direction when the CADREPS are scaled or rotated. Work features are used when specifying the graphical generation procedures in EASYBUILD (described in more detail next chapter).

5.4.3.1.2. *Database Implementation*

The DESIGN_CONCEPT schema was implemented as a relational database file with several tables shown in Table 5.5. The relations between the various tables are shown in Figure 5.14. The relations are based on the defined relations of the entities in the product model. For example the DESIGN_CONCEPT table has a "one_to_many" relation to the Location table (via the DESIGN_CONCEPT's location field, as shown in Figure 5.14.). Each record in the Location table can have many matching records in the DESIGN_CONCEPT record (i.e. many buildings can be located in the same location e.g. Blacksburg, Cairo). However, each record in the

DESIGN_CONCEPT table can have one matching record in the Location table (i.e. each building can have be located only in one location).

Each of the CADREPs for the different assembly types in a design concept is stored in a separate table. Since each CADREP can be any one of many different CADREP types (i.e. surface, box solid models, L-shaped solid models etc…), a "one_to_many" relationship exists between the CADREP table and CADREP types table.

A template was created for creating new projects, i.e. new DESIGN_CONCEPT entities are created by instancing the template (Figure 5.15.). So for example, in EASYBUILD the user creates a new project and is asked to fill in the explicit attributes. The various derived attributes, and criteria are calculated automatically after the explicit user input attributes are defined. Also the constraints are checked and the applicability of the assembly constructions is verified as described above. The graphical derived attributes are automatically extracted from the stored in the cad (in EASYBUILD these graphical attributes are stored as Xdata of the entities in the AutoCad) file. Derived relational attributes (pointers to other entities) can also be extracted automatically from the cad file.

Design_Concept Schema Tables (relations)
Design concept
Locations
Construction types (type classification)
Building types (codes)
CADREPS: External Walls, Internal Walls, Roofs, Floors, Slab_on_grade, Foundations, Columns, Windows, Doors
CADREP TYPES
Site
Zones
Spaces (space types)

Table 5.5. Tables in the DESIGN_CONCEPT schema

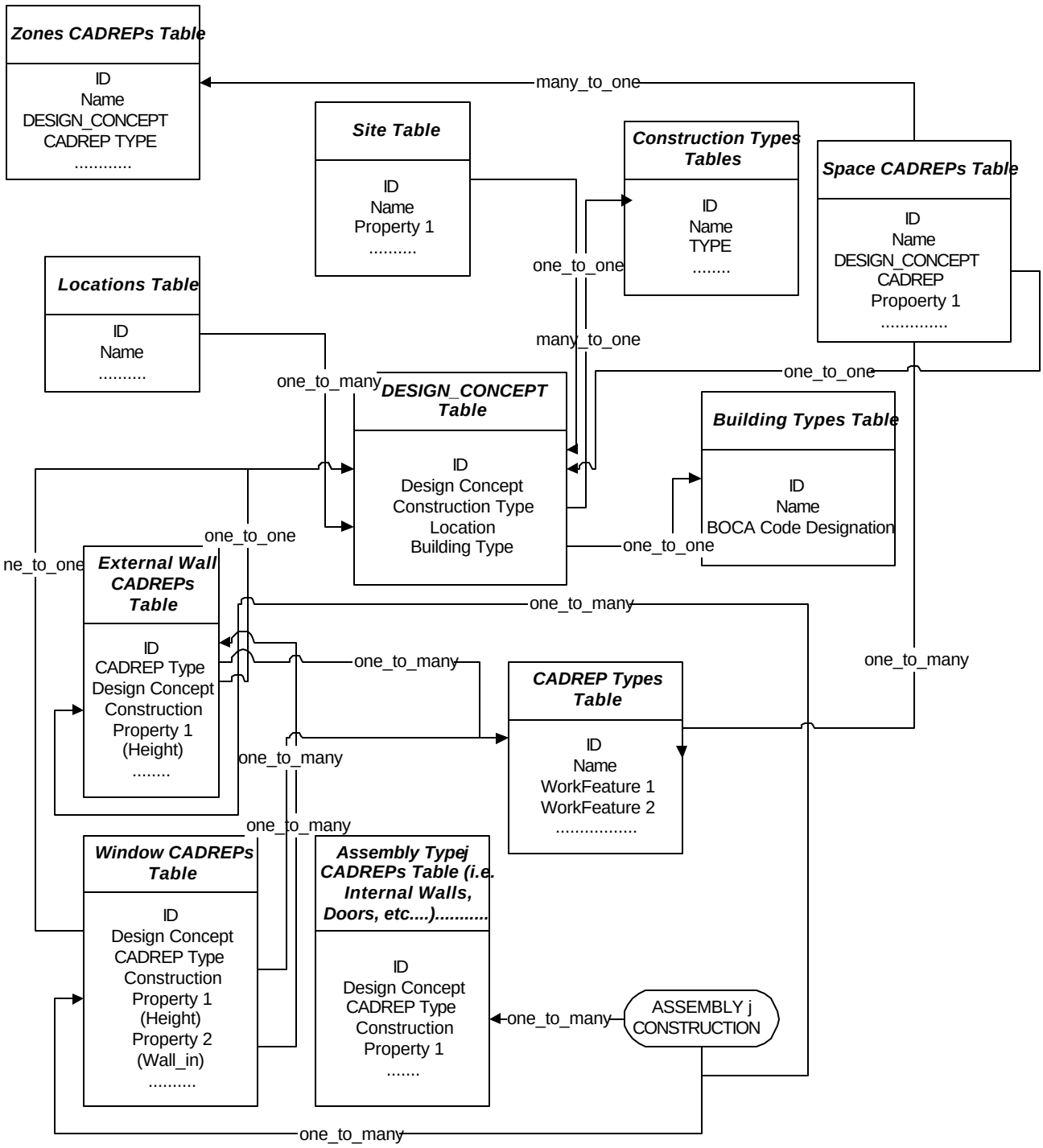


Figure 5.14. The Relations between the DESIGN_CONCEPT tables

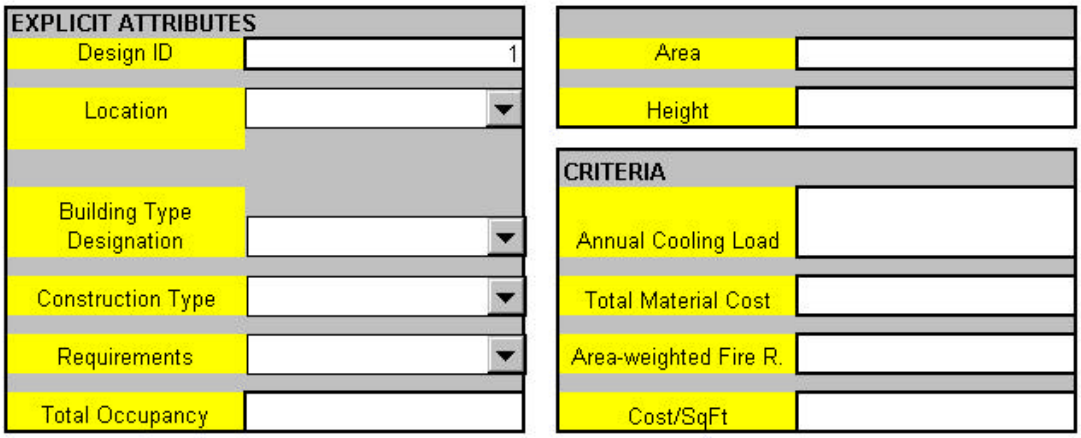


Figure 5.15. The DESIGN_CONCEPT template

5.4.3.2. ASSEMBLY Schema

The assembly schema is the schema that holds the definitions of the various building assembly types and constructions. Each of the assembly construction entities defined in the ASSEMBLY schema can have its own attributes constraints and, criteria as described above.

MASTERFORMAT	MCACES	UNIFORMAT
01 General Conditions	00 General Conditions	01 Foundations
02 Site Work	01 Substructure	02 Substructure
03 Concrete	02 Structural Frame	03 Superstructure
04 Masonry	03 Roofing	04 Exterior Closure
05 Metals	04 Exterior Closure	05 Roofing
06 Wood and Plastics	05 Interior Construction	06 Interior Construction
07 Moisture Thermal Control	06 Interior Finishes	07 Conveying
	08 Specialties	08 Mechanical
08 Doors Windows, and Glass	09 Plumbing	09 Electrical
	10 HVAC	10 General Conditions
09 Finishes	10 Special Mechanical	11 Specialties
10 Specialties	11 Interior Electrical	12 Site Work
11 Equipment	12 Special Interior Electrical	
12 Furnishings	13 Equipment and Conveying	
13 Special Construction	14 Site Preparation	
14 Conveying Systems	15 Site Improvements	
15 Mechanical	16 Site Utilities	
16 Electrical		

Table 5.6. Building Breakdown Structures

Table 5.6. shows 3 breakdown structures of buildings. These breakdown structures are usually used for tasks like specifications, construction contracts and, cost control. The assembly breakdown structure used in EASYBUILD is defined so that designers can use the assemblies as elements for their design. The assembly types identified in EASYBUILD are:

ID	Assembly Type Name	Generic Type
1	Foundation	ST
2	Beam	ST
3	Column	ST
4	Slab	ST
5	External Bearing Wall	ST
6	Internal Bearing Wall	ST
7	External Wall	NONST
8	Ballustrade	NONST
9	Balcony	NONST
10	External Stairs	NONST
11	External Ramp	NONST
12	Internal Stairs	NONST
13	Exposed External Floor	NONST
14	Roof	NONST
15	Partition Wall	NONST
16	Internal Floor	NONST
17	Exposed Internal Ramp	NONST
18	Excavation	NONST
19	Window	NONST
20	Door	NONST
21	Slab on Grade	ST
22	Window	NONST

5.4.3.2.1. EXRESS-G Model

We define a building assembly as a collection of one or more construction components (e.g. sheathing, bricks, flashing etc…) that are joined together to form a part of the building. The most important aspect of an assembly is that it can be used as a distinct drawing element in the representation of the building. Building assemblies are analogous to what architects call design vocabulary.

EASYBUILD currently supports the flat plate construction type. The nine assembly types used in the flat plate construction type are modeled (Figure 5.16.): foundation, column, slab (floor), roof, external wall, internal partition, window, door and, slab on grade. In Figure 5.16. we present the attributes, and relations for an example assembly type: the external wall assembly. The rest of the assemblies are defined in appendix.

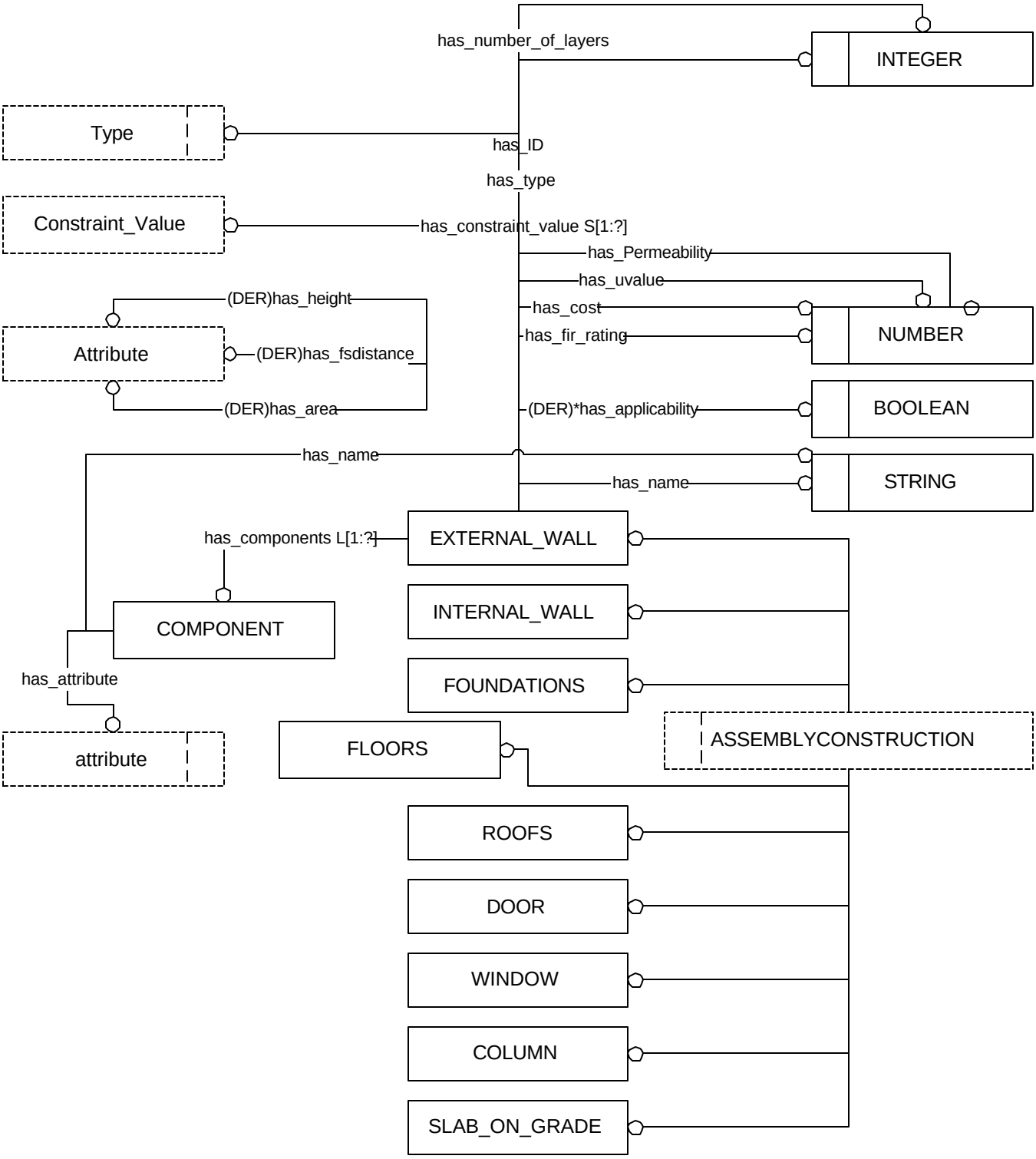


Figure 5.16. The ASSEMBLY entities

Attributes of the external wall entity include cost, replacements, weight, Uvalue, decrement factor and thickness. Examples of external wall assembly instances would be: "35/8in light weight hollow cement block painted both sides" or "3in cinder block, 5/8 in plaster both sides".

Some attributes, constraints and, procedures are generic (inherited from the wall assembly entity) and apply to all assembly constructions. However, each new assembly construction can be defined with its own set of attributes, constraints and procedures. For example, each assembly construction has a set of components. Components also have CADREPs.

5.4.3.2.2. Database Implementation

Similar to the DESIGN_CONCEPT schema, the ASSEMBLY schema was implemented as a database file with several table shown in Table 5.7. The various tables store the assembly constructions for the different assembly types. Also the components that make up the assembly constructions are stored in separate database table. The relational database structure allows us to store the information about the components only once. Then, for each assembly construction we reference the appropriate components.

ASSEMBLY Schema Tables (relations)
External Wall Assembly Constructions
Internal Wall Assembly Constructions
Roof Wall Assembly Constructions
Floor Wall Assembly Constructions
Slab_on_grade Wall Assembly Constructions
Window Wall Assembly Constructions
Door Assembly Constructions
Column Assembly Constructions
Foundations Wall Assembly Constructions
Components

Table 5.7. The tables in the ASSEMBLY schema

The various relations between the tables in the ASSEMBLY schema is shown in Figure 5.17. There is a "one_to_many" relation between the components used in each assembly construction and the components stored in the components table. The relations are shown for the external wall assembly type, but the relations are the same for the other assembly types.

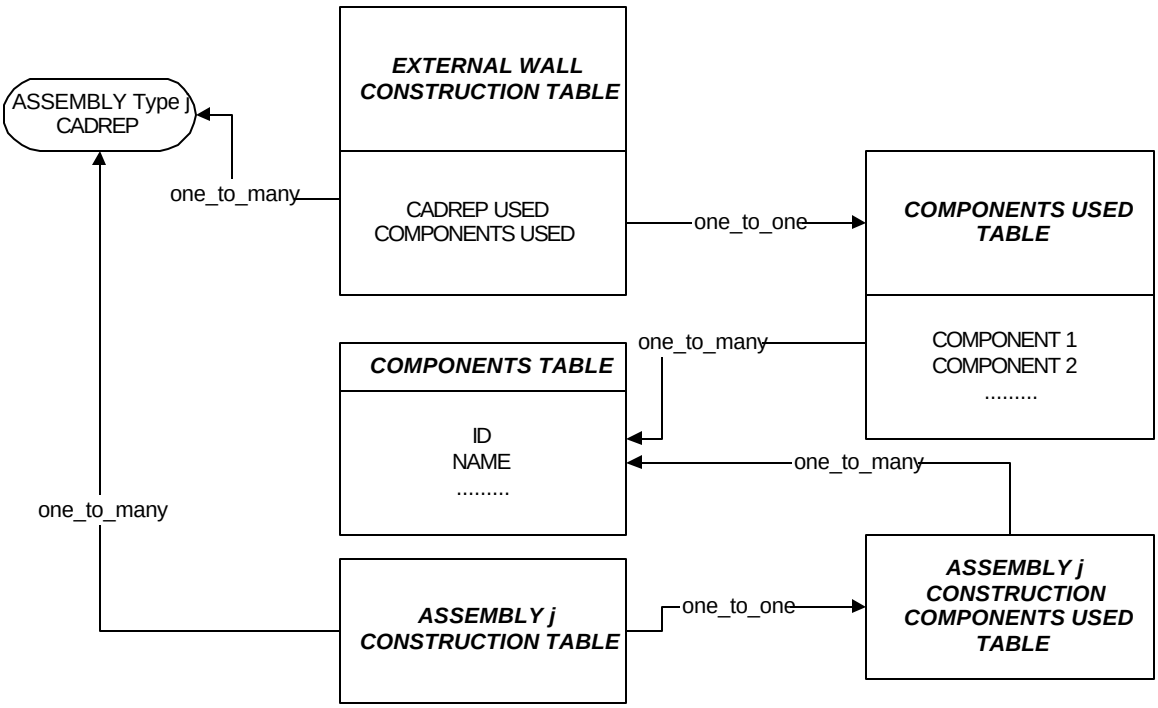


Figure 5.17. The Relations between the ASSEMBLY tables

5.5. SYNOPSIS

This chapter described the database used to store the design description and the various assembly constructions. The product model behind the database was also introduced. Th different steps in developing the database and the product model, were identified. The conceptual breakdown of the building was introduced. This was followed by the description of the formal model and the database. The EASYBUILD system was briefly introduced, along with its different modules.

The database, which is one of the modules of EASYBUILD, includes the criteria and constraints used in the selection procedure. The database also has two schemas: the DESIGN_CONCEPT schema, which store the design description and the ASSEMBLY schema, which stores the different assembly constructions. The kinds of constraints and criteria that can be defined are presented along with examples of the actual criteria and constraints implemented in the EASYBUILD system. The EXPRESS-G model and the database implementation of the two schemas were then presented. In the next chapter we will describe the assembly selection and generation procedures.

CHAPTER 6. ASSEMBLY SELECTION AND GENERATION

One of the most important tasks in the design development stage is the selection of the appropriate assembly constructions for the various building assemblies. Therefor, in order to automate the detail stage a procedure must be identified to automate the selection. Once the best assembly construction is selected the next task becomes to generate the 3D solid model of the assembly for the specified design concept.

This chapter introduces the assembly selection and generation procedure (Figure 6.1.). In the first part of the chapter, we will begin by introducing the assembly selection procedure and the various steps required. Each of the steps will be described in detail. In the second part of the chapter the assembly generation procedure will be described. Generating the 3D detailed solid of the assembly by parametrically saving the assembly construction is introduced. The different parametric operations required to store the details are described, along with the variables of each operation.

6.1. ASSEMBLY SELECTION

The EASYBUILD database includes a number of examples of assembly constructions for different assembly types. Table 6.1. shows examples of some assembly constructions for the various construction types. The EASYBUILD database has several assembly constructions stored (a complete list can be found in appendix). The goal of the assembly selection procedure is to select which of these assembly constructions for each construction type is best for a particular situation (For example whether to use a “stud-backed veneer” assembly construction versus a “solid CMU wall” assembly construction for the external wall assembly type).

In the next section we will introduce the definition of the best assembly construction and then we will introduce assembly selection problem in section 6.1.2. The steps of the proposed selection procedure are described in the following section.

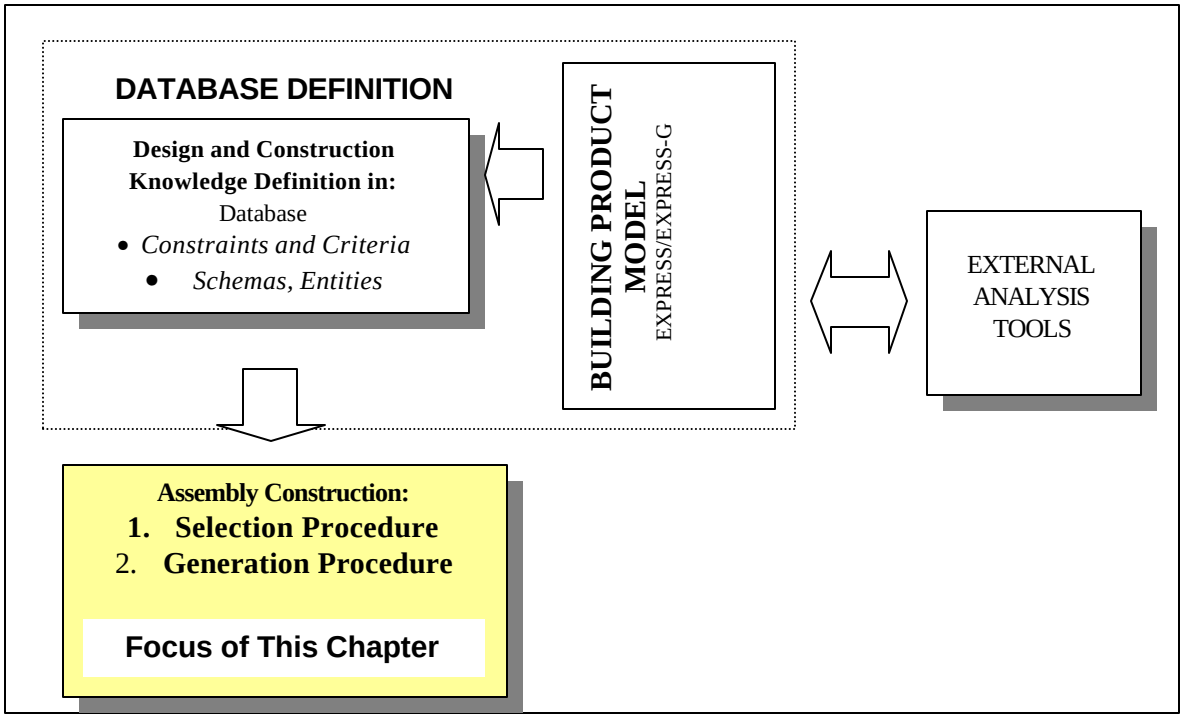


Figure 6.1. The Focus of this Chapter

6.1.1. The selection criteria and the 'best' assembly construction

The definition of best assembly construction is dependent on the designer requirements. There are several criteria that affect the selection of one assembly construction versus the other. We have identified these criteria in the last chapter under two categories: criteria that can affect the building entity as a whole (DESIGN_CONCEPT entity level) like annual thermal load and criteria that affect each assembly type by itself e.g. external wall durability or internal wall sound transmission class. Depending on the designer preference and also the building design situation some of these criteria will be considered while others will not. Furthermore depending on the designer and the building design relative importance for these criteria will be assigned. For example, in one design situation the building initial material cost is of utmost importance while other criteria like the fire resistance or constructability are not of importance at all, while in more critical situations some

performance criteria might become more important that cost e.g. the sound performance characteristics. It is important to find which assemblies are best for a specific design case, but also it is important to find out how the various importance weights of the criteria will affect the final selection.

The goal here is to let the designer select the criteria that define the "best" assembly. Also the designer needs to define relative importance for these criteria and let EASYBUILD determine which assembly constructions are the best for this particular design case. EASYBUILD has procedures and information to calculate some of the criteria and other criteria can be added in the future.

EXTERNAL WALL
15in solid concrete blocks, 0.5 in. plaster on each side
4.5in brick, 0.5 plaster on each side
9in brick, 0.5 plaster on each side
12in brick, 0.5 plaster on each side
35/8in light weight hollow cement block painted both sides
35/8 light weight hollow cement block, 1/4in plywood covered lead on 1x2 wood furing strips one side
3in cinder block, 5/8 in plaster both sides

ROOF
3" RC slab and built-up asphalt roof
3" RC slab and built-up roof, Coal Tar pitch
3" RC slab and Modified Bitumen (APP modifier)

INTERNAL WALL
2x3 staggered wood studs 8 in o.c. two layers 0.5in gypsum board both sides
2x4 wood studs 16 in o.c. 0.5in plaster on 5/8 in. gypsum board on resilient channels both sides.
2x4 wood studs 16 in o.c. 0.5in plaster on 0.5 in. gypsum board laminated to 5/8 in gypsum board on resilient channels both sides.
3.25 in wire studs 16 in. o.c. 7/8 in. plaster on metal lath both sides
3.25 in wire studs 16 in. o.c. 7/8 in. plaster on 3/8 in gypsum lath both sides (one side resilient clips)

Table 6.1. Examples of different building assembly constructions for different assembly types

6.1.2. The assembly selection problem

The selection of the assembly constructions is a challenging task because choosing an assembly construction that is best in terms of one criterion might affect the performance of another criterion. For example, consider the design concept shown in Figure 6.2. The design concept is made up of different assembly types. The goal is to select from the database (i.e. Table 6.1.), the best assemblies that satisfy the user-selected criteria (defined in the last chapter).

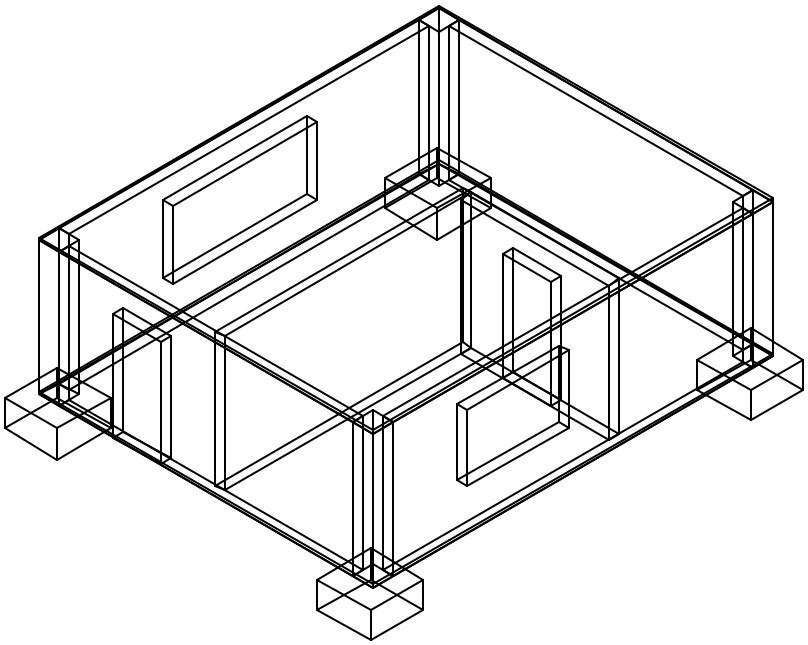


Figure 6.2. The example Design Concept

The best assembly construction is defined as the assembly that optimizes the user-selected criteria. For example, the user-selected criteria for the design concept in Figure 2, can be threefold: the design concept thermal load, the external wall's permeability and, the floor's NIC rating. Selecting the wall construction with the highest permeability is a trivial task by choosing the assembly construction that has the highest value for the permeability attribute. Selecting, the best floor NIC rating is also a similarly trivial task. However, the selected wall and floor assembly constructions might not be the best assemblies in terms of minimizing the design concept's thermal load. There are two problems here.

The first problem is the conflicting criteria. This problem can be overcome by assigning relative weights to the various criteria. Assigning the weights is an important task that can be accomplished in several ways depending on the user preference and is discussed in detail later on.

The second problem is that we have to consider the assembly constructions for the various assembly types as a whole and not only consider the assembly constructions for each assembly type by itself. This means that all the combinations of the assembly constructions have to be considered (in other words *the whole is different than the sum of the parts)*. In the EASYBUILD system there are currently about 50 different external wall assembly constructions, 20 internal wall assembly constructions, 10 different door and window constructions, and 8 roof constructions. If we were to consider all the combinations we would have to consider 800,000 combinations (*50x20x10x10x8 = 800,000*). As the database increases the number of combinations will become significant. In EASYBUILD we use the ASHRAE method for calculating the thermal load, which is a fairly simple procedure that does not require a lot of computer time. If however we were to use an external more sophisticated tool the computer time would be considerable. Furthermore, if more criteria were added the time to perform the selection increases substantially and becomes unmanageable. This problem can be tackled by breaking down the assembly into stages and states as will be described in the selection procedure below. Next we will introduce the steps of the selection procedure.

6.1.3. The Steps of the Selection Procedure

The selection procedure defined here takes the two problems described above in consideration. The assembly selection procedure is shown in Figure 6.3. The selection procedure can be divided in to five main steps:

- Design Concept Definition
- Criteria Selection and Assigning Importance Weights
- Determining Criteria Scores

- Formulating the ‘best’ solution

- Performing the selection.

In the next sections we will describe each in detail.

6.1.3.1. Design Concept Definition:

The first step is the definition of the design concept. The design concept data is stored in the database according to the data structures described in the last chapter. The actual building design is defined using the assembly types CADREPs. Requirements are specified for the various assembly types. For example, for the design concept shown in Figure 6.2. (see page 120), a designer can specify the finishing material of the external wall assemblies.

As soon as a new design concept is defined a set of explicit user defined attributes have to be defined. One of the design concept’s explicit attributes is the construction type as defined in the last chapter. The construction type determines what assembly types are to be used (e.g. a masonry wall-bearing construction does not have a column assembly type).

After all the design concept assemblies and their attributes are defined, the corresponding template is updated, and the constraints are evaluated to determine the applicability field. We have described two types of constraints last chapter: Generic constraints and assembly-specific constraints. The generic constraints apply to all the assembly constructions while assembly-specific constraints only deal with specific assembly constructions. The various constraints are evaluated and the assemblies that do not violate any of the constraints are added to an applicable assembly list. The violated constraints are output to a text file.

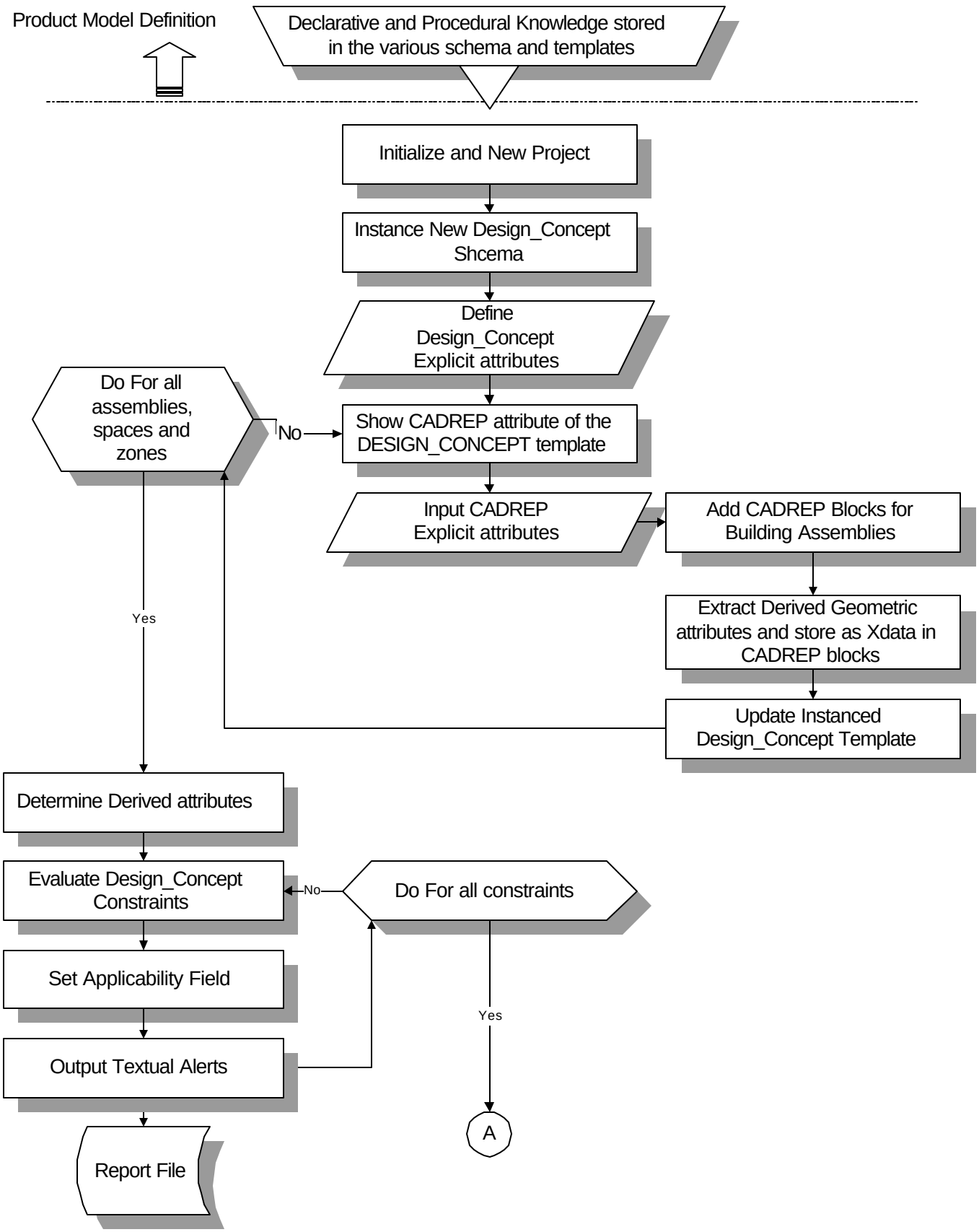


Figure 6.3. The Selection Procedure (A)

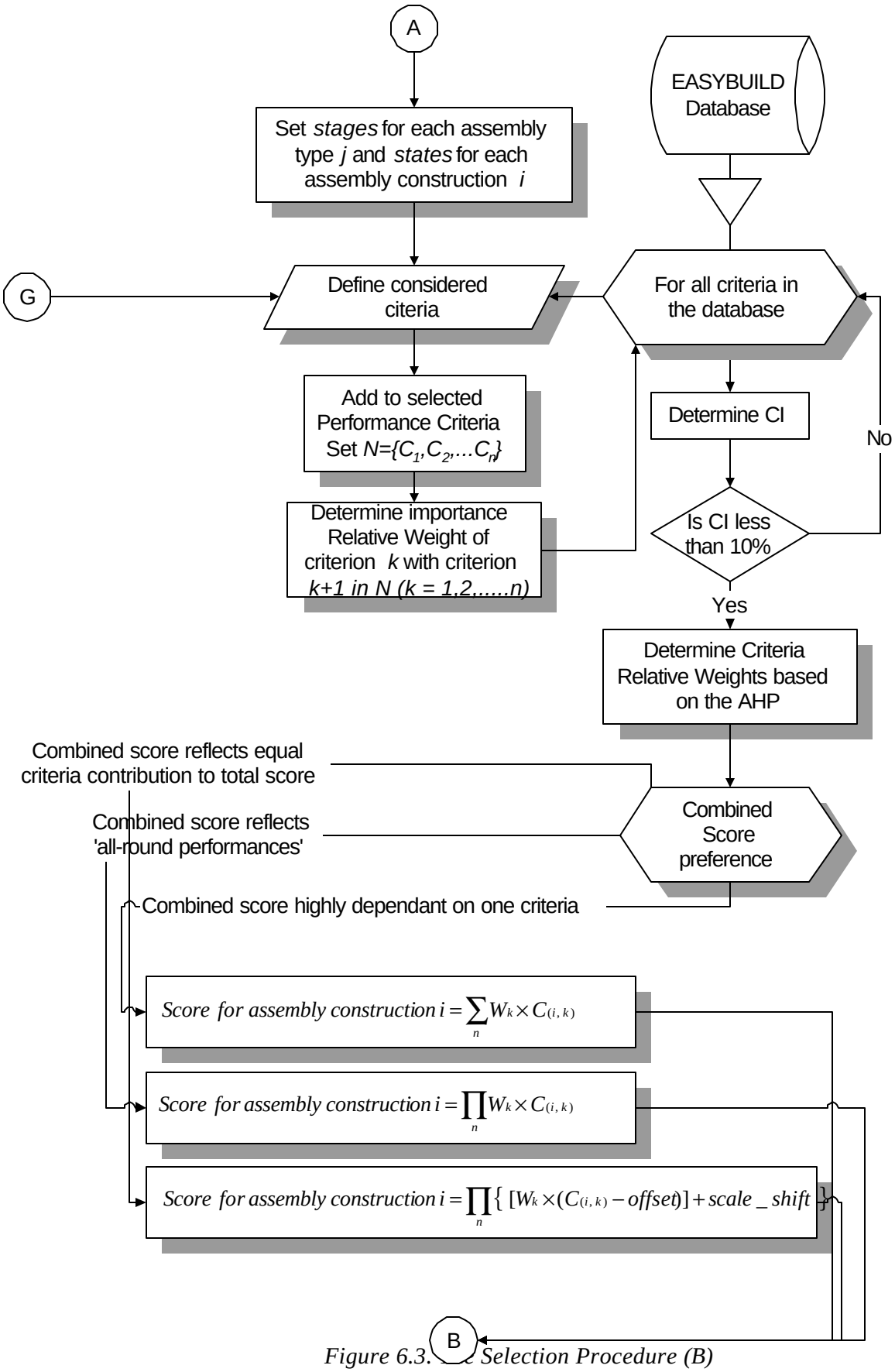


Figure 6.3. [illegible] Selection Procedure (B)

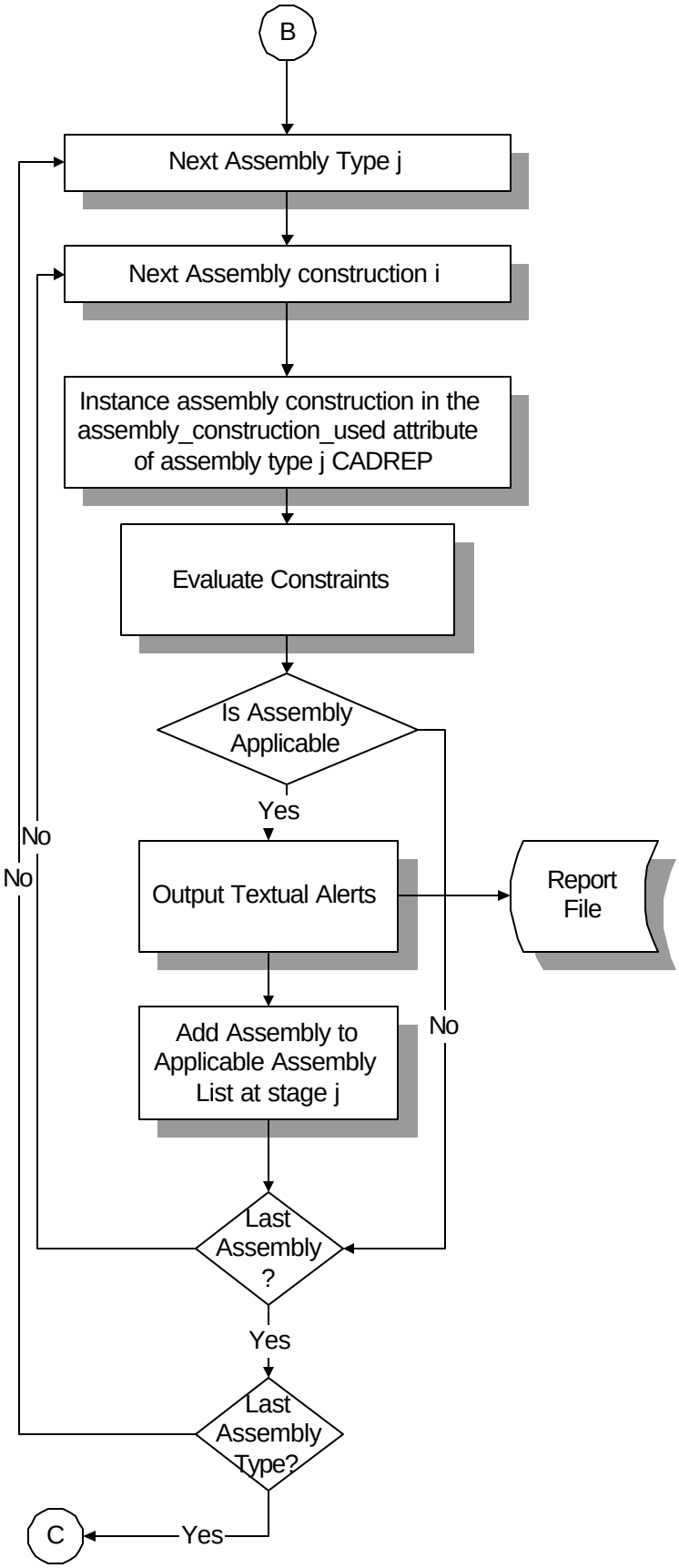


Figure 6.3. The Selection Procedure (C)

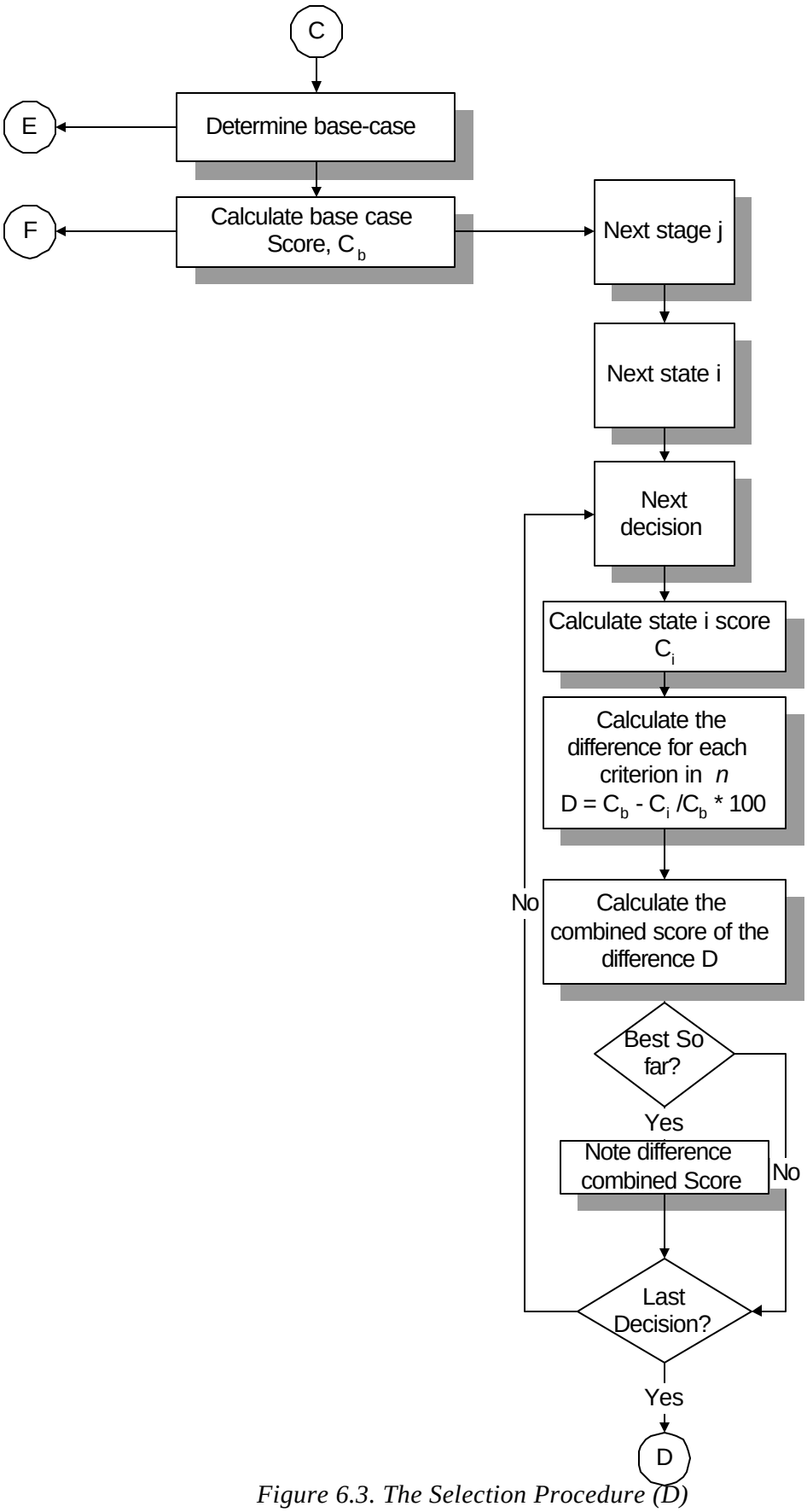


Figure 6.3. The Selection Procedure (D)

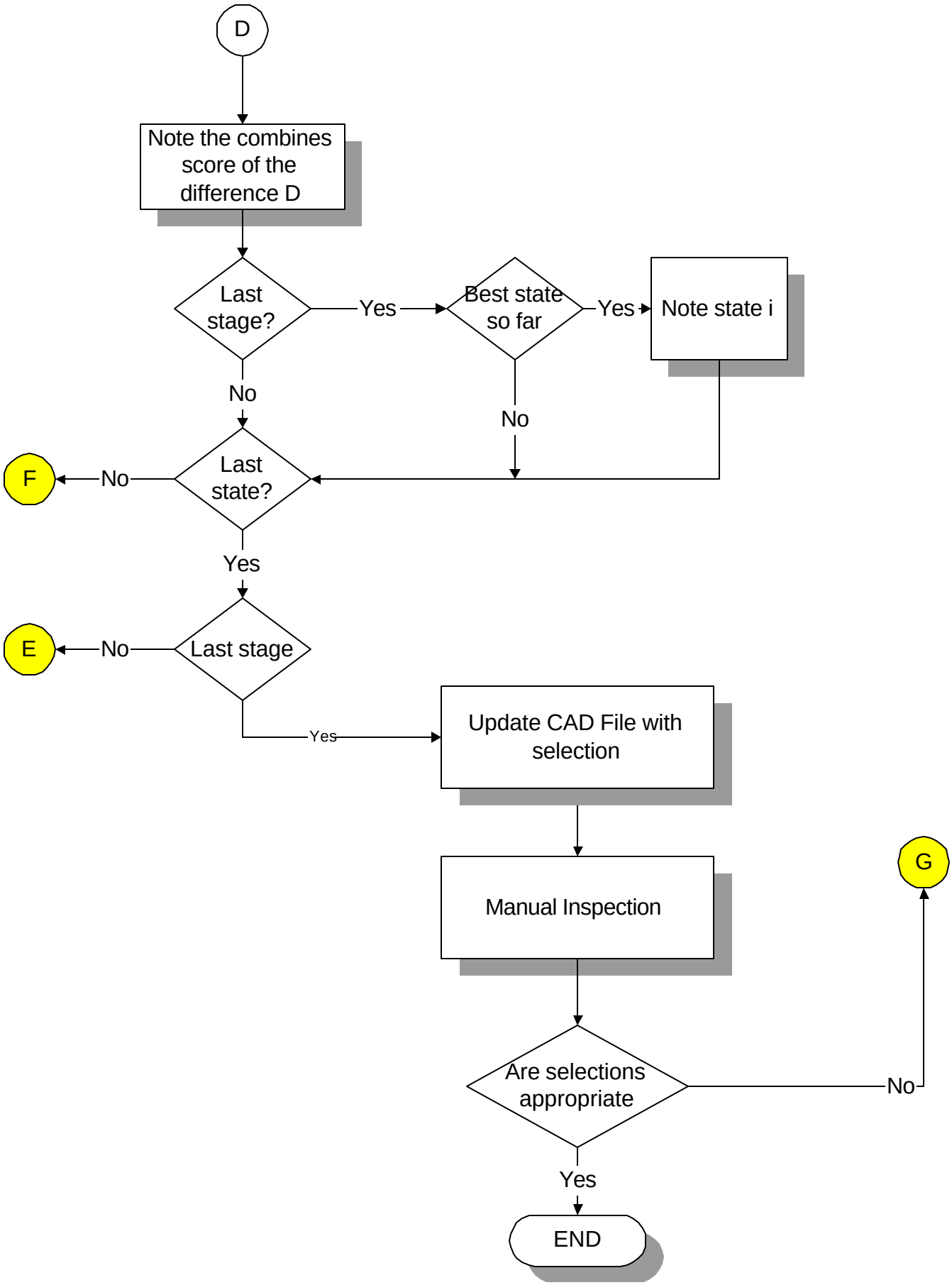


Figure 6.3. The Selection Procedure (E)

6.1.3.2. Criteria Selection and Assigning Importance Weights:

The next step after defining the design concept is to select the criteria that are important for this particular design case. First the selection problem is formulated as a set of stages with different states in each stage. The stages correspond to the assembly types, which in turn depends on the DESIGN_CONCEPT's construction type. The number of stages is equal to the number of assembly types that have to be selected *(j)*. For example in EASYBUILD's flat plate construction there are nine stages corresponding to the nine assembly types, as shown in Figure 6.4. Figure 6.4. also shows that at each stage there are different states. The states correspond to the number of assembly construction options that we can select from for each assembly type *(i)*. The goal is to find the set of assembly constructions *(ij)* that best fit the selected criteria. This is similar to finding the best path in the graph shown in Figure 6.4.

Before we find the best path (described in the 'performing the selection' section) we must specify the criteria and their importance weights so that we know what 'best' is.

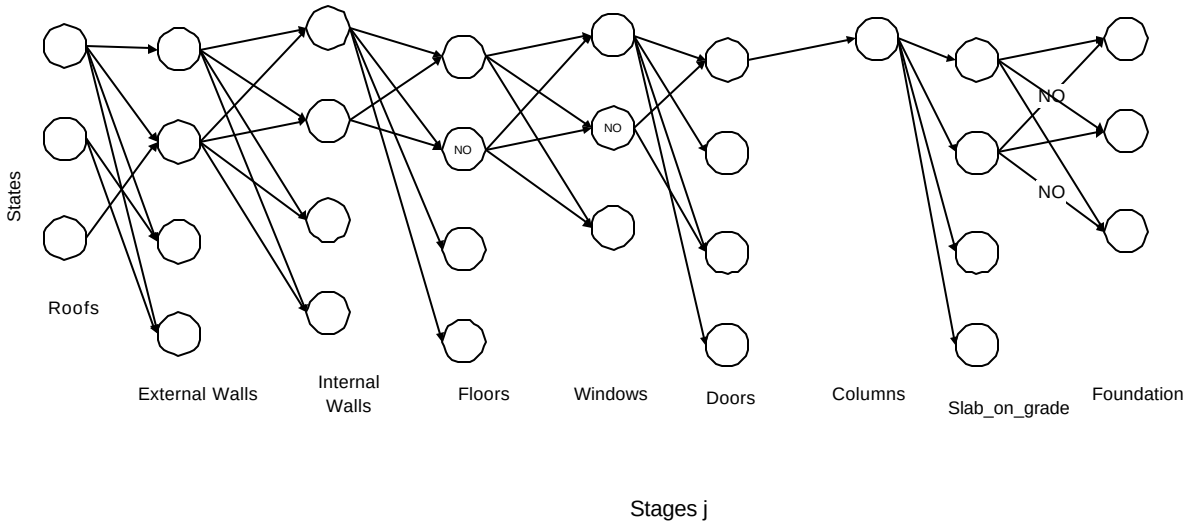


Figure 6.4. The selection problem formulated as stages and states

The selection of the criteria is dependent on the design situation. The designer selects, from the list of defined criteria (described last chapter, section 5.4.2.1), the set of criteria *(N)* to be considered for the design concept (this process can also be automated as will be described in the next section). The designer can perform a what-if analysis by changing the selected criteria set and then rerunning the selection procedure.

The user also is asked to assign importance weights to each of the selected criteria. When assigning importance weights to the criteria one can define the best solution or the objective function by giving the various criteria a percentage weight. For example, external wall construction permeability 20%, annual thermal load 20%, roof construction compatibility 20% and, material cost 40%. Some of these criteria can be further broken down into sub-criteria. For example the roof compatibility can be broken down into roof-traffic compatibility (whether high traffic is expected), CADREP compatibility (suitability with the shape of the roof, i.e. several corners and sharp angles) and building height compatibility (suitability with high-rise use). Or, for example to measure the durability of the external wall durability one might want to consider the wall's surface weight, the used CMU block's weathering index, and the finishing material type. Furthermore, it might be important to define the importance weights for the criteria by considering the importance of one criterion over the other in a pair-wise comparison versus a comparison of all the criteria together. In EASYBUILD the criteria and the sub-criteria weights are considered using the Analytical Hierarchy Process (AHP), [Hastak 1998]. The AHP was recently used for selecting whether to use advanced automation or conventional construction processes [Hastak 1998]. In EASYBUILD the AHP is utilized to define the importance weights for the various criteria.

The AHP establishes a hierarchy of criteria and facilitates the generation of relative importance weights. Establishing a hierarchy in a decision problem is important in order to properly account for the various performance criteria involved in the decision making process and to establish the interdependencies between these criteria1. The AHP determines the preference order among various alternatives, by setting up a hierarchy of criteria and sub-criteria. It is important to note that all criteria in one level of the hierarchy do not necessarily have to relate to all the sub-criteria in the next lower level of the hierarchy.

The AHP is based on matrix computations and pair-wise comparison of the various criteria. The matrix $\mathbf{A = (a_{kl})}$ is evaluated by comparing each criterion k with the other criterion l of the set of considered criteria $N\ (k, l = 1, 2, \ldots, n)$.

$$A = (a_{kl}) = \begin{array}{c} \\ C1 \\ C2 \\ C3 \end{array} \begin{array}{c} \begin{array}{ccc} C1 & C2 & C3 \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 1/a_{12} & a_{22} & a_{23} \\ 1/a_{13} & 1/a_{23} & a_{33} \end{bmatrix} \end{array}$$

The importance of one criterion over the other is determined by utilizing a preference scale shown in Table 6.2. Therefore the entry a_{kl} in the matrix is the relative importance of criterion k over criterion l,

$$a_{kl} = \frac{W_k}{W_l}$$

All the entries in the matrix are positive, and satisfy the condition for a reciprocal matrix, $a_{kl} = \frac{1}{a_{lk}}$ [Hastak 1998]. The weight vector WA is defined as the eigen vector corresponding to the max. eigen value of A,

$$WA = \begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix}$$

For example consider the four following criteria,

N= {C_1= External walls permeability, C_2= Annual thermal load, C_3= Roof compatibility, C_4= Initial material cost}

If this was the set of criteria to be considered, then our matrix could be,

$$\begin{array}{c} \\ C1 \\ C2 \\ C3 \\ C4 \end{array}\begin{array}{c} \begin{array}{cccc} C1 & C2 & C3 & C4 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 \\ 0.5 & 1 & 1 & 2 \\ 0.5 & 0.5 & 1 & 2 \\ 0.25 & 0.5 & 0.5 & 1 \end{bmatrix} \end{array}$$

For example, the evaluation of element $a_{14} = 4$ by the designer means that the designer expresses that the external wall permeability has a strong importance over the initial material cost (for example due to the existence of the design concept in a high risk of water migration location).

The upper triangle of the matrix is determined by the relative comparisons of the various criteria. The lower triangle of the matrix is determined by taking the reciprocal of the corresponding elements from the upper triangle. The corresponding weight vector, which is established by computing the eigen vector (computing eigen vectors is described in appendix), corresponding to the highest eigen value of the comparison matrix above, would be,

$$WA = \begin{bmatrix} 0.44 \\ 0.22 \\ 0.22 \\ 0.11 \end{bmatrix}$$

The relative weight vector *WA* is developed for the upper level of a criteria hierarchy. Each of the criteria in *N* might be decomposed into a set of sub-criteria. Using the AHP, these sub-criteria can be accounted for by setting up a similar vector for the sub-criteria of each criterion. The AHP does not necessarily require that each criterion to be decomposed into sub-criteria. For example for the set *N* above, *C3* can be decomposed into *C3.1* = roof-traffic compatibility, *C3.2* = CADREP compatibility, *C3.3* = steel deck compatibility, and *C3.4* = building height compatibility. The preference matrix can look something like,

$$= \begin{array}{c} \\ C3.1 \\ C3.2 \\ C3.3 \\ C3.4 \end{array}\begin{array}{c} \begin{array}{cccc} C3.1 & C3.2 & C3.3 & C3.4 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0.5 & 1 & 2 & 3 \\ 0.5 & 0.5 & 1 & 2 \\ 0.25 & 0.33 & 0.5 & 1 \end{bmatrix} \end{array}$$

From this matrix one can see that for the roof type construction compatibility with high rise building is significantly more important than the roof construction compatibility with high rise traffic (for example due to less roof traffic expected).

Similar to the criteria, we then want to determine the relative weight vector by finding the eigen vector corresponding to the highest eigen value of the sub-criteria comparison matrix. For example, the relative weight vector for the matrix above would be,

$$= \begin{bmatrix} 0.47 \\ 0.28 \\ 0.16 \\ 0.10 \end{bmatrix}$$

Each of sub-criteria can be in turn decomposed into other smaller criteria. At every level of this hierarchy the procedure of pair-wise comparison is carried out between all the sub-criteria of the preceding level. The matrices of the comparisons are evaluated and the priority vectors are established (by finding the eigen vectors). The priority vectors from one level are multiplied by the weight of the corresponding criteria from the preceding level. For example, for the above example the priority vector for criteria *C3* is multiplied by the weight of the criteria *C3* determined form the upper level *(0.22)*. This process is carried out for all the criteria and the levels of the decomposition to determine the weighted priority vectors.

Degree of importance	Definition
1	Equal importance of elements
3	Importance of one element over another
5	Strong importance of one element over another
7	Demonstrated or very strong importance of one element over another
9	Absolute importance of one element over another
2,4,6,8	Intermediate values between two adjacent degrees of importance

Table 6.2. Comparison Scale adapted from [Saaty 1982]

The resulting weighed priority vectors are added up to compute the aggregate vectors. The aggregate vectors are then used as rows of an aggregate matrix. The weights of the various criteria are established by adding the column entries of the aggregate matrix.

It is important to maintain the consistency of the developed matrices for the pair-wise comparison of the criteria and sub-criterion. A small deviation in the consistency of the matrices is allowed in the AHP. The consistency of the matrices can be determined by,

$$Consistency\ Ratio = (CR) = \frac{CI}{RCI}$$

$$Consistency\ Index = (CI) = \frac{(l_{\max} - n)}{n-1}$$

The Random Consistency Index (RCI) is the average for a large number of reciprocal matrices of the same order with random entries and λ_{max} is the maximum eigen value of the matrix [Hastak 1998]. Empirical studies shows that a deviation in the consistency ration CI of about 10% is acceptable. However, if the consistency ratio is more than that 10% then the values in the comparison matrix have to be determined again (one designer can choose to delete a criterion, while another will choose to change the assigned pair-wise weights).

Each of roof assembly constructions can be assigned a score for the criteria described above. Experienced professionals or professional publications can be used to determine these scores (in EASYBUILD a few examples are given). The relative importance of these criteria will however depend on the designer and the actual building concept under consideration. Using the procedure described above it is possible to systematically assign the relative weights.

The preference scale used here is [Saaty 1982] is a relative scale that is useful for subjective measurements where the goal is to define a priority among various criteria. The AHP method has been praised [Hastak 1998], [Saaty 1982] for facilitating exact analysis of criteria and producing more accurate importance weights. The decision model reflects the perception and judgment of the designer through a structured, systematic and careful analysis. The detailed example given in

the next chapter will provide more insight about this step of the assembly selection procedure. It is important to note that the importance weight vector can be saved so that it can be used for similar design situations. Now lets consider how the actual scores for the criteria can be calculated for the different criteria types, and how we can automate the selection of the applicable criteria for a given design situation.

6.1.3.3. Determining Criteria Scores

The different criteria score can be determined in different ways. Some criteria values are static and can therefore be retrieved from the database. For example the permeability of the external walls is stored as a value with each external wall assembly construction. Other criteria can be calculated by certain procedures. For example the annual cooling load is calculated by the ASHRAE CLTD/CLF method. On the other hand some criteria do not have a fixed static value that can be saved in a database and in the same time there is no analytical procedure or formula to calculate the values of the criteria. For example the durability of an external wall construction can not be determined by an analytical formula and also might be dependent on the actual design situation, e.g. the building use, the building location rainfall etc…[17]

For these kinds of criteria we turn to look at how the computer can determine the criteria performance scores automatically by looking at previous building design cases using machine learning. In chapter three, we presented three such methods, neural networks, case-based reasoning and expert system rules. We concluded that the neural network output is not always stable. Furthermore the output of neural networks is usually characterized as coming from a black box. This usually means that no stated rules of the actual patterns or conditions found in the data is explicitly stated.

On the other hand, case-based reasoning techniques are found to be archival methods that can retrieve previous cases, which are most similar to the existing cases. This means that case-based

[17] Although sometimes in texts and practice manuals, the durability of the various wall constructions are ranked beforehand and can be saved as static criteria in a database

reasoning can not also generate explicit rules, which is stressed to be important when determining the criteria performance scores for building assemblies. The ability to be able to define explicitly the patterns and rules in the previous cases as rules has been emphasized in previous studies as was mentioned in chapter three (see).

Therefore we use rule induction as a technique to learn from previous cases. Rule induction is a technique that involves generating a general rule of the type:

IF <general circunstance> Then <general conclusion>

from previous specific examples. The aim is to build rules, which are successful as often as possible, and to quickly modify them when they are found to be wrong. This implies that whatever is being learnt, should match the positive examples but not the negative ones. The process is in essence similar to what a human analyst would do in exploratory analysis of a case history. Rule induction can discover very general rules, which deal with both numeric and non-numeric data. And rules can combine conditional and affinity statements into hybrid patterns.

To use this approach two tasks are required; firstly a database design and its records for previous cases which require the building to be broken down into different entities and their attributes. The building product model discussed in the chapter five forms the database design used to store the previous cases.

The second required task is a rule induction algorithm that will extract the rules from previous case histories. We use a tool that looks at previous cases to try to predict the current criteria score. WhizWhy is a tool that is based on a proprietary approach that can find rules in databases and predict values of certain fields. This tool is integrated with the EASYBUILD system. The approach used is similar to case based reasoning (described earlier in chapter 3) and neural networks, in that the approach is based on case histories. However, we said that neural networks are influenced by subjective choices made by the user during the learning phase, such as the topology of the network, the size of the sample and the order in which the cases are submitted. For example, if the operator selects too few layers for the network topology, relevant patterns in the data may not be revealed; if the network contains too many layers, a non-applicable pattern may be generated. Furthermore, neural networks are generally limited to binary input for dealing with non-parametric data. Also, unlike the case based reasoning approach described in chapter 3,

the approach used here has the ability to provide comprehensive and clearly-stated rules as was stated earlier.

Previous case histories of design concepts are stored in a database in a certain format to allow for the prediction of the criteria value. An example was developed, where 100 hypothetical cases were stored in a database to try to determine the durability of an external wall assembly construction.

Building.Use	Building Location.Rainfall	Cavity or Solid	Wall Assembly Construction	Wall Durability after 10 years
Single Family	3	Cavity	A1	Excellent (10)
Multi Family	5	Solid	A12	Fair (5)
Single Family	8	Cavity	A3	Good (6)
Single Family	4	Solid	A11	Poor (4)
Multi Family	7	Cavity	A24	Excellent (10)

The cases were stored in a database with the following fields;

The wall durability filed is chosen to be the field to predict. This field is also referred to as the "dependent variable while the other fields are "independent variables" or "conditions". The database is first read. The rules that relate between the Field to Predict and the other fields are listed. The rules are formulated as "if-then" sentences or mathematical formulas. For example the following rule was generated by the system for the cases shown above,

If **CAVITY_OR_** *is* **Solid**
And
RAINFALL < 10
Then
WALL_DURAB *is* **Excellent**
Rule's probability: **0.627**
The rule exists in **63** *records.*
Significance Level: Error probability < 0.2

This rule states that if the external wall is a cavity wall and the rainfall in the building's location is less than then ten inches, excellent durability is expected for the wall construction. The durability rating here ("excellent") can be easily converted to a numerical value.

A significance level is given for each rule. The Significance Level is equal to 1 minus the error probability. It may be interpreted as designating the extent to which the rule is reliable for prediction. The error probability designates the probability of discovering "by chance" the rule under discussion.

Significance level = 1 – Error probability

On the basis of the rules discovered, the system can make predictions about new cases; for instance, given the data above the system can predict future values of the wall durability. The system can make predictions based on information entered manually by the designer or information in a separate file.

The system can also translate the rules it finds into an SQL (Standard Query Language) query. This feature enables you to select records in any database that meet the conditions in the specified rules. One can also "fine-tune" the analysis by defining parameters such as "Minimum Probability of the rules" and "Minimum number of cases in each rule." The system follows these "instructions" when issuing the rules. Of course the cases here are hypothetical but the same procedure would apply for actual previous cases once several DESIGN_CONCEPT records are added to the EASYBUILD system. The same approach can also be used to determine which criteria are applicable for a particular design case.

Notice that we use this technique here to help us in determining the criteria performance scores from previous case histories and not to perform the actual construction assembly selection. Using machine learning techniques that learn from previous case histories to automatically select the appropriate construction assemblies is discussed in more detail in chapter eight, under recommendations for future work.

6.1.3.4. Formulating the 'best' solution:

Now that we have the relative importance weights of each criterion and can determine the score of each criterion, we want to define the 'best' solution or formulate the scoring method to calculate the score for each assembly construction. Each criterion will have a different unit for scoring. Therefore we need first to normalize the scores of the various criteria so that we can add them up in a combined score.

Normalizing the criteria score in EASYBUILD is done by first determining the maximum and lower criteria scores. Then a normalized scale *(X_k -100)* is set up between the maximum and minimum scores to determine the actual score of each criterion. By mapping the raw criteria score to the scale it is possible to determine a normalized score.

$$C_{ik} = \frac{C_{ik_{raw}} - (C_{k\,\min} - o_k)}{(C_{k\,\max} + o_k) - C_{k\,\min}} \times 100$$

where, C_{ik} is the normalized criteria score of assembly construction *i* for criteria *k*, $C_{ikactual}$ is the raw score of assembly construction *i* for criteria *k*, C_{kmin} and C_{kmax} are the minimum and maximum scores for criteria *k* and o_k is the offset that determines the lowest normalized value *X* of the criteria *k*. The value of *o* is determined in EASYBUILD so that the lowest normalized value X_k is usually 10%, and can be changed by the designer. This provides a sensitivity measure to determine the criteria scores for the various assembly constructions.

For example the lowest STC rating for a wall assembly construction in the EASYBUILD is C_{min} *= 36* and the highest is C_{max} *= 66*, and for an assembly construction *i* whose STC value Ci_{min} = 51, the normalized value (with o_k = 3) would be = 54.55 = *51-(36-3)/((66+3)-30) * 100*. Figure 6.5. shows a plot of the raw STC scale and the normalized scale.

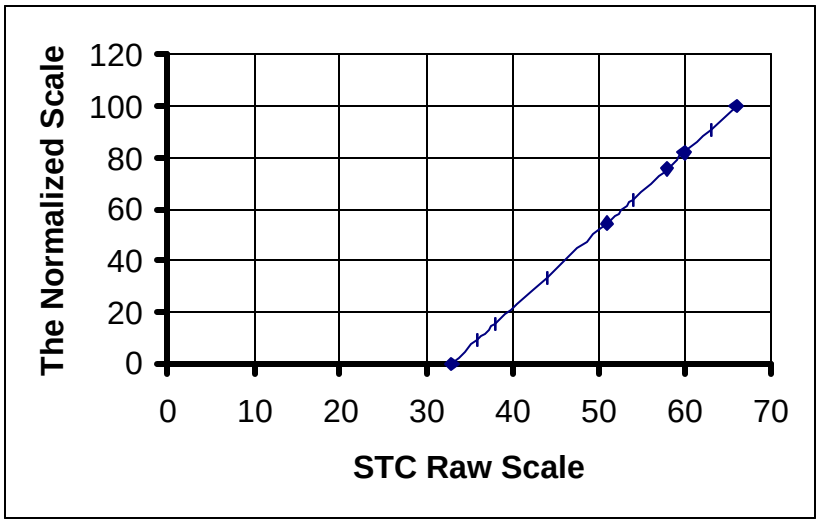


Figure 6.5. The STC Raw Scale versus the Normalized Score

It is important to note that the scale described above for the dependent criteria score is a linear scale. However certain criteria may have changes that do not follow that linear scale. In other word the changes in the criteria score have different significance at various values. In this case a sliding scale percentage or a nonlinear scale could be used. In a nonlinear scale incremental changes at the start of the raw scale reflect larger value changes in the normalized scale than similar incremental changes at the end of the raw scale. Also, if the raw scale values need to be normalized to that a minimum is sought, we could use the same method described above, with one modification. The gradient of the normalized scale would have to be negative, and therefore the higher the raw scale the lower the normalized scale.

The procedures described above allow us to normalize the various criteria to a normalized scale. Given the weight of the criteria and the normalized score of each assembly construction with respect to each criterion, we want to determine the combined performance score of each assembly construction. If we have many criteria to consider, any solution (the set of selected assembly constructions in our case) will likely be a non-dominating solution. A non-dominating solution (sometimes also called a Pareto Solution) is one for which no other solution exists that is

capable of providing a better performance with respect to one criterion and no worse performance with respect to all other criteria [Nassar 1995].

Given the set of criteria *N*, which define the various goals of the selection, the best solution must lie within a set of non-dominating solutions. The designer's choice of which solution in that non-dominating set of solutions is actually the best solution is dependent on the designer's preference. The designer's preference is expressed as a tradeoff of performances in conflicting criteria. This tradeoff is assigned in terms of relative the importance weights described in the section above. The solution that the designer chooses from the set of non-dominating solutions is sometimes called the *best compromise* solution, because it is almost always a compromise between the performance of one criterion versus another.

In EASYBUILD, combining the score of each criterion and its relative importance weight to form a combined score can be done using one of three different methods, depending on the designer preference. Next we will describe the three methods and their applications.

One method to determine the total score of an assembly construction is to multiply the criterion weight W_k by the performance score C_{ik} of the assembly construction i for all criteria n. The then add them up to form the combined score for the assembly construction i.

$$Score\ \ for\ assembly\ construction\ i = \sum_{n} W_k \times C_{(i,\,k)}$$

The assembly constructions that yield the highest combined score are recommended to the designer. This method was used in a previous research by the author [Nassar 1995] to optimize the form of the building (shape, height, length, aspect ratio etc…) with respect to several environmental criteria. However, the summation method is good when one criterion is to have a dominating effect over the other. This is demonstrated when one assembly construction yields a completely inappropriate score with respect to one criterion, which has a low relative importance weight (calculated from the previous step). This assembly construction can still be placed in a higher order and may still be selected if it performs well with respect to a criterion with a higher relative weight.

The second method is to combine the scores and weights of the various criteria by multiplication, instead of addition. So the weight W_k of criteria score C_{ik} for assembly construction i are multiplied for each n criteria,

$$Score\ \ for\ assembly\ construction\ i = \prod_{n} W_k \times C_{(i,\,k)}$$

When using the multiplication method, the assembly construction's performance score with respect to other criteria less readily compensates a poor performance score for a criterion (for an assembly construction). This means that an assembly construction, which has a bad performance score with respect to one criterion, gets eliminated from the recommended assembly constructions. Using this method an assembly construction which has good all-round performances is more likely to be recommended. For example, consider two assembly constructions A, B have a set of 'scores x weights' values with respect to three criteria (1,2,3) and (2,2,2) respectively. Evaluating the combined score using the first method would yield a combined score of 6, while evaluating the combined score using the multiplication method would yield 8. This shows that the multiplication method will give a higher score to a solution that performs well with respect to all criteria over a solution that performs well with respect to some and poorly with respect to others, i.e. a good "all-round performance" will be recommended.

Both the summation and multiplication methods will over emphasis the weight of the criteria over the value of the criteria score. When a criterion is considered important (i.e. has a high relative weight), and an assembly construction has a high score for this criterion, the total effect on the combined score is large. On the other hand when an assembly construction has a low criterion score for a less important criterion (i.e. has a low relative weight), the total effect on the combined score is the smallest. In some cases, such an assembly construction would should still be recommended, even if the weight of the criterion (in question) has a low relative importance weight.

Therefore, a new third method was developed to overcome this problem. This method has been used in the selection of polymers by [Reid 1980], [Hopgood 1993]. In this dissertation we introduce the use of this technique in the selection of construction assemblies. Using this third method the least appropriate assembly construction is actually the one which has the lowest

performance score for the criteria with the highest relative importance. The combined assembly construction score is still calculated by multiplication but two new terms are added so that the combined score becomes,

$$Score\ \ for\ assembly\ construction\ i = \prod_{n} \{\ [W_k \times (C_{(i,k)} - offset)] + scale\_shift\ \}$$

In this method the *scale_shift* term is the smallest number that will ensure that the combined score of the weight and performance rating, is positive. The *offset* term is used to indicate degrees of undesirability. Performance scores lower that the *offset* value indicate the degree of undesirability of the assembly construction with respect to the particular criterion.

Figure 6.6. shows a plot of the weights, criteria score and the final score calculated using the third method, for the roof maintenance criterion (with an *offset* value of 0.5 and a *scale_shift* value of 0.5). One can see that if a roof assembly construction has a poor maintenance score and the maintenance criteria has a low relative importance weight the calculated weighted combined score is set as the *scale_shift* value. So unlike the second method, the lowest weighed score is when the maintenance score is lowest and the maintenance criterion weight is greatest.

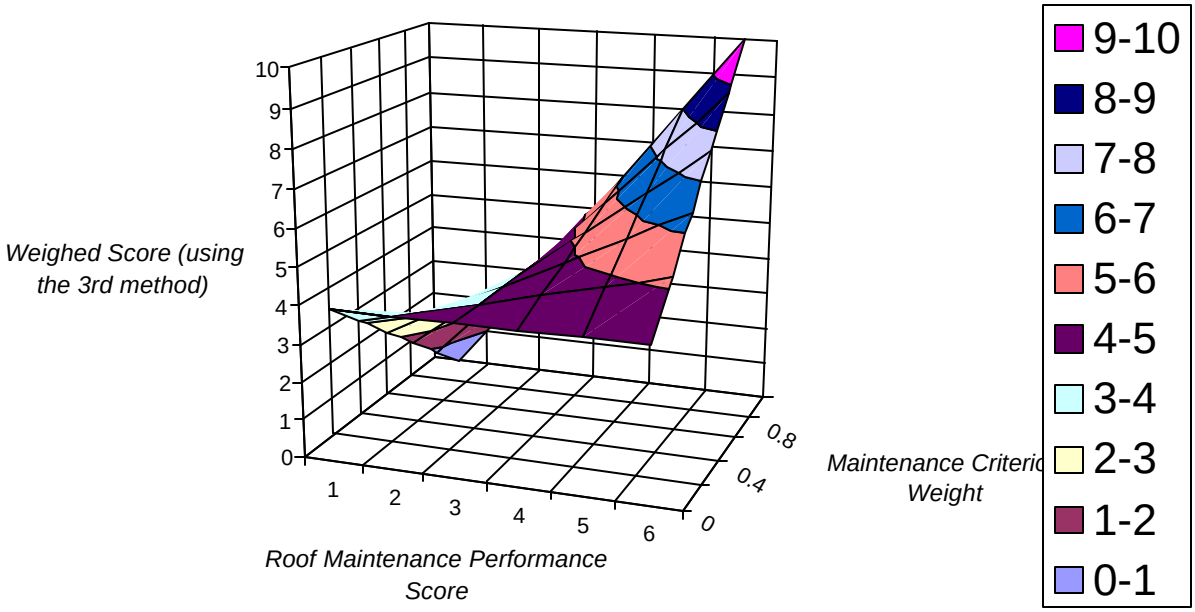
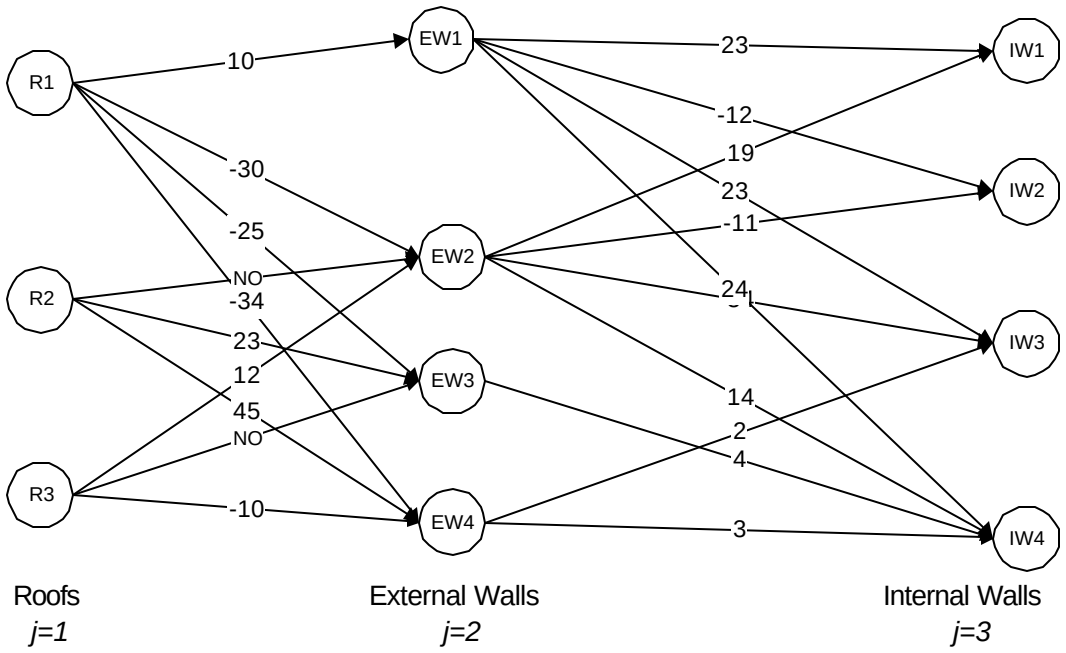


Figure 6.6. The Roof Maintenance Criterion score versus the calculated weighed score

In EASYBUILD the designer can select which method to use to evaluate the combined sore based on the designer preference. If the designer wishes to have the contribution of one criterion to be maximized he chooses the first method, while if a good all-round performance is required the second method is chosen. If the designer wants the least appropriate assembly construction to be the one that has the lowest performance score for the criteria with the highest relative importance, the designer chooses the third method. This gives increased flexibility in the selection procedure.

If the number of assembly constructions is not large, then the AHP can be used to rank the assembly constructions by pair-wise comparison. The assembly constructions would be placed as alternatives in the last level of the decision hierarchy. The last step in the selection procedure shown in figure is performing the selection.

N = {C1, C2, C3}
C1 = Initial material cost, W1 = 0.2
C2 = Annual thermal load, W2 = 0.4
C3 = roof's Ease of maintenance, W3 = 0.4



IW1 = 2x4 wood stud 16in oc, 0.5in gypsum board both sides
IW2 = 3.25 in wire studs 16 in. o.c., 7/8 in. plaster on metal lath both sides
IW3 = 35/8 in metal channel studs, 24 in o.c., two layer 5/8 in. gypsum board both sides

EW1 = 3in hollow gypsum block, 0.5in plaster both sides
EW2 = 4in hollow gypsum block, 0.5in plaster both sides
EW3 = 9in brick, 0.5 plaster on each side
EW4 = Double 4in solid conder block wall, 2in cavity (no wire ties), 0.5in plaster both outsides

R1 = 3" RC slab and built-up roof, Coal Tar pitch
R2 = 3" RC slab and Modified Bitumen (APP modifier)
R3 = 3" RC slab and non-vulcanized elastomers, mechanically attached

Figure 6.7. An excerpt of the three stages of the selection problem

6.1.3.5. Performing the selection:

Now that the criteria weights and the scoring method are established, we can start looking for the best solution. The selection problem has been formulated as a series of stages and states. The stages correspond to the various building assembly types, while the states resemble the various assembly construction options. One can think of route the between the stages and states as a path. The nodes in the path resemble the assembly construction selections for the various assembly types. On the other hand the arcs in the path (the connecting lines) resembles the evaluated combined score. Therefore the goal is to find path in the graph in Figure 6.7. that produces the best combined score.

One approach is to consider all the paths and calculate the combined score for each path and then select the path with the highest combined score. However we have said that this approach is not

feasible due to the large number of paths and performance criteria that have to be evaluated. In EASYBUILD, instead of evaluating all the paths in figure we use an algorithm that minimizes the search.

The algorithm used in EASYBUILD is based on dynamic programming [Radford and S 1988]. The algorithm searches for the best path from one stage to the next. The algorithm also considers the previous stages and their states. To explain the algorithm used in EASYBUILD, we will demonstrate by an example. Consider the first three stages of the assembly construction selection problem with 3,4 and 2 states at the stages respectively. The stages represent the roof construction, the external wall construction and, the internal wall construction. This is only part of the complete assembly selection problem for a flat plate construction type, with nine stages representing the nine assembly constructions in a flat plate construction type. Furthermore, we will limit the applicable assembly constructions at each stage. The assembly construction options, which are represented as states, are shown in Figure 6.7.

Offset	*4.5*			*scale shift*	*0*
Criteria	C1	C2	C3	*Summation*	*Multiplication*
Weights	0.2	0.4	0.4		
Base Case Solution Scores	6	8	10		
Weight*Score	1.2	3.2	4	8.4	15.36
	0.3	1.4	2.2		0.924

Figure 6.8. The Combined Score calculation for the example

First we will define the applicable criteria set *N* as shown in Figure 6.7. Then we will select a base case solution. The base case is an initial set of assembly constructions for the three assembly types used. The combined score of the criteria is calculated for the base case solution using the third method described above. The calculation of combined score for the base case solution is shown in Figure 6.8. At each stage the assembly construction is substituted for the assembly construction used in the base case solution. The percentage increase (or decrease) in the combined criteria score is calculated by comparing the combined score of the base case with the combined score using the assembly construction in the current stage,

$$= \frac{C_{(i,\ j),\ (i,\ j\ +\ 1)} - C_{base}}{C_{base}} \times 100$$

The assembly construction that makes the largest increase in the combined score is considered to be the best assembly construction. The percentage improvements are shown in the arcs connecting the various states in Figure 6.9. To minimize the number of assembly construction evaluation we first consider the arcs with the maximum values using backward tracking.

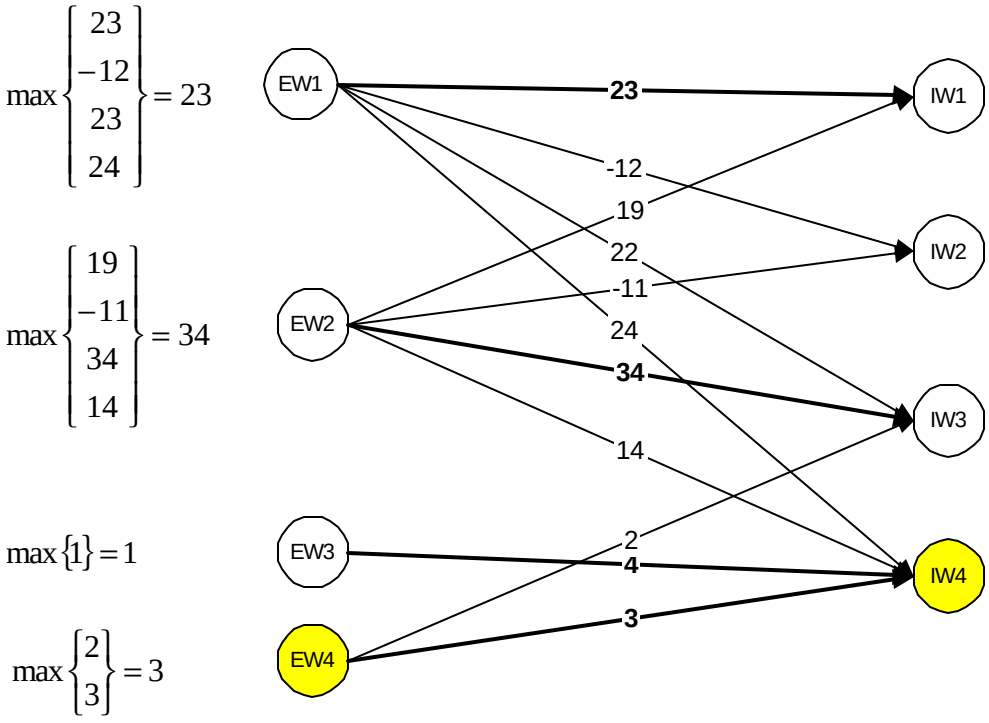


Figure 6.9. Third and second Stage evaluation

So we first consider the last stage. We want to maximize the combined score improvement by considering the arcs between each state in the third stage and the second stage. Next we look for the arc that maximizes the combined score improvement knowing the best options from the previous stage. The process would go on until all the assembly constructions have been chosen for all the assembly types. Figure 6.10. shows the best assembly constructions based on the set of specified importance weight for the criteria. If we change the relative weight assigned to the criteria a new set of assembly constructions might be chosen to be the best assembly construction in this case. The best assembly construction is dependent on the criteria selected, the relative

weight assigned to the selected criteria and the particular design concept being considered. So for a new design concept the solution will be different.

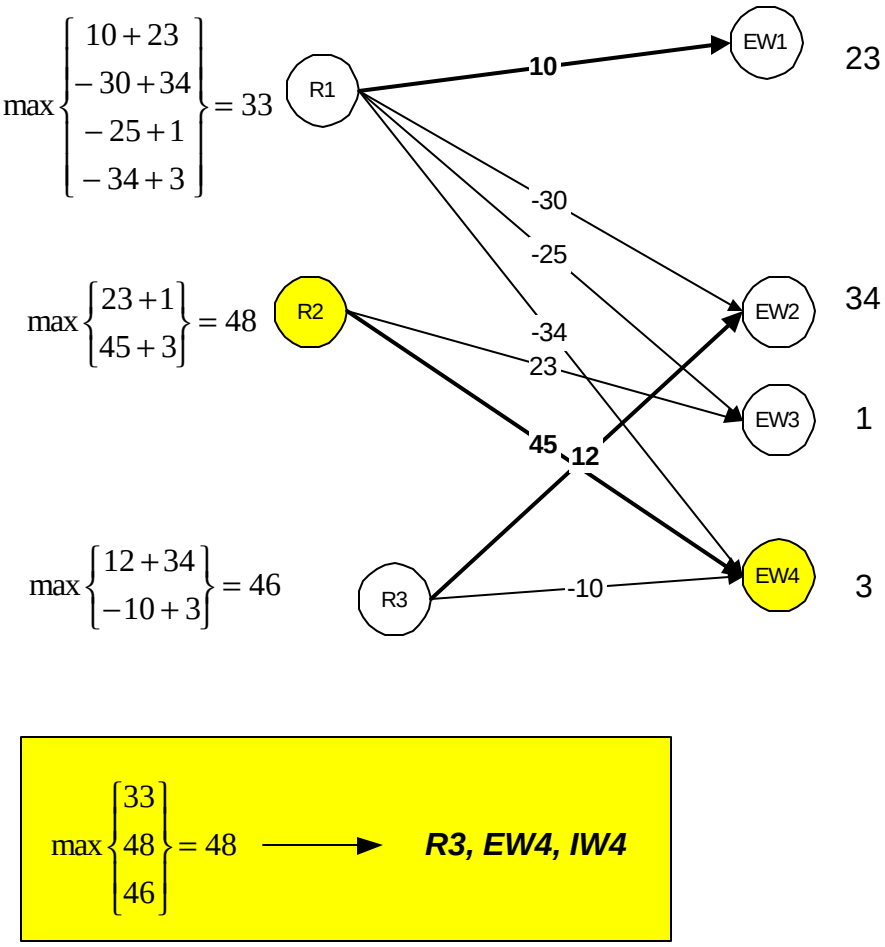


Figure 6.10. Second and first stage evaluations

It is important to note that some assembly constructions that are not applicable for the particular design case (e.g. a few assembly constructions were eliminated in the first step by determining the applicable assembly list) can be eliminated from the selection procedure. Eliminating these assembly constructions reduces the number of evaluations that have to be made.

To formulate the problem this way, two main conditions must be satisfied. Firstly the values chosen to be the basis of selection (i.e. the improvement to the base case in our case) must be additive (i.e. can be added from stage to stage). This is sometimes known as separability and is

satisfied here, since we are looking for the largest percentage improvement of the combined score. Secondly, the decisions on states in one stage can not invalidate earlier states. So in EASYBUILD's flat plate construction type, the size of the foundation assembly can not be determined until the size of the column assemblies have been determined which in turn needs the roof assembly's concrete slab size to be determined. Each construction type will have an assembly selection sequence.

Finally three issues must be noted. Firstly, the base case solution can have a significant meaning if it is interactively chosen by the designer. In this case the selection procedure will come up with a solution that produces the highest improvement in the combined score to an original design. The same solution should be reached independently of the base case solution. Secondly, if there exists an assembly in stage j that does not fit with an assembly in a stage other than $j\pm1$, a feed forward loop can be set up with an infinitely low weight so that this assembly construction combination does not get selected.

The third issue to note is that the relativity of the criteria score is more important than the actual scores themselves. For example, since we are seeking a comparison, relative costs are more important than actual cost; it is not important if our cost data becomes outdated as long as correct relativity is maintained.

We have discussed the assembly selection procedure used in EASYBUILD. A complete example is given in the next chapter. Next we will introduce the assembly generation procedure used in EASYBUILD.

6.2. ASSEMBLY GENERATION

6.2.1. Introduction

The result of the selection procedure is the best assembly constructions for the various assemblies in the building. The next step is to generate the graphical details for the selected assembly constructions. One method for generating the details is to store 2D details of the various assembly constructions in a detail library and retrieve the correct details for the selected assemblies (from

the previous step) for each new design concept. However in this case, the components of the selected assemblies can be different from the components of the retrieved library detail. The shapes, locations or number of components in the new selected detail can be different. Therefore the retrieved detail becomes invalid and must be changed. Furthermore 2D details have been criticized for being misleading sometimes [DeVries 1996], [Shah and Mantyla 1995]. Therefore several designers have turned to representing the details as 3 dimensional drawings. This makes reading and understanding the detail easier by the various construction professionals and unprofessional (like owners) alike. There is also another advantage of 3D details; a solid model (versus a 2D or a 3D surface or wire frame model), can be sectioned and various 2D details of the assemblies can be generated from a single 3D model. It has been argued that a 3D solid model of the building is the best representation of the any object [DeVries 1996]. Very little confusion can arise about the shape and the relations of the components of the model, because it is the closest graphical model of the object.

In order to generate the 3D details, we can also save them in a library and then retrieve the details for the selected assemblies. However, 3D details also suffer from the same problem as 2D. If the number, shapes, or locations of the assembly components change then we have to draw the 3D detail again. We would have to draw the detail again if any of the components of the new selected assemblies are different from the components of the stored 3D detail.

In this section we describe a procedure, that makes the generation of the 3D solid model (of the selected assembly constructions) easier. The procedure entails specifying parametric constraining operations to describe the locations of the components in the assembly. Then these operations are run for each new assembly detail to place the components in their correct positions. We will start by describing the types of constraints that can be used to constrain the components' locations and orientation in an assembly. In section we will describe how we these constraints can be specified. Then, in section , we will describe the actual constraining operations proposed.

6.2.2. Types of constraints

The research presented here proposes using constraint-based geometric modeling (usually used in mechanical design) to help the designer in generation of the building assembly details. Constraint-

based modeling is a type of geometric modeling, often used in the mechanical design domain, where the drawn objects' locations, dimensions or, orientations are specified in terms of parameters and geometric constraints on these parameters. By changing these parameters a new model can be derived automatically. For example, the depth of a beam can be constrained to equal a certain parameter. This parameter relates to the length and load on the beam and in turn is calculated by a structural sizing procedure. By altering the length and load of the beam the depth of the beam will be changed automatically in the geometric model. These kinds of constraints can sometimes be called continuos constraints [Nassar et al. 1999c].

Continuos constraints are hard to specify and solve and usually requires expert intervention. Another kind of constraints is declarative (sometimes called discrete) constraints. These kinds of constraints relate the location of two objects together. Declarative constraints can be used to restrict the locations or orientations of certain objects in the model. For example a *mate* constraint can be used to insure that the beam object is geometrically located flush with a column as seen in Figure 6.11. So even if the depth of the beam changes, the location of the beam is always on top of the column. Declarative constraints can be specified as a sequence of constraining operations that constrain the locations and orientations of certain objects in relation to others.

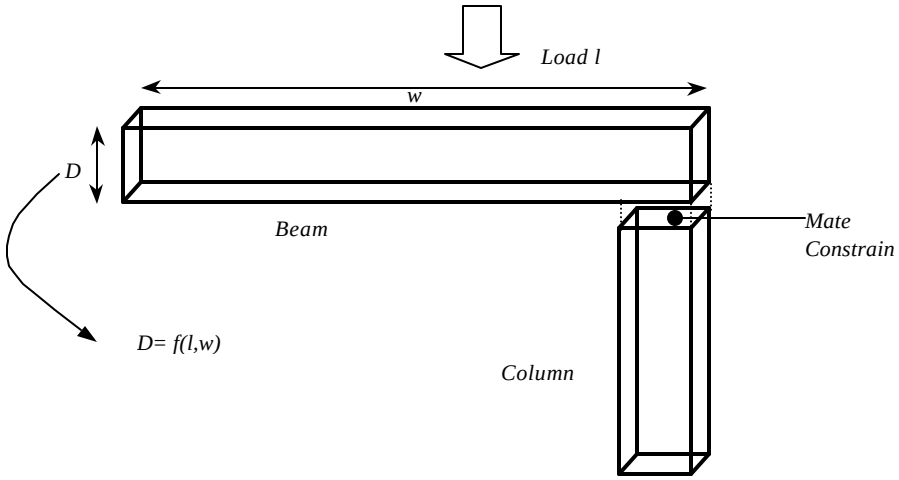


Figure 6.11. Examples of Constraints

The *mate* constraint above is only one type of constraint. Various generic constraint sets have been proposed in the literature. However there are, as yet, no standards for specifying or

representing constraints in the domain of mechanical design [Shah and Mantyla 1995]. Figure 6.12. shows an example of generic constraint classification [Shah and Mantyla 1995]. The constraints are divided into two main groups; orientation and position. The position constraints are used to define distances between two points as constraints. They specify the distance measure from a reference entity to a target entity. A special case of the position constraint which is used in this research is the coincident constraint, which constraints two points on the target and reference entities to coincide. The orientation constraints are divided into five types: parallel, perpendicular, angle, coplanar and coaxial. The co-axial constraints, which are used in this research, are applied between two lines. The z-axes of both the target and the reference entities are specified to be coaxial.

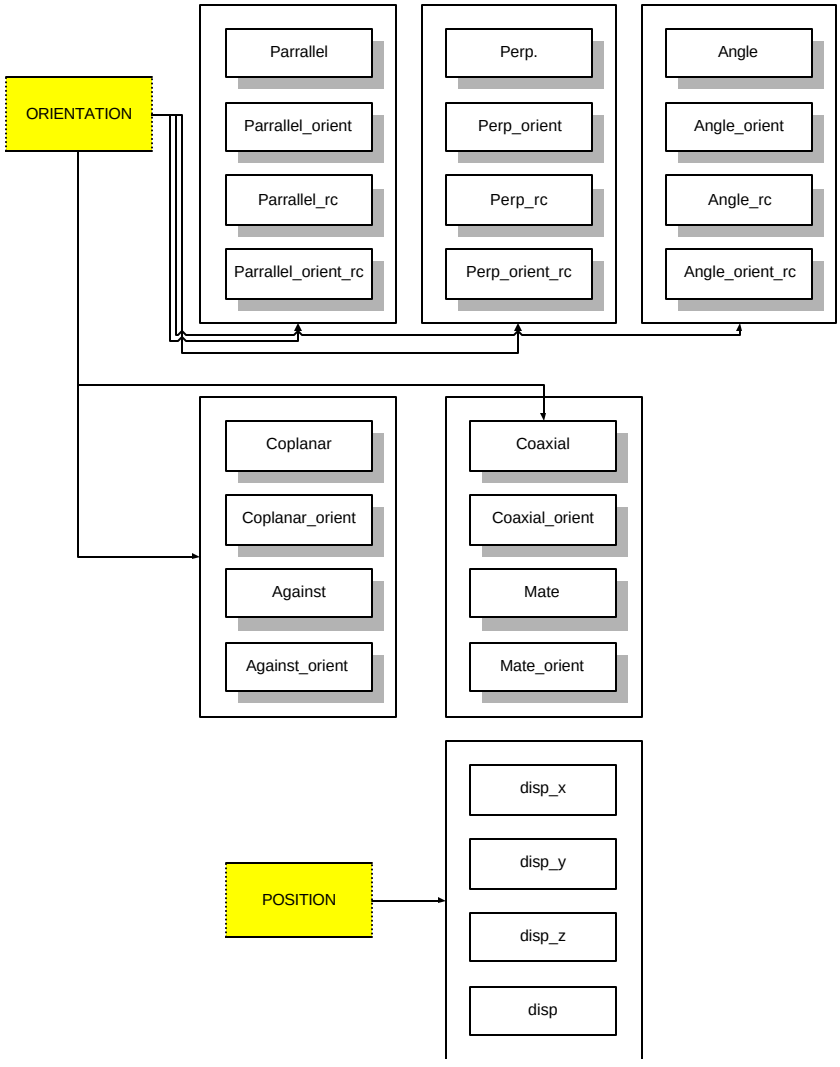


Figure 6.12. Different Kinds of Constraints, adapted from [Shah and Mantyla 1995]

6.2.3. Constraint Specification

The assembly constructions selected for the CADREPs are made up of different components. A solid model represents each of these components (component CADREP). Various solid models for components are shown in Figure 13. In order to generate the 3D solid model of the complete assembly detail, from the selected assemblies' components, we have to parametrically describe how the components' CADREPs fit together. This means that the various components' locations in the assembly have to be constrained in a way that if the shape, location, or number of the components change, then the detail would change automatically.

Using the declarative constraints described above we can constrain the locations and orientations of the components in an assembly. Some constraint-based modelers exist that allow the specification of some declarative constraints, e.g. AutoCad Designer, ProEngineer, etc… For example consider the detail in Figure 6.14. This detail was constrained specified in a commercial constraint-based modeler, AutoCad Designer. AutoCad Designer has four types of declarative constraints (of the types specified above). The various constrains are shown in Table 6.3.

Constraint	Description
Mate	To join points, axes, planes, or non-planar faces.
Insert	To align two circles, including their center axes and planes, use the Insert constraint.
Flush	To make two planes coplanar with their faces aligned in the same direction, use the Flush constraint.
Angle	To control an angle between two planes or two vectors, use the Angle constraint

Table 6.3. Constraints in Mechanical Desktop

Once the constraints have been specified as shown in Figure 6.14. we can change certain parameters of the components in the assembly and the assembly will be updated automatically to the reflect the parameter change while keeping the same constraint relations. For example, if we change the width or length of the column in the detail in Figure 6.14. the assembly will automatically reflect the change and a new detail with the new column dimension will be shown.

These constraint-based modelers are developed for mechanical applications and have not been used in architectural design. Although the constraints and parameters used in these systems can be used to constrain the components in an architectural detail, we found that the use of these constraints and parameters, as they are, in architectural detail development has a few drawbacks.

The kinds of parameters that can be changed in these modelers are discrete parameters (that are geared towards mechanical design), e.g. example the length or width of an element, a radius of cam, etc.... In order to change parameters like the shape of an element (e.g. new brick shape) or, number or location of the components (e.g. number of metal ties, or brick spacing) in this 3D model we have to manually do so. The Number of Components/Elements must be known beforehand and can not be dependent on the design case e.g. the number of ties in the example of Figure 6.4. can not be changed unless we manually do so. Also, because we are using the constraints to specify the relationships between the components, the detail must be fully constrained. Constraints are geometrical and do not bare any relation to the designer (i.e. as construction operations). The constraints are harder to specify so that the architectural detail is fully constrained.

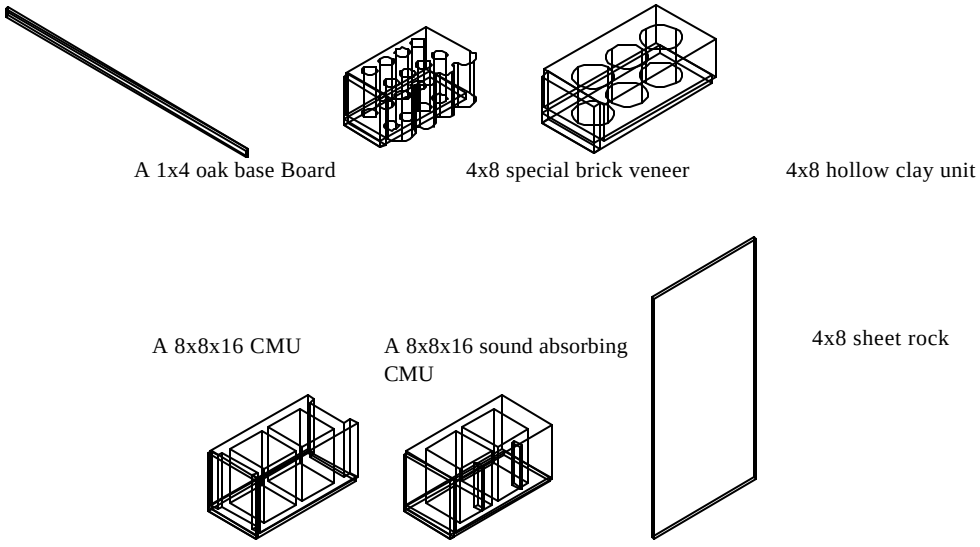


Figure 6.13. Examples of Component CADREPs

Therefore, for easier specification, the constraints in EASYBUILD are associated with a set of parametric operations. These operations describe the steps in positioning the components

CADREPs in the assemblies CADREPs, and also relate to construction operations. This approach is sometimes called constructive specification [Nassar and Beliveau 1999c].

The operations and the constraints associated with them are shown in Table 6.4. Each operation takes component CADREPs of a particular type. For example the "STACK" operation is used for masonry type CADREPs and can take a brick or a CMU CADREP. The "LAYOUT" operation on the other hand takes CADREPs that are to be placed at certain interval and can take for example the metal ties CADREPs. Each operation also has a set of parameters associated with it. The "LAYOUT" operation for example has the spacing parameter to determine the spacing between the elements.

The "ASSEMBLE" operation is the operation used to connect components together. "PLACE" is the operation used for placing a component or assembled components in relation to an assembly CADREP. "COVER" is the operation for overlaying material over a surface like tiles, ply roofing, etc… "CUT" operation is used to penetrate or trim components e.g. wood. Other operations that can also be used are DIG, AND SPRAY.

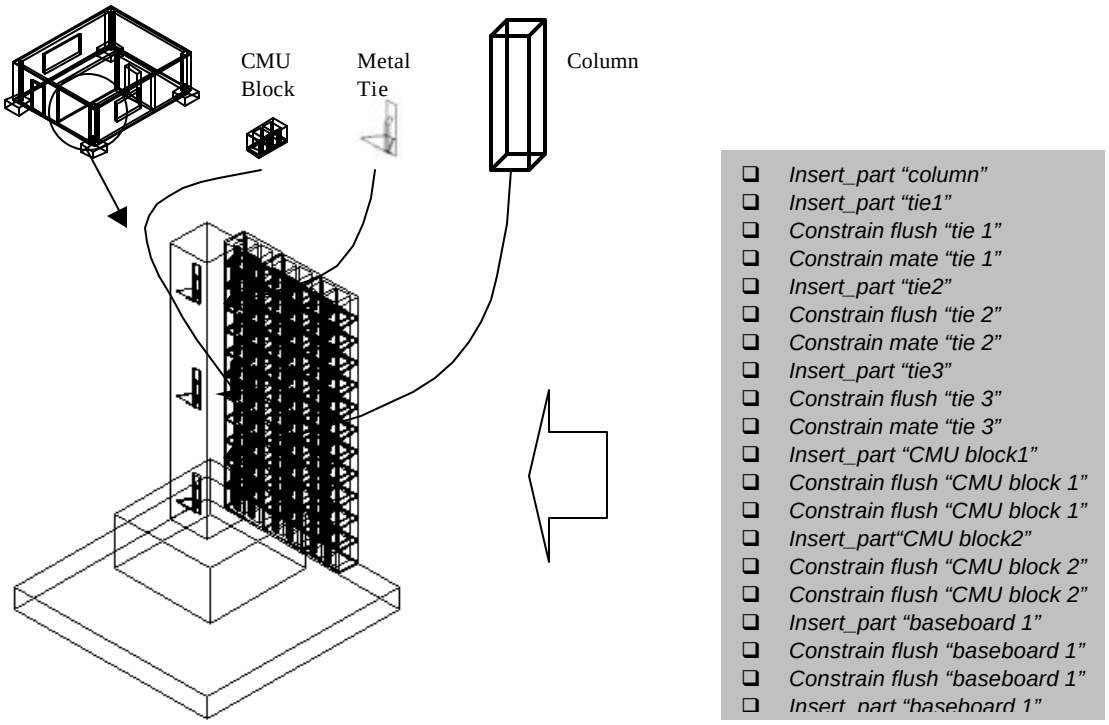


Figure 6.14. Developed Example

Each operation also has a set of geometric constraining parameters that define the geometric constraints. The geometric parameters define the constraints on the component CADERPs in the assembly. Constraints usually have a target and a reference entity. The target entity is the entity that is to be constrained. The reference entity is the entity the target is constrained to. In order to specify the constraints in each operation the target and reference entities are actually the work features (defined last chapter) on the assembly and component CADREPs. For example the "PLACE" operation will take two points (i.e. one on the component CADREP and the other on the assembly CADREP) the have satisfy a coincide constraint along with two directions (one on the component CADREP and the other on the assembly CADREP) that have to satisfy a coaxial constraint (Figure 6.15.).

Operation	Target Entity	Reference Entity	Geometric Parameters	Parameters	Example
LAYOUT	Component	Assembly	Two points, Two directions	Spacing	Metal ties, fixtures
ASSEMBLE	Component	Component	Two points, Two directions	-	Bolts, Screws
PLACE	Assembly	Component	Two points, Two directions	-	Studs, baseboards
POUR	Assembly	-		-	Concrete
COVER	Assembly	Component	Two points, Two directions, start point, end point	Angle, Spacing, Overlap	Tiles, Sheet-Rock
CUT	Component	-	One point, one, distance, one plane	Width, Length, etc…	Sawing wood
STACK	Assembly	Component	Two points, two distances, start point, end point	Vertical joint spacing , Horizontal joint spacing	Masonry

Table 6.4. The defined constraining operations

The constraining operations for the same detail shown in the example above (Figure 6.14.) are shown in Figure 6.16. Once these operations have been specified for this assembly detail, the same operations can be used to generate other details with different CMU shapes, spacing, or metal tie shapes or spacing.

Notice that the operations also relate to actual construction operations. This has the benefit of simplifying constraint definition, since a designer can relate more easily to these operations than abstract geometric operations and constraints. More importantly, since these operations relate to building components, each component will be associated with the same operation regardless of the assembly CADREP used. For example a "Vinyl Tile" component will always be associated with the "COVER" operation. After the assembly construction is selected, the components of the selected assembly construction are known and presented to the designer with the set of operations associated with them. It is up to the designer to use these operations and other to specify the complete set of constraint operations for the particular assembly CADREP.

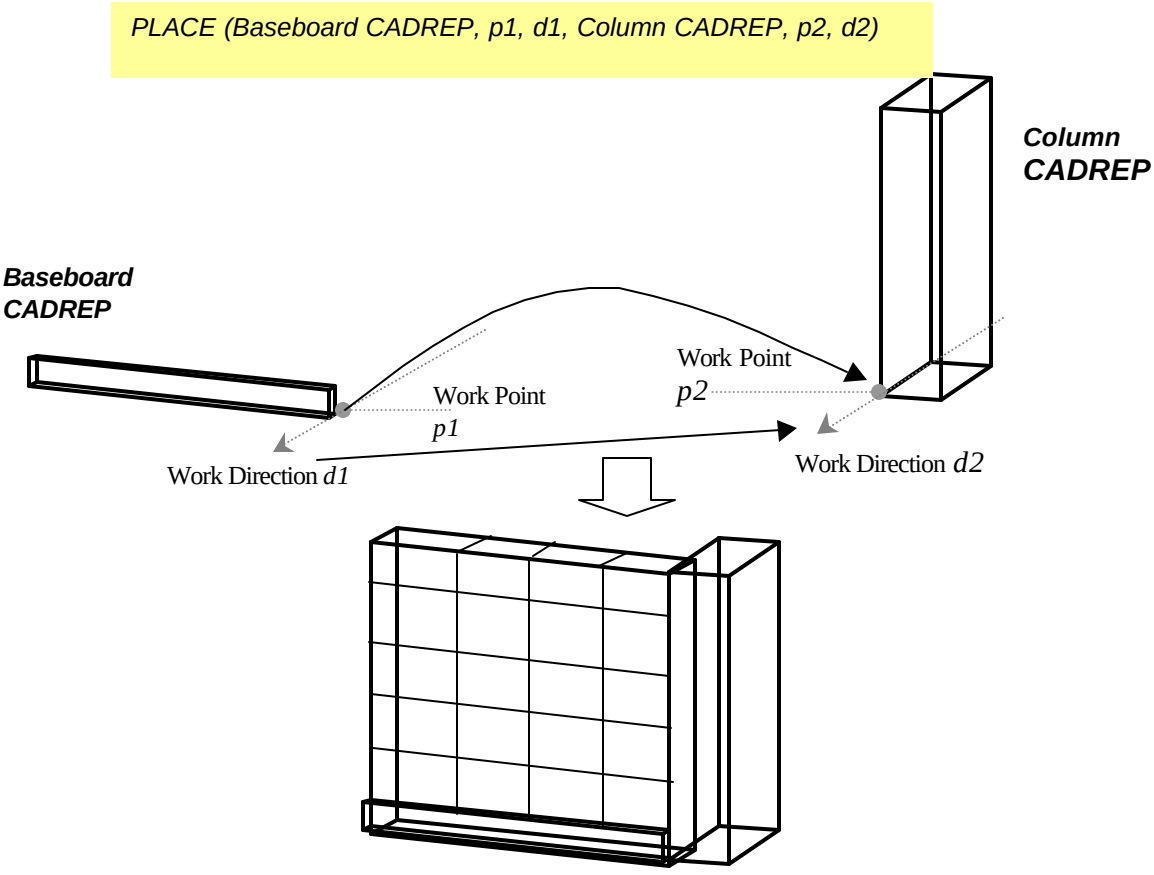


Figure 6.15. Example Place operation

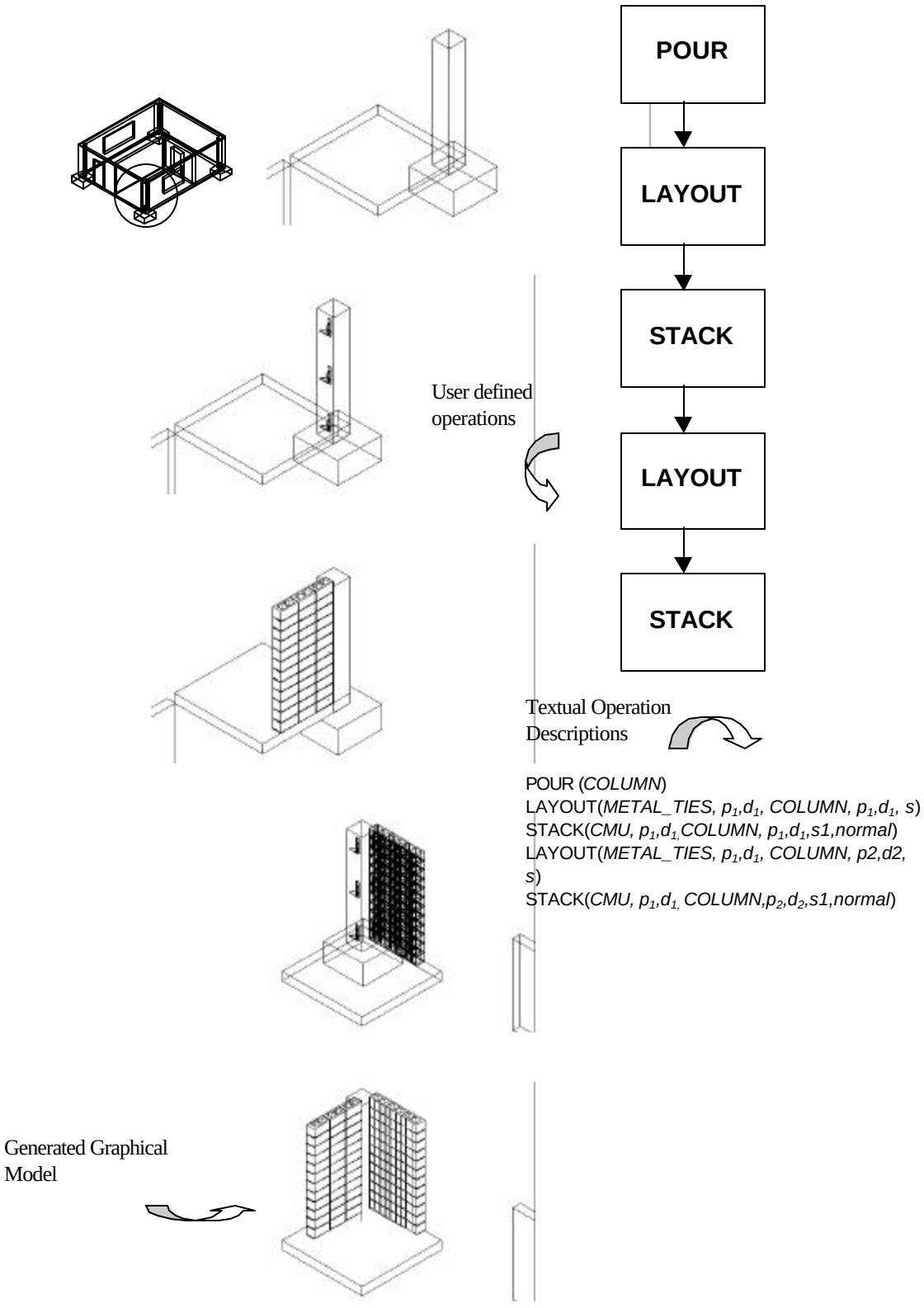


Figure 6.16. The Operations for the developed example

In EASYBUILD an interface is developed for specifying these operations, but currently only the place operation is functional. The positioning algorithm used in the place operation is shown in Figure 5.17. By using the algorithm in Figure 5.17. the target entity is placed by in the correct location in the reference entity in a single step. Notice that both the direction and the point are independent of the actual scale of the assembly unit CADREP, and therefore can be specified only once.

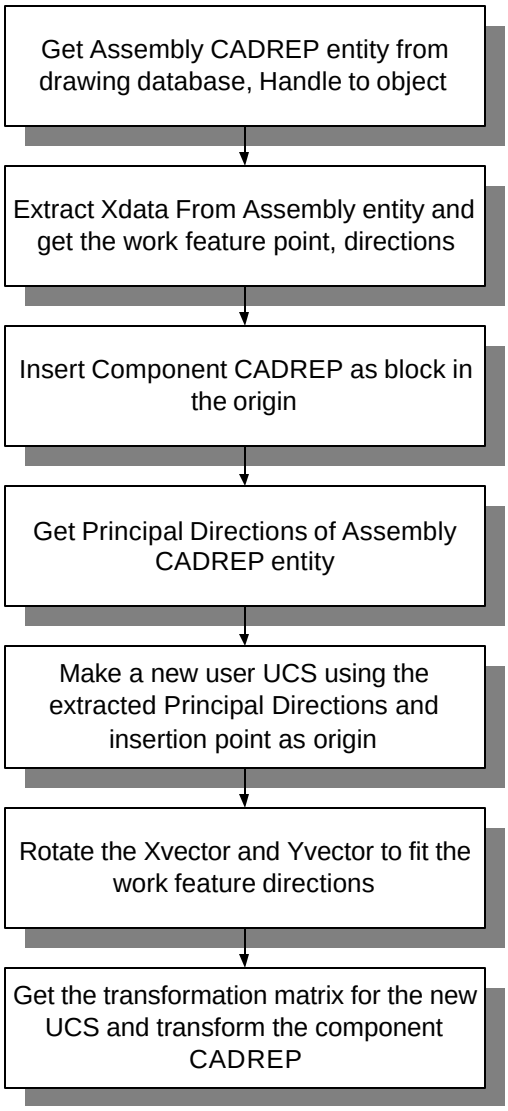


Figure 6.17. The Place operation algorithm

The generated assembly detail is a solid model. This means that the assembly can actually be sectioned in many ways to produced different 2D details if desired. Examples of 2D details generated from the solid model of the detail of the developed example above are shown in Figure 6.18.

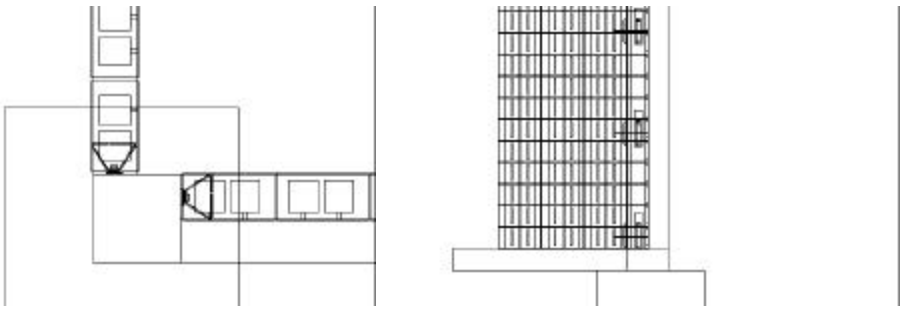


Figure 6.18. The sectioned 2D details

6.3. SYNOPSIS

In this chapter we presented the assembly selection and generation procedure used in EASYBUILD. First the assembly selection procedure was described. The assembly construction selection problem was defined and the definition of the best assembly was discussed. The definition of the best is dependent on the designer input. The various criteria, and their relative importance weights determine the best solution. The AHP for assigning the weights to criteria and the sub criteria was discussed. The weight assigning process used allows for a more accurate definition of criteria weights and can also handles a hierarchy of sub-criteria. Next, the formulation of the combined score was introduced. It was shown that there are three methods for formulating the combined score of the criteria for each assembly construction. The three methods and the special uses for them were discussed. In order to limit the number of assembly construction combinations and hence the criteria evaluations, the selection problem was divided into stages and states. A technique based on dynamic programming was used to search for the best assembly combination.

The database constraint-based selection approach described in this chapter is important because there is no analytical evaluation models for a lot of the assembly performance criteria. For

example there is no formula to tell us the STC rating of or the fire rating an assembly. These performance criteria have to be evaluated as a result of laboratory tests and stored in a database of assembly constructions.

Furthermore, some of the criteria are not associative and have to be evaluated for the building as a whole and not as individually. *The whole is different than the sum of the parts*. Even the easier-to-calculate performance criteria like cost are not associative criteria, and the cost of one assembly construction will depend on the existence of other assembly constructions. The absence of analytical methods to evaluate the performance criteria makes it difficult to use analytical generation methods of building assemblies.

The assembly generation procedure was also introduced. The assembly generation procedure uses constraint based modeling to help the designer in generating the 3D solid model of the building. Once a 3D solid model is generated, the model can be sectioned and various 2D details can also be generated. This section described the various operations that are coupled with the constraints to generate the assembly details.

The EASYBUILD system was developed to validate the procedures and concepts introduced in this research. In the next chapter we will describe the prototype system EASYBUILD. The design concept shown in Figure 6.2. will be used in detailed worked example.

CHAPTER 7. **THE EASYBUILD SYSTEM**

In this chapter we present the developed prototype system EASYBUILD. We will describe the functions and structure of the EASYBUILD system. First in section 7.1. we will give a description of the system functionality. The various modules that make up the prototype system will be presented: the Design concept definition module, the Database, and the Assembly Generation Definition module. For each of the modules the developed graphical user interface (GUI) will be described.

We will also give a detailed example of using the EASYBUILD system. The example design concept shown in Figure 6.2. will be used as an example building. We will use the developed prototype to select various assembly constructions based on different sets of criteria.

7.1. SYSTEM DESCRIPTION

The EASYBUILD prototype system was developed as a proof-of-concept to validate the ideas presented in this research. The EASYBUILD data structures are based on the developed product model, which was discussed in chapter five. The assembly selection and generation procedures discussed in the last chapter are also incorporated in the EASYBUILD system. The system allows a user to define a design in terms of graphical objects called CADREPs. As we described the CADREPs are representations of the different assembly types (external walls, roofs, etc.).

The user can attach different kinds of information to the graphic assembly objects (i.e. the CADREPS). For example, for the external wall assembly CADREPs the user can add constraint requirements on the finishing material of external wall assemblies. The system also has a database

of different assembly constructions with various criteria and constraints to govern how the assembly constructions are used.

The assembly selection procedure helps the user select the best building assembly constructions for the various assembly types (i.e. external walls, roofs, etc..). The selection is based on a set of criteria that can be selected by the user or generated automatically from previous cases. For example, given a set of performance criteria (e.g. annual cooling load, cost, wall permeability etc…) the system will select the best assembly construction for the external walls (e.g. solid CMU masonry wall or pre-cast concrete panel wall etc…) as well as for the other assembly types. The selected assembly constructions are the ones that best fit the criteria and do not violate any of the defined constraints. Currently, only examples of the various constraint and criteria types are defined in EASYBUILD (described in chapter six). An expert user can add new constraints and criteria. The assembly generation procedure defined in EASYBUILD helps the designer to automate the graphical generation of the assembly 3D solid model detail.

As can be shown in Figure 7.1. the system consists of three main modules;

- The DESIGN_CONCEPT Definition Module (DCDM)
- The Database
- The Assembly Generation Definition Module (AGDM)

Next we will look at each module in detail.

7.1.1. The DESIGN_CONCEPT definition module (DCDM)

7.1.1.1. Development Of The DCDM

The DCDM is the module that allows a user to model the building as building assemblies, as opposed to lines and circles that represent these assemblies. The user models the design automatically in three dimensions. EASYBUILD allows the user to specify different information about the building (e.g. location, building type, different requirements) and its assemblies (e.g. finish material requirements). The EASYBUILD system also allows the user to specify certain requirements about the assembly construction to be selected, like finishing material or assembly types etc…

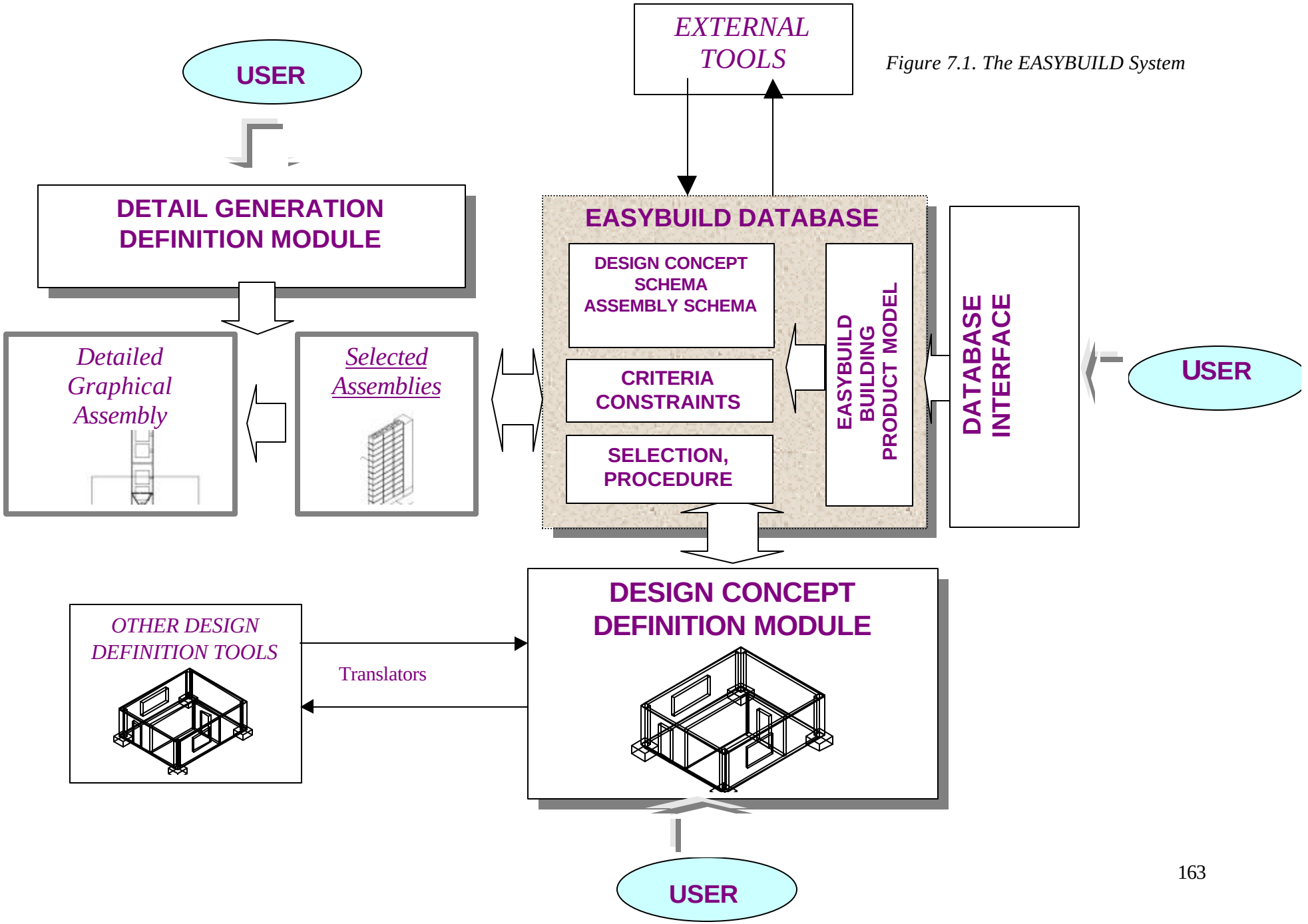


Figure 7.1. The EASYBUILD System

The designer can select the assembly construction types for the different assemblies from the EASYBUILD database e.g. "*stud-backed brick veneer*" for wall assemblies or, "*single ply membrane roof on 4" concrete*" for roof assemblies. On the other hand these assembly constructions can be selected automatically by the selection procedure, based on a set of *criteria* and *constraints* defined in EASYBUILD. The DCDM module also is used as the graphical engine for generating the assembly detail.

In developing the DCDM we had to review the available state of the art technologies in CAD tools that allow for customization (i.e. programmability). We needed to consider CAD tools that support 3D. It has been postulated that the reason architects do not directly design in 3D, is the lack of adequately flexible representations for 3-D design. Clay, chipboard and other materials are too rigid and incorporate structural constraints that do not allow for the flexibility needed (however few architects like Antonio Gaudi relied primarily on 3D models). The introduction of 3D computer modeling and representation schemes promised to provide an alternative flexible media. A comprehensive history and summary of various 3D computer representation schemes can be found in [1], i.e. wire-frames, polygons, sculptured surfaces and solid models. Solid modeling has been declared to be the scheme that provides the most complete description of a design object.

Solid modeling is a technique used to represent solid objects in computers. Solid modeling of buildings promises to provide the capability to directly view in perspective the results of candidate design operations, which makes it more appropriate than 2D designs in paper and pencil. Also solid modeling provides a geometrically consistent representation of the design. All the drawings would be based on one 3D model of the building and all projects would be derived from the 3D model. Finally solid modeling allows integration of lighting, energy and other analyses and decision-making tools.

When considering the development of our system, two primary approaches are possible:

1. One can use a programming language with a low-level graphics library or an APPLICATION PROGRAMMERS INTERFACE (API)
2. An alternative approach is to use an existing CAD package and add the required functionality to it using one of its development tools.

First Approach: Many software development graphics libraries and application programmers interface are available commercially, for low level development. These development tools can be divided to vertex based and primitive based systems.

Vertex based systems are based on defining the various drawing objects in terms of their vertices, while primitive based systems have predefined geometric primitives (i.e. boxes, spheres, cylinders etc…) that provide a higher level development. One of the currently most popular vertex based systems is OpenGL [2]. OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that specify objects and operations needed to produce interactive three-dimensional applications. It is designed as a streamlined, hardware independent interface to be implemented on many different hardware platforms. Using OpenGL the user must build his objects from a small set of geometric entities- points, lines, and polygons. Many extensions are available to extend its functionality to solid models, like the GLUT library (GL Utility library). Other popular examples of vertex systems are Phigs, Direct 3D and MESA, Virtual Reality CAVE library.

Also many primitive based toolkits and APIs are available. One of the most widely used is ACIS. These types of systems allow the modeling of objects directly as solid models and therefore relieve the user from dealing with solid representations. The user must build his objects from a set of geometric primitives- cubes, spheres, cones etc. Other examples of such systems are Heidi, Optimizer and Open Inventor and ACIS.

Second Approach: On the other hand, many existing CAD package allow for development and adding more functionality, based on specific tools. An advantage of such systems is that many also allow for an integrated development environment (IDE) within their regular graphics and modeling tools.

Based on the review of such packages, the Design concept Definition Module was developed as an extension to a popular commercial CAD package: AutoCaD. One of the reasons AutoCad is a popular package is its ease of customization. AutoCad offers a multitude of ways to develop customized applications on top of the regular drafting interface. Among the ways of developing extensions is AutoLisp, Visual Lisp, C++ and ADS and Visual Basic For Applications (VBA). In this research the development tool was Visual Basic for Applications. The reason for choosing this development tool is its ease of extension and compatibility with several other software packages like databases and spreadsheets. Visual Basic was first introduced in AutoCad Version 14 (which was used to develop EASYBUILD).

7.1.1.2. The user Interface

Figure 7.2. shows the interface developed for defining the design concept. The system currently allows the definition of the various assembly types for the flat_plate_construction type, i.e. external walls, internal walls, floors, roofs etc. By selecting the icon for the appropriate assembly type the user can insert the CADREP for that assembly. If the assembly has different CADREP types, the user can also select the desired type.

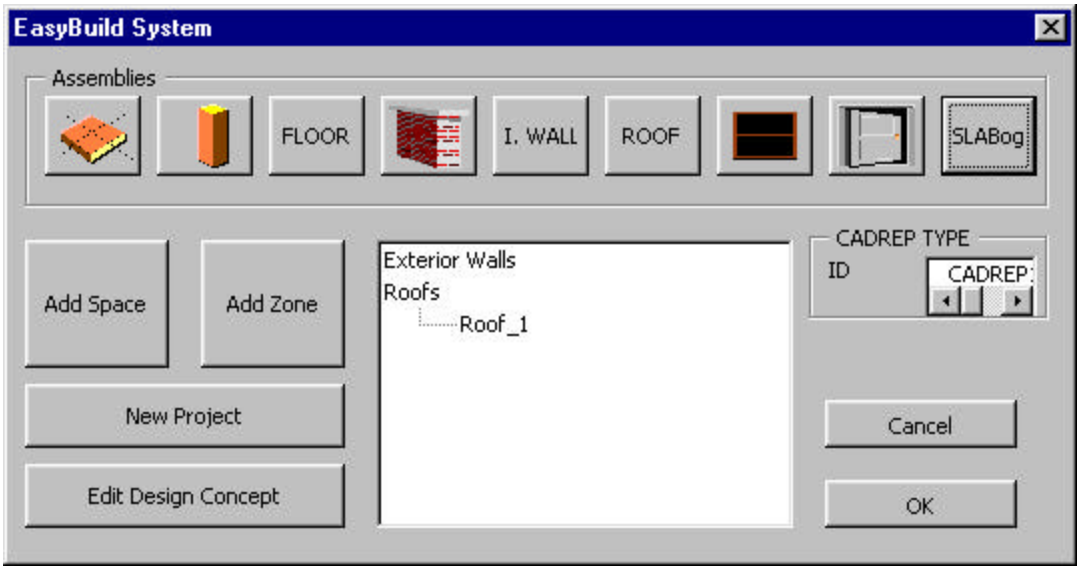


Figure 7.2. The main DCDM Interface

The EASYBUILD system automatically creates a record for the defined CADREP in the appropriate database table and keeps track of the various CADREPs in defined in the drawing. The various CADREPs are shown in a tree view as shown in Figure 7.2. The user can select the appropriate assembly and view the data in a template to edit it. For example in Figure 7.3. the template for an external wall CADREP is shown. The user can specify certain requirements on the finishing material or on the assembly selection used. Furthermore, if the user does not want the selection procedure to automatically select the assembly construction, the designer can manually select the assembly construction from the applicable assembly construction list (this is the list of assembly constructions that do not violate any of the constraints that limit the use of the assembly construction). Alternatively, if the assembly construction is not specified, the assembly selection procedure will determine the best assembly construction automatically.

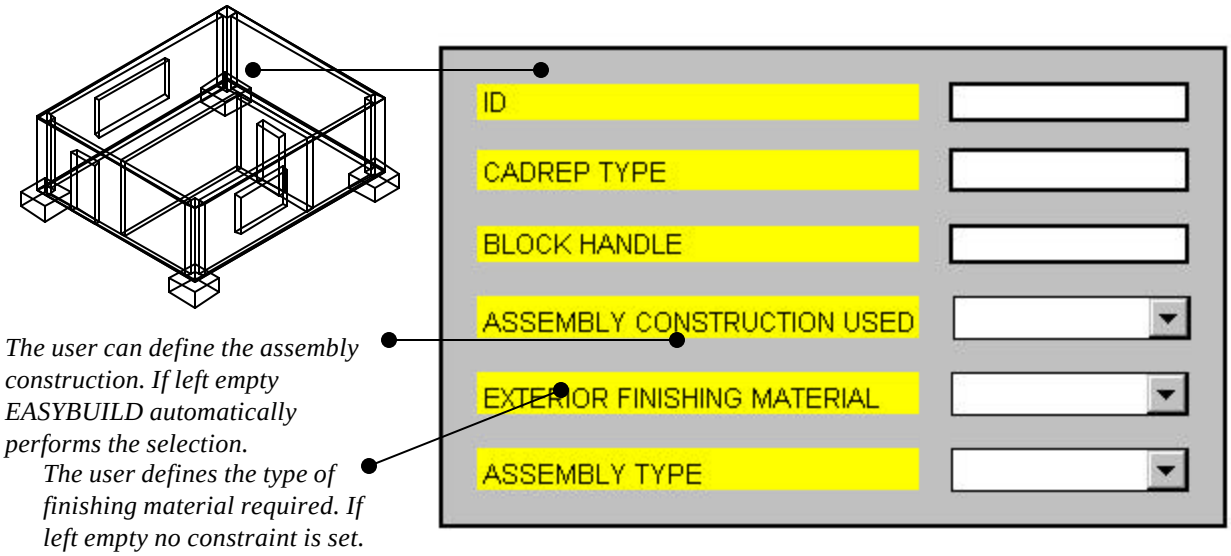


Figure 7.3. Defining Assembly Data

7.1.2. The Database and its Interface

7.1.2.1. Description

The EASYBUILD database (which was described in chapter 5) is an implementation of the formal building product model. The database is used to store both the design concept description entered by the designer using the DCDM and the assembly construction options. As we mentioned before, the design concept description includes all the information of the building before the assembly constructions are selected for the different assembly types. The Assembly constructions are the different possible constructions for the various assembly types. The goal is to select from the assembly constructions the assembly that best satisfies the designer's criteria and constraints. Therefore an interface is required that will capture the designer's input and then perform the selection based on the procedure defined in the previous chapter.

7.1.2.2. The interface

The database is also used as the interface for defining the various performance criteria to be considered when performing the selection as well as the various other options of the selection procedure (like the combined score method, weights etc…) as described in the last chapter.

Figure 7.4. shows the criteria selection menu. The designer selects the criteria which are to be used considered for the particular design case. By selecting the various criteria an applicable criteria list is created. The user also can specify the relative weights of each criterion. This is done using the AHP described in the last chapter.

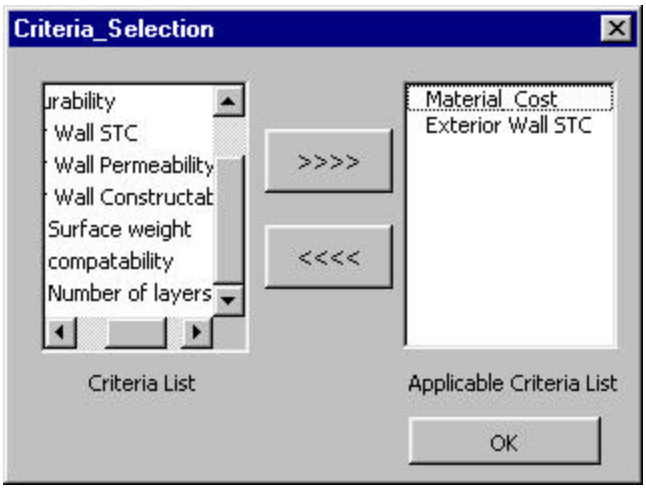


Figure 7.4. Selecting the criteria

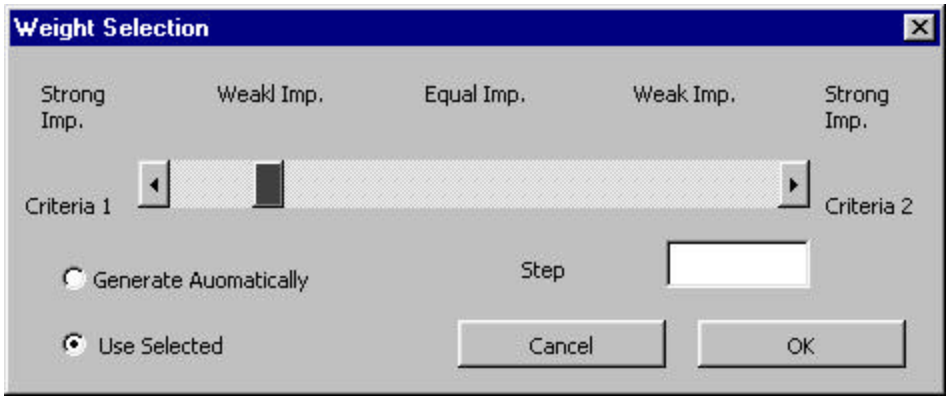


Figure 7.5. Defining Criteria weight

The AHP requires pair-wise comparison of criteria. Therefore an interface for defining criteria relative weights is developed as can be seen in Figure 7.5. Figure 7.6. shows the calculation of the relative weight vector. By clicking on each cell in matrix of Figure 7.6. the criteria weight definition window of Figure 7.5.

appears and allows for weight definition. Then the resultant weight vector is calculated and the consistency ratio is checked.

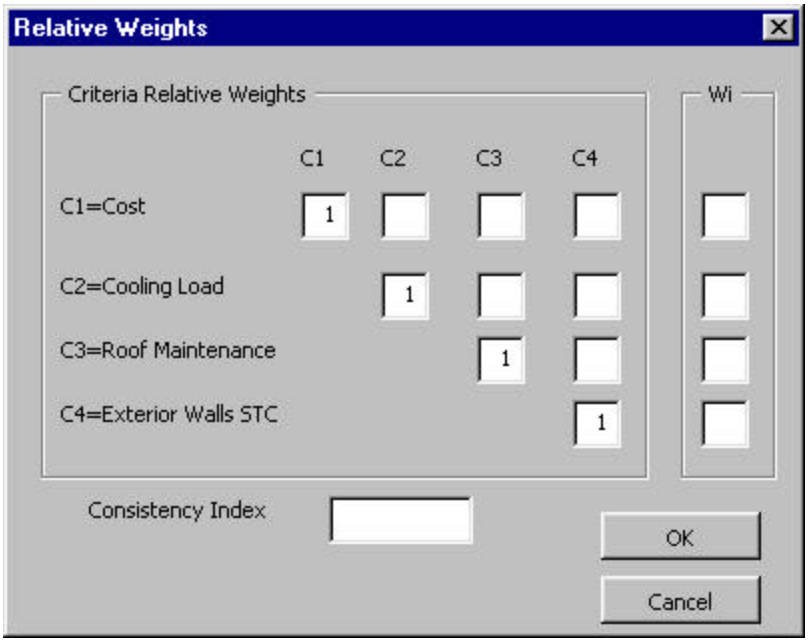


Figure 7.6. Assigning the weights and computing the resulting weight vector

The designer can save the set of applicable criteria list and their relative weights in order to use the same criteria set for other design concepts. The designer can also select the automatic weight assignment option. Currently the automatic weight assignment option in EASYBUILD, will result in the generation of a set of step weights assigned to the applicable criteria, in order to determine the best compromise solutions. For, future extension another option could be developed for the automatic criteria and weight assignment. In this option the applicable criteria and their weights are generated from previous cases as described in chapter 6. There are three methods of combining the different criteria scores as described also in chapter 6. An interface is developed for selecting the appropriate method for combining the criteria score depending on whether the designer wants a simple combined solution, the best well-round solution or a performance-offset solution. The default is the simple combined solution. The selected assemblies are then output as shown in Figure 7.8.

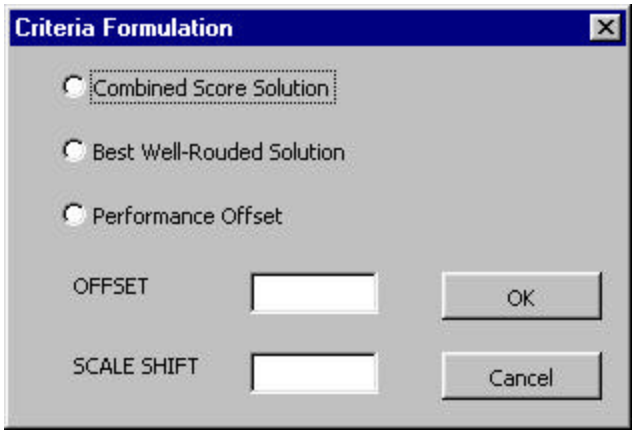


Figure 7.7. Defining the combined score formulation method

7.1.3. The Assembly Generation Definition Module (AGDM)

As we mentioned last chapter, the next step after the selection of the assemblies is to generate the details. The research presented here proposes using constraint based modeling to define the 3D solid model detail of the selected assembly once the assembly construction are selected. The constrained solid model of the assembly can be used in the future to generate similar details with different component shapes, sizes or number.

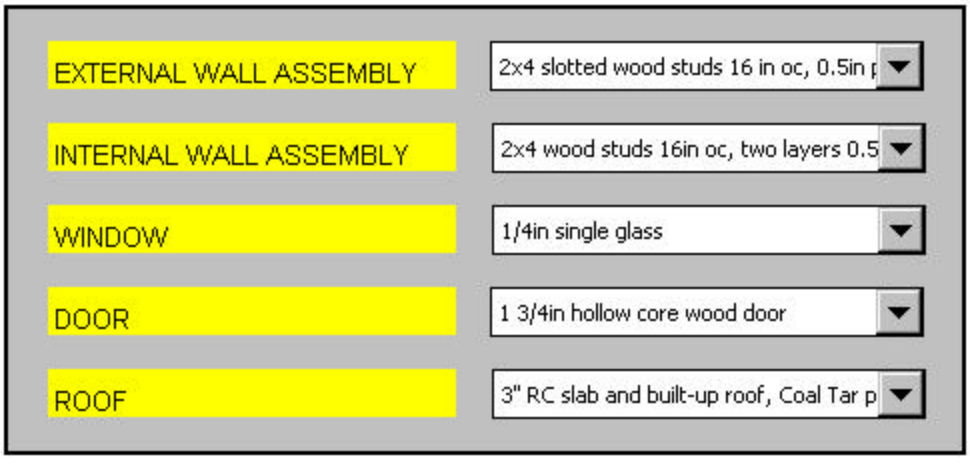


Figure 7.8. The Selected Assembly Constructions

The definition of the constrained 3D model is carried out using parametric geometric operations. An interface is developed for defining these operations. This interface is the assembly generation module. Figure 7.9. shows the AGDM. Currently only the PLACE operation is functional. By dragging and dropping the operations shown in Figure 7.9. the designer can build the sequence of operations required to generate the detail. For example, an interface is developed for the PLACE operation to define the PLACE operation parameter.

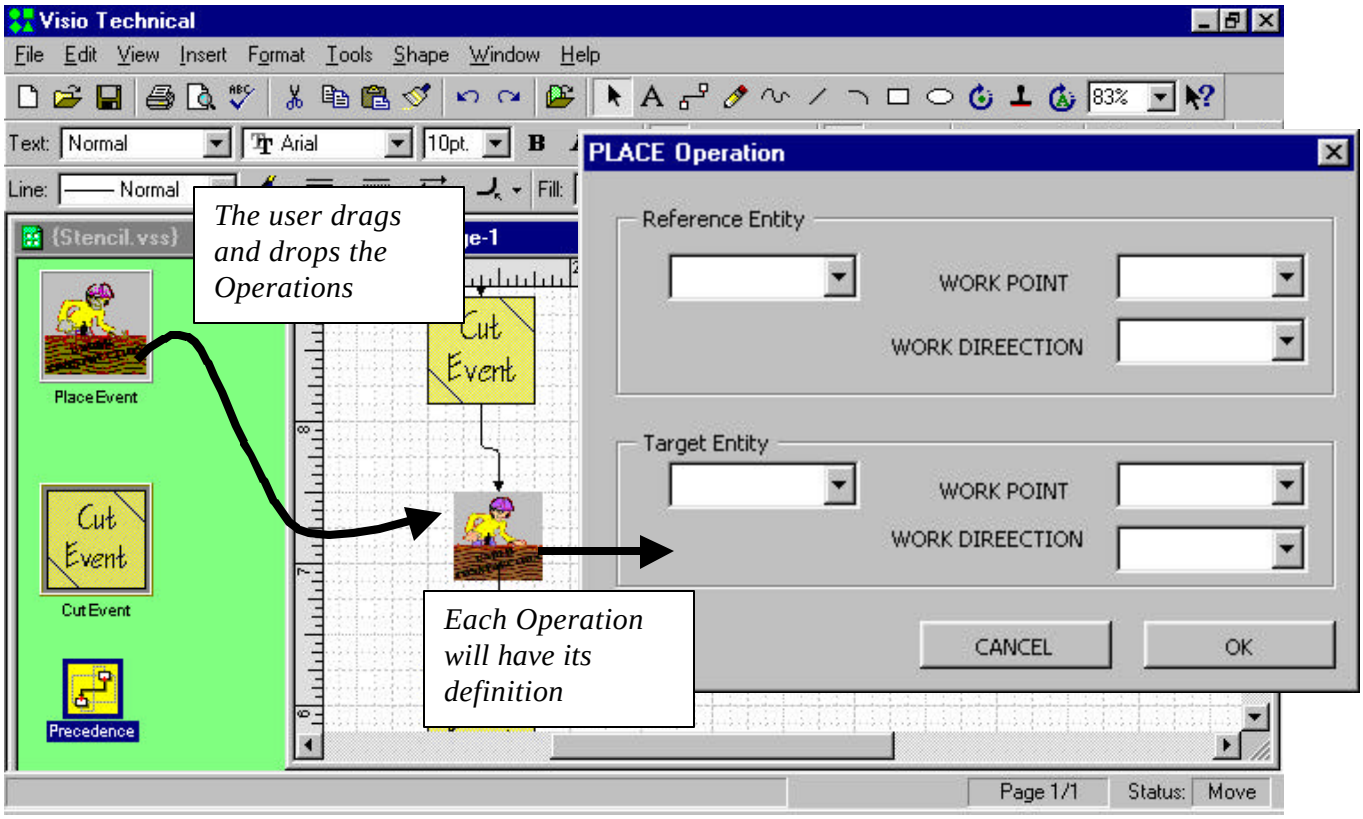


Figure 7.9. The DCDM

7.2. AN EXAMPLE

In this section we will go through an example of automated assembly selection based on the selection procedure described in the last chapter. The example will also help to demonstrate how EASYBUILD works. The design concept shown in Figure 6.2. was modeled using the EASYBUILD DCDM. A new design concept was created. The different information pertaining to the building like the location, use group, occupancy and other attributes of the design concept were defined by the designer.

Next the assemblies were modeled using the different CADREPs which represent the different assembly types. EASYBUILD automatically instances the excel templates to create a new database for the particular design concept instance. The created database contains the various tables with the fields and relations defined in the original template (which was based on the building product model defined in chapter 4). For example, EASYBUILD creates new tables for the new assembly CADREP types (i.e. external walls, floors, roofs, etc…) and links the new CADRES in the design concept to the records in the CADEEP tables.

The designer can also select any particular assembly and view its information. Also some user requirements can be added. For example a constraint was added to limit the finishing material on the external walls to be a painted surface. Next the generic and assembly-specific constraints are evaluated automatically.

The building shown in Figure 6.2. is a small office and a store in a physical plant located. The building will house a loud piece of equipment so sound performance and floor resistance are important factors. Also the building is located in a hot climate and does not have any HVAC systems, so the cooling load is important. Other criteria to be considered are cost and the roof durability. Figure 7.10. shows "spider diagrams" that plot the different normalized criteria performance. Spider diagrams are diagrams with multiple axes, each showing a specific performance. The performances of a specific solution are shown on the various axes as can be seen in Figure 7.10. The spider diagrams help the designer visualize the combined performances of the selected solutions. Multiple solutions can be plotted to show the performances of various solutions and perform a visual comparison.

Firstly, in order to test the output of the prototype system, only one criterion was selected at a time. In Figure 7.10.(a) the cost was set as the criteria with the highest weight. The selected assembly constructions were: external wall: *"4.5in brick, 0.5 plaster on each side"*, roof: *"3" RC slab single ply, adhered"*, window: *"3/32in single glass with wood frame"*, internal wall: *"2x4 wood stud 16in o.c. 0.5in gypsum board both sides"* and, door: *"1 3/4in hollow core wood door"*. The selected assembly constructions had the lowest cost and did not violate any of the constraints, e.g. user requirement of painted exterior walls. In Figure 7.10.(b) the cooling load was chosen to be the most important criteria. The assemblies that were selected were the ones with the highest U-Value and increased the thermal mass of the building. However, notice that the cost performance for the second solution went down. In Figure 7.10.(c) the set of best compromise solutions were selected. These are the solutions which are generated by trying to find the best solution for each criterion. Notice that the two previous solutions are a subset of this set of best compromise solution. It is up to the designer or decision-maker in this case to make the selection among the various

solutions. Finally, a fixed set of criteria weights was set and the single best compromise solution was selected (this is also a subset of the previous set).

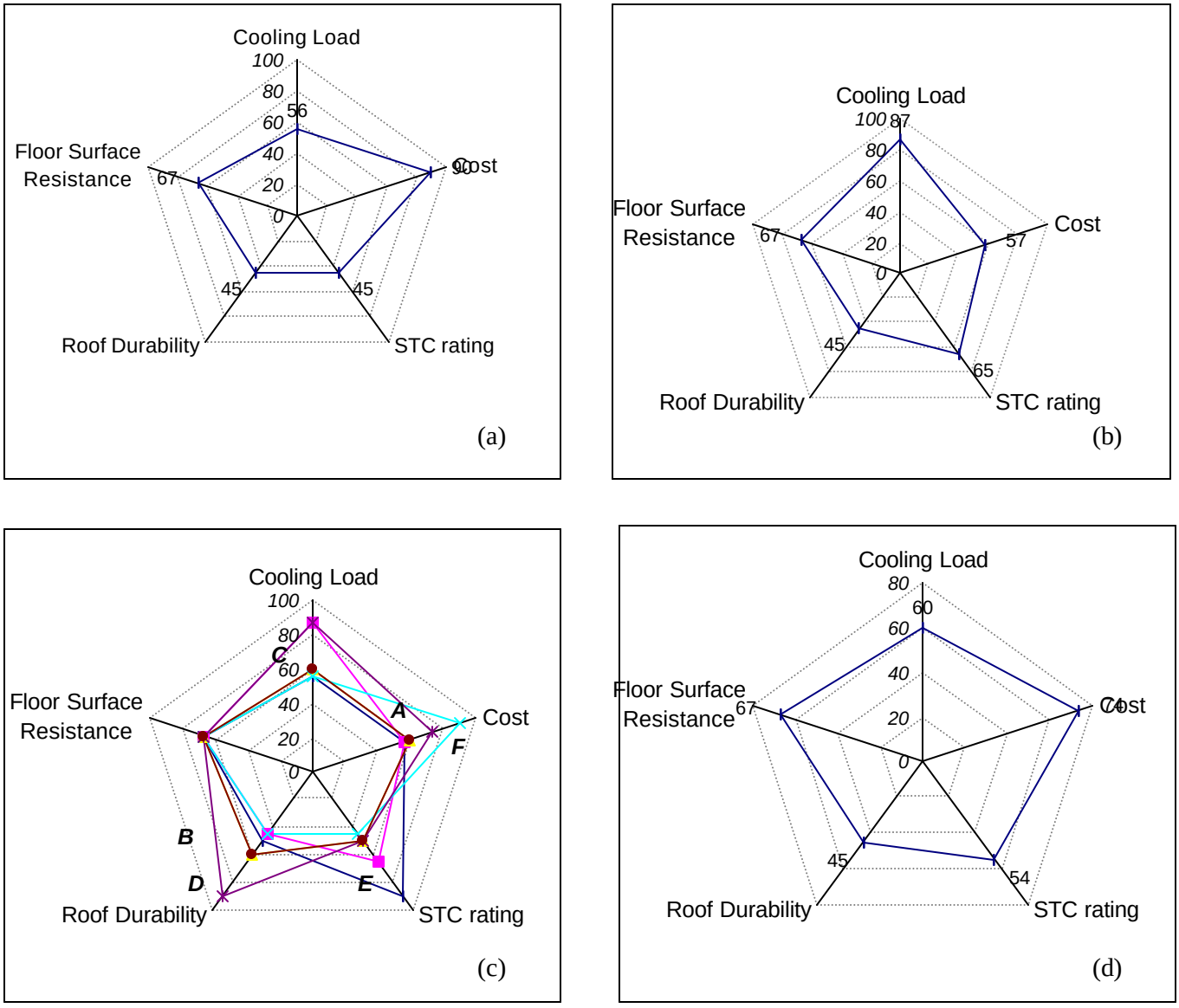


Figure 7.10. Spider Diagrams for the generated solutions

7.3. SYNOPSIS

In this chapter we presented the developed EASYBUILD prototype system. The three main modules of the system were described: the design concept definition module, the database and, the assembly generation definition module. For each module we gave a brief description of its functionality and its structure. We also

presented the developed graphical user interfaces for the various modules. The various development options for the DCDM were also discussed. Finally an example was given and the output was presented. In the next section we will present the contributions and suggestions for future research and development.

CHAPTER 8. SUMMARY, CONTRIBUTION, CONCLUSION, AND RECOMMENDATIONS

In the last chapter we presented the developed prototype system and example of its use. In this chapter, we will summarize the research presented in this dissertation and present the contributions and recommendations for future research. We start this section by summarizing the research presented in this dissertation. The contributions of the research are presented in the following section. Then in section 8.3. we present suggestions for future development and research and finally we conclude with a vision for the future.

8.1. RESEARCH SUMMARY

In order to develop an automated way for selecting and generating building assemblies, several tasks were carried out. In chapter two, we studied how this automation can be integrated with the design process. The different stages of design were described. Also, the different types of design were identified: creative, innovative, and parametric design. Parametric design or is usually characterized by changing the parameters of a design prototype without fundamentally altering the prototype. The second type, innovative design, is usually concerned with the adaptation and combination of two or more design prototypes, while creative design deals with creating a new design prototype.

Chapter three presented a review of the research carried out in the field of automated assembly selection and generation. We found that three research directions can be identified for automatic assembly selection: neural networks, case-base reasoning and, expert systems.

Neural networks were found to be useful for design emergence and in the generation of new assembly types, which have never been used before. However, using neural networks, it is difficult to add exact knowledge like fixed. Case based reasoning on the hand was found to be useful in finding cases from a case history that best correspond to the current design situation. Similar to neural networks, case based reasoning can not be used to represent exact knowledge like fixed rules. Expert systems were then reviewed. It was found that expert systems allow the representation of declarative knowledge only and generally can not handle procedural or algorithmic knowledge.

Also in chapter three, constraint-based editors were reviewed for assembly generation. Specifically the Construction Kit Builder system (CKB) and the SEED system were reviewed. It was found that the CKB system allows capturing geometric rules about the assembly layout in grids, while the SEED system has hard coded constraints for specifying constraints in enclosures.

In chapter four, the various product modeling techniques and examples of developed product models were reviewed. The two goals of product modeling were reviewed: developing various information bases like classes, databases etc… and to help in information exchange. Various modeling techniques used to develop product models were reviewed. Specifically, NIAM, EXPRESS, IDEFx1, Feature-Based Modeling, EDM and Object Oriented Analysis were reviewed and the findings were presented. It was found for example, that if the target system is not a relational system, for example, an object-oriented system, IDEF1X is not the best method. Also, since one of the uses of the product model developed in this dissertation is data exchange, EXPRESS was used.

Different building product models developed using these techniques were also reviewed and the findings were presented. It was found that, the building systems model, GRAM, RATAS, SIGMA/MOCIB and, the Building Core Model were developed as general reference models that follow the traditional approach for translation. It was important to study the objects, relationships, building breakdown structures and, the modeling techniques used in these models.

In chapter five the database and the product model behind it were presented. Firstly the conceptual model of the building was presented. The different entities in the model were presented and we discussed the steps of developing the database and the model. Secondly, a brief description of the developed proof-of-concept prototype EASYBUILD was discussed. The various modules that make up the system were briefly introduced: the design concept definition module, the assembly generation definition module and the supporting database.

Also in chapter five the database and its criteria and constraints were presented. The criteria were classified generically as two kinds, design concept criteria and assembly criteria. Design concept criteria are criteria that affect the building as a whole and take into consideration the interaction of the various assembly constructions e.g. the total material cost or the annual cooling load. On the other hand assembly criteria are those local to each assembly type like the external walls or roofs. Examples of both types of criteria that are implemented in the EASYBUILD system were identified.

Chapter six presented the assembly selection and generation procedures. The assembly selection problem was described. This was followed by the description of the proposed selection procedure. The selection procedure was broken down into five steps: design concept definition, criteria selection and assigning importance weights, determining criteria scores, formulating the 'best' solution and, performing the selection. Each step was described in detail. Also in chapter six the assembly generation procedure was described. Constraint based modeling was introduced as an approach to generically define the assembly construction solid model details in the computer, and automatically change components' sizes, shapes or numbers to generate a new assembly detail. A set of operations was described that facilitate that procedure.

In chapter seven the developed prototype system, EASYBUILD was described. The different modules of the system were described in more detail. The two options that could be used in the development of the

design concept definition module were described: using a low-level graphical library versus using an existing CAD package. An example was also given.

8.2. CONTRIBUTION

Most of the design support automation tools have been directed either partially or completely at the schematic design stage. This is because the highest impact on the building performance can be achieved at this stage. However, the schematic design stage requires a considerable amount of design synthesis tasks. Design synthesis tasks have been described as being highly subjective and creative. Therefore the use of design support tools in practice have been limited.

In this dissertation a new method for automating assembly selection and generation is provided. The design concept description and user requirements from the schematic design stage are given as user-input. The automation is geared at selecting and generating the correct or best compromise assembly constructions for the given design concept. This provides a more practical design tool that allows the designer to make an informed decision, given the designer's concept of the building design. Better decisions by the designer about the assembly constructions will lead to better building performance. Structuring the knowledge about the assembly constructions' selection criteria and constraints provides for fewer errors and omissions in the final design. Furthermore, by automating the assembly selection and generation procedures, significant time savings in the design process are expected.

So far the selection and generation of building assembly constructions has been carried out manually, given the designers experience. The developed method for selecting and generating building assemblies, complements the current manual design practices by structuring the designer's knowledge and experience to reach a better solution in less time. The developed tool can also store design cases, learn and use them for future design situations.

A formal building product was developed. This new product model is developed to support the automation of the selection and generation of building assemblies. In the model, the building is broken down into various entities like building assemblies (i.e. external walls, roofs, etc…) and different attributes are assigned to the

entities. The product model is used to facilitate future data exchange with other tools and to develop the structure of the database, which will store the different building data. The database stores two kinds of building data.

Firstly, the description of the design concept description. The design concept description includes all the data of the building before the assembly constructions (e.g. solid painted CMU wall, pre-cast painted concrete panels, etc…) are selected for the different assembly types (i.e. external walls). The second kind of building data stored in the database is the different assembly construction options.

The database also includes the criteria and the constraints governing the assembly construction selection. The different kinds of criteria were identified and examples of each were given. Similarly, the constraints on the assembly construction use and selection were identified and classified into the different types. Examples of each constraint type were discussed.

Selection and generation procedures were then developed. The selection procedure selects the best compromise assembly construction given the set of the different constraints and criteria. The multi-criteria aspect of the assembly selection problem was tackled and a solution procedure was identified. The generation procedure helps the designer in developing the details for the selected assemblies.

Furthermore, several tasks had to be accomplished,

- A study of integrating the automation of assembly selection and generation with the design paradigm.
- A review of research efforts directed towards the later stages of design and specifically assembly selection and generation.
- A review of information modeling techniques and building product models.
- A developed prototype based on the ideas presented in this research

In summary, the approach described in this dissertation provided a contribution by providing a method for intelligently selecting building assembly constructions and automating the generation of the details of the selected assemblies.

8.3. RECOMMENDATIONS FOR FUTURE RESEARCH

The research presented in this dissertation is only a step towards the complete automation support for building design. Many directions for future research can be identified.

In this research we discussed how the selection of the applicable criteria can be calculated automatically based on previous design cases. The computer can be "taught" to recognize which criteria are applicable and which are not as well as which criteria are more important than others for particular design situations. This computer learning is based on the previous design cases stored in the database. A number of hypothetical cases were used as an example. If the developed examples were actual previous design cases it would be useful to consider how the previous cases of different designers can affect the computer learning. For example, previous design cases for two different designers can be collected. Then using the computer learning tool described in chapter 6, the computer would generate the different rules for both design case histories. The generated rules can be compared and analyzed.

Although, the actual domain knowledge presented in this dissertation was only given as examples of the kinds of knowledge that could be defined, an extension of such knowledge is a future research direction. It would be useful to extend the domain knowledge presented here, by for example adding more constraints and criteria to the database.

The extension to the generation operations is another important future research direction. In the current EASYBUILD system only the "place" operation is fully developed. The other operations can be developed so that the Design Generation Definition Module is fully functional. Different kinds of assembly details can be

modeled and stored. Furthermore different construction notes and recommendations can be added to each operation.

An important area for future research and development arising from the concepts presented in this dissertation is machine learning. We have already presented several of the machine learning methods throughout the dissertation. From our review we can divide the machine learning methods that can be used in the construction assembly selection into two main approaches; retrieval approaches and analytical approaches as seen in Figure 8.1. Retrieval approaches like case-based reasoning are concerned with recover form a database the case closest to the one at hand. Analytical approaches on the other hand, like neural networks are concerned with analyzing the data in a database to generate either rules, patterns or equations from the data.

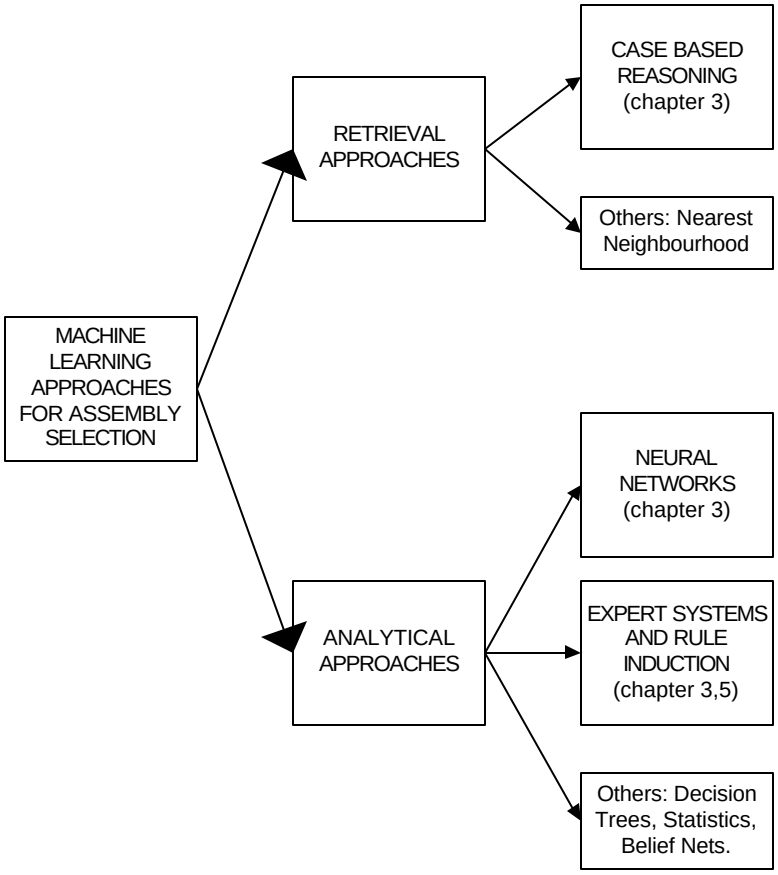


Figure 8.1. Approaches to Machine Learning

In chapter six we showed how machine learning can be used to determine the assembly criteria performance scores for those cases were we can not determine the score by any analytical method and where we have a set of previous cases that can be used to learn from. However there are other ways we can use machine learning to integrate it with our system and use it in the selection of the construction assemblies.

Two focus areas can be developed; firstly, machines-learning in the selection of assemblies as a whole and secondly, machine-learning for the selection of components to build a new assembly. Computers can be made to learn the to select the best assembly constructions given a set of previous cases. The "best" in this case is dependent on the previous cases and not the criteria/weights definition.

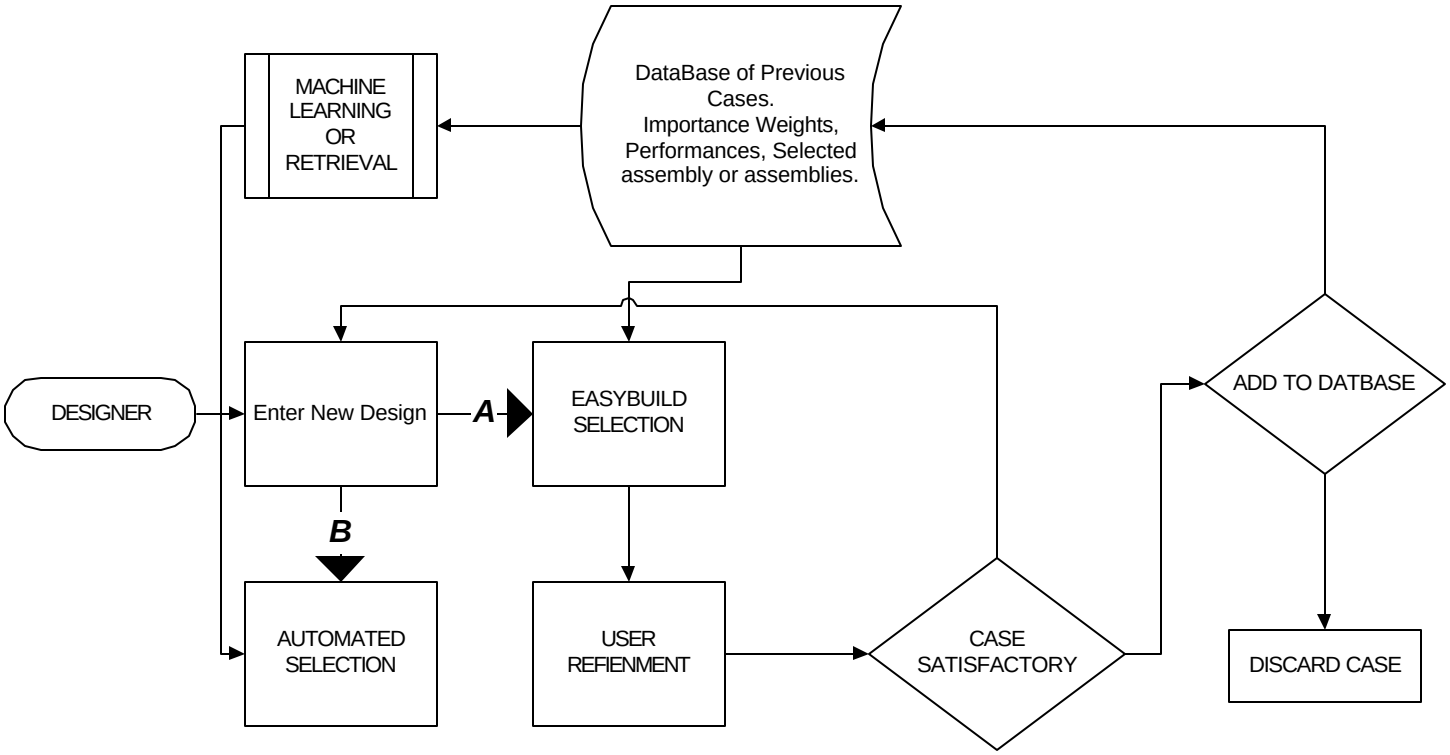


Figure 8.2. Integration of Machine Learning with the developed system

The integration of this idea with the developed system is shown in Figure 8.2. The designer first inputs his design description based on the developed product model. The designer can then have two courses of action; firstly to use the selection procedure defined in the system to select the assembly construction and then revise or modify the selected solution. This solution then becomes part of the case base that is used for machine learning. The other course of action, perhaps taken by a novice designer, is to use any of the

machine learning, retrieval or analytical approaches for automated selection of the assembly construction., which in turn is added to the case base.

Also, instead of performing selection from a set of assembly constructions, the computer can be used to generate a whole new assembly construction. Using the assembly constructions defined in the database, new assembly constructions that have not been used before can be explored.

Very few architects use design support tools when developing the schematic building design. Design firms and architects are usually reluctant to use design support tools because they are usually cumbersome and hard to use. The development of the prototype tool to become a practical every day tool requires the study of actual design practices of architects.

8.4. CONCLUSION

The research presented here is a step towards a broader vision for architectural design in the AEC industry using. We can advance the architectural design of a building from an architect's idea to complete detailed design construction documents in an easier and faster way than current practices using automation. Automation can help designer who might not have insufficient information about many of the possible construction assemblies available for selection or incomplete information on a particular assembly construction. We have presented an automated way to represent the available assembly constructions and their criteria and constraints.

Further more, using the selection and generation presented in this research, lost time due to repetitive processes in design can be saved. The automation of these tasks would save a great deal of time and reduce errors.

Not only will we save the designer's time, but a more complete integration of the design with construction can be achieved on the long run by accelerating the design development stage. The developed detailed design

can be used for different construction planning operations and then easily returned to the designer for any major or minor modification. This is compared to the current practice. Currently, due to the long time it takes to develop detailed designs the construction planning usually does not start until the final set of design details are finished and major changes can not be made in the building design.

REFERENCES

Alcantra, P. (1996), "Development of a Computer Representation of Design Rationale to Support Value Engineering." Civil Engineering. Blacksburg, Virginia Polytechnic Institute and State University

Alexander, C., S. Ishikawa and M. M. Silverstien (1977), "A Pattern Language.", Oxford University Press.

Allen, E. (1990), "Fundamentals of Buildnig Construction." New Jork, John Wiley &Sons, Inc.

Allen, E. and J. Iano (1995), "The Architect's Studio Companion." New York, John Wiley &Sons, Inc.

Alshawi, M. and J. Underwood (1996), "Applying object-oriented analysis to the integration of design and construction." Automation in Construction 5: 105-121.

ASHRAE (1996), "Handbook of Fundamentals.", American Society of Heating Refrigeration and Air Conditioning Engineers, Inc., Atlanta, Gorgia.

Assal, H. (1992), "A New Data Model for Engineering Design." First International Conference of Expert Systems in Development, Cairo, Egypt.

Assal, H. and C. Eastman (1995), "Engineering Database as a Medium for Translation." Modeling of Buildings through their life-cycle: CIB W78 Workshop, Stanford, CA: Stanford University.

Assal, H. H. (1996), "Translation Methodology for Design Databases." Architecture. Los Angeles, University of California

Augenbroe, G. and W. Rombouts (1994), "Extended Topology in Building Design Systems." First Congress on Computing in Civil Engineering, Washington DC.

Ballast, D. (1990), "Architects Handbook of Architectural Detailing." Englewood Cliffs, New Jersey, Prentice Hall.

Bazjanac, V. (1998), “Industry Foundation Classes.” The Construction Specifier(June 1998): 48-54.

Bjork, B. (1989), “Basic Structure of a Proposed Building Product Model.” Computer aided Design 21(2): 71-78.

Brown, R. (1993), “IDEF Methods: Integration Definition For Information Modeling (IDEFX1).”, Knowledge Based Systems, Inc. 1998. http://www.idef.com/.

Cornick, T. and S. Bull (1987), “Expert Systems For Detail Design in Buildings.” CAAD Futures'87, Elsvier Publishers.

Coyne, R. (1991), “Modelling the emergence of design descriptions across schemata.” Environment and Planning B: Planning and Design 18: 427-458.

Coyne, R. and S. Newton (1990), “ Design reasoning by association.” Environment and Planning B: Planning and Design 17: 39-56.

Coyne, R., S. Newton and F. Sudweeks (1993), “A connectionist view of creative design reasoning.” Modeling Creativity and Knowledge-Based Creative Design J. Gero and M. Maher. Hillsdale, NJ., Lawrence Erlbaum Associates.

Coyne, R. and M. Yokozawa (1992), “Computer assistance in designing from precedent.” Environment and Planning B: Planning and Design 19: 143-171.

CSI (1994), “Construction Specification Institute Manual of Practice.”. 1

Cugini, U., F. Folini and I. Vinici (1988), “A procedural system for the definition and storage of technical drawings in parametric form.” Eurographics '88, Nice, France.

Date, C. J. (1995), “An Introduction to Database Systems.” Reading, MA, Addison-Wesley Publishing Company.

DeVries, W. (1996), “Manufacturable Design operations.”. Eindhoven, Eindhoven University of Technology. http://se.wtb.tue.nl/~net/diss_mdo/kopblad.htm.

Eastman, C. (1989), "Architectural CAD: a ten year assessment of the state of the art." Computer-Aided Design 21(5): 289-292.

Eastman, C. (1994), "A data model for design knowledge." Automation in Construction 3: 135-147.

Eastman, C., H. Assal and T. S. Jeng (1995), "Structure of a Product Model Database Supporting Model Evolution." Modeling of Buildings through their life-cycle: CIB W78 Workshop, Stanford, CA: Stanford University.

Eastman, C. and A. Siabiris (1995), "A generic building product model incorporating building type information." Automation in Construction 3: 283-304.

Ekholm, A. (1994), "A Systemic Approach to Building Modelling; analysis of some object oriented building product models." CIB W78 Workshop on Computer Integrated Construction, Esbo, Finland.

Ferleger, S. B. and M. Walker (1987), "Le Corbusier, architect of the century." London, Arts Council of Great Britain.

Flemming, U., C. Baykan, R. Coyne and M. Fox (1992), "Hierarchical generate-and-test vs. constraint-directed search. A comparison in the context of layout synthesis." AID'92, New York, NY, Kluwer.

Flemming, U., R. Coyne, S. Fenves, J. Garrett, *et al.* (1994), "SEED; Software Environemnt to Support the Early Phases in Building Design." IKM'94, Weimar, Germany.

Flemming, U., R. Coyne and R. Woodbury (1993), "SEED: A software Environment to Support the Early Phases in Building Design." ARECDAO 93 (IV Symposium on Computer Aided Design in Architecture and Civil Engineering), Barcelona, Spain.

Garza, J. M. d. l. and P. Alcantra (1997), "Using Parameter Dependency Network to Represent Design Rationale." Computing in Civil Engineering 11(2): 102-112.

Garza, S. and M. Maher (1996), "Design by Interactive Exploration Using Memory-Based Techniques." Knowledge-Based Systems 9(1).

Graham, I. (1994), “Object Oriented Methods.” Wokingham, Addison Wesley.

Gross, M. (1996), “Why cannot CAD be more like Lego? CKB, A Program For Building Construction Kits.” Automation in Construction 2(5): 285-300.

Hannus, M. (1996), “RATAS IT in Finnish Construction.”, VTT. http://www.vtt.fi/cic/ratas/index.html.

Hastak, M. (1998), “Advanced automation or conventional construction process.” Automation in construction 7: 299-314.

Hay, D. (1997), “Nijssen's Information Analysis Methodology.”, Essential Strategies. http://essentialstrategies.com/publications/modeling/niam.htm.

Herbert, R. (1989), “Roofing Design Criteria Options, Selection.” Kingston, MA, R.S. Means Company, inc.

Hopgood, A. (1993), “Knowledge-based Systems.” Boca Raton, CRC press.

Hoskins, E. (1973), “Computer Design Aids for System Building.” Industrialization Forum 4(5): 27-42.

IAI (1997), “Industry Foundation Classes - Release 1.5: IFC End User Guide.”. Washington, DC, International Alliance For Interoperability

ISO (1996), “Industrial automation systems and integration-Product data representation and exchange-Building Construction Core Model.”, ISO JDW/FT

Jeng, T.-S. and C. M. Eastman (1997), “A Database Architecture for Design Collaboration.” TeamCAD Workshop, Atlanta.

Laseau, P. (1980), “Graphic Thinking For Architects and Designers.” New York, NY, Van Nostrand Reinhold Company.

Leeuwen, J. and H. Wagter (1998), “A Features Framework for Architectural Information.” Artificial Intelligence in Design '98, Lisbon, Kluwer Academic Publishers.

Leeuwen, J. v. and H. Wagter (1997), “Architectural Design by Features.” CAAD Futures 1997, Proceedings of the 7th Conference on Computer Aided Architectural Design Futures, Munich, Germany, Kluwer Academic Publishers.

Leeuwen, J. v., H. Wagter and R. Oxman (1995), “A Feature Based Approach to Modeling Architectural Information.” Modeling of Buildings through their life-cycle: CIB W78 Workshop, Stanford, CA: Stanford University.

Leeuwen, J. v., H. Wagter and R. Oxman (1996), “Information Modeling for Design Support - a Feature-based Approach.” 3rd Conference on Design and Support Systems in Architecture and Urban Planning, Spa, Belgium.

Lord, D. (1996), “Environmental Control Systems; Spreadsheet Simulations.”. NewYork, McGraw Hill. http://www.calpoly.com/~dlord/.

Maher, M. (1990), “Process models of design synthesis.” AI Magazine 11(4): 49-58.

Mahoney, W. and F. Horsley (1986), “Means Graphic Construction Standards.”, E. Norman Peterson Jr.

MATURA “The MATURA System.”. The Netherlands, Voorerf 14, Breda

Miles, J. and C. Moore (1994), “Practical Knowledge-Based Systems in Conceptual Design.” London, Springler-Verlag.

Nassar, K. (1995), “A Multi-Criteria Optimization Model for Building Design.” Master's Thesis, Architectural Engineering. Cairo, Cairo University, Faculty of Engineering

Nassar, K. (1998a), “Architectural CAD Product Models.” ACSP'98 Association Schools of Planning, University of Southern California, Los Angeles.

Nassar, K. and Y. Beliveau (1999b), “Intelligent Building Assemblies and Design.” Housing Innovation-Management Business and Marketing Practices, Dallas, Tx.

Nassar, K. and Y. Beliveau (1999c), “Integrating Parametric Modeling and Construction Simulation.” CIB'99, Vancouver, Canada.

Nassar, K. and W. Thabet (1998b), “Intelligent Build Assemblies; Useful tools.” Virginia Academy of Science, annual meeting'98, Washington DC.

Nassar, K., W. Thabet and Y. Beliveau (1999a), “A Building Product Model to Support Intelligent Building Assemblies.” Swiss CAD/CAM Conference, Neuchatel University, Switzerland.

Nijssen, G. M. and T. Halpin (1989), “Conceptual Schema and Relational Database Design.” Sydney, Prentice Hall.

Owen, J. (1993), “STEP an Introduction.” Winchester, Information Geometers.

Radford, A. and J. G. J. S (1988), “Design by Optimization in Architecture, Building and Construction.” New York, NY, Van Nostrand Reinhold Company.

Reid, C. (1980), “An exercise in materials selection.” Metal and Materials July 80: 385.

Rosenman, A. and J.Gero (1994), “A Conceptual Framework for Knowledge-based Design Research at Sydney University's Design Computing Unit.” Artificial Intelligence in Design J. Gero. Southhampton, CMP/Spronger-Verlag.

RSMeans (1997), “Assemblies Cost Data: 22 Annual Edition.”.

Saaty, T. (1982), “Decision Making for Leaders.” Belmont, California, Lifetime Learning Publications.

Schenck, D. A. and P. R. Wilson (1994), “Information Modeling the EXPRESS Way.” N.Y., Oxford U. Press.

Schmit, G. (1990), “Classes of Design - Classes of Tools.” The Electronic Design Studio: Architectural Knowledge And Media In The Computer Era M. McCullough and W. J. Mitchell. Cambridge, Mass, MIT Press.

Shah, J. and M. Mantyla (1995), “Parametric and Feature Based CAD/CAM; concepts techniques and applications.” New York, John Wiley &Sons Inc.

Silvia, N. and A. Bridges (1997), “Human Computer Interaction and Neural Networks in Architectural Design.” CAAD Futures 1997, proceedings of the fourth International Conference on Computer-aided Architectural Design Futures, Munich, Germany.

Sjogren, B.-H. and H. Klausen (1998), “NIAM for Windows Version 3.2., The NIAM Suite for Windows User's Guide.”, IT Integrator

Skinbniewski, M., T. Arciszewski and K. Lueprasert (1997), “Constructability Analysis:Machine Learning Approach.” Journal of Computing in Civil Engineering 11(1): 8-16.

Tonarelli, P., B. Ferries, J. Delaporte and C. Tahon (1997), “Proposal of a product model for building trade.” Automation in Construction 5: 501-520.

Turner, J. (1990), “AEC Building Systems Model.”. Delft, Netherlands

Turner, J. (1992), “Information Modeling and the Design of Databases for Future Computer Aided Building Design Systems.” GDS World 1992.

Turner, J. (1998), “Guide to Reading NIAM Diagrams.”, University of Michigan, Department of Architecture

Wakita, O. and R. Linde (1987), “Professional Handbook of Architectural Detailing.” New York, John Wiley & Sons.

Wix, J. (1997), “Purpose of a Core Model.”. http://helios.bre.co.uk/ccit/info/bccm/purpose.htm.

Yau, N. and J. Yang (1998), “Applying case-based reasoning techniques to retaining wall selection.” Automation in Construction 7: 271-283.

Yu, K., T. Froese and F. Grobler (1998), “International Alliance For Interopability: IFCs.” ASCE Annual Convention, Boston, MA.

APPENDIX A

Permeability

The following definition of permeability is used. The permeability of the assembly is calculated by adding up the resistances of the various components. The resistance and permeability of the various components in the assembly are obtained from [ASHRAE 1996].

$$Mv = MAq\Delta p$$
$$Mv = total\ mass\ of\ vapor\ transmitted$$
$$A = Area, (ft^2)$$
$$q = time\ during\ transmission\ occured\ (h)$$
$$\Delta p = difference\ of\ vapor\ between\ ends\ of\ flow\ path\ (in.Hg)$$
$$M = (Perm)\ permeance\ gr/h.ft^2.in.Hg$$

$$G = \frac{1}{M}$$
$$G = (\text{Re}\, s)\,\text{Re}\, sis\ \text{tan}\ ce (h.ft^2.in.Hg/gr)$$
$$\text{Re}\, sis\ \text{tan}\ ce\ of\ an\ assembly = G_1 + G_2 + ...G_n$$

Procedure For calculating Space Design Thermal Load, ASHRAE CLTD method, (from [ASHRAE 1996]).

$q_1 = UA(CLTD) = roof, walls\ and\ conduction\ through\ glass$
$q_2 = A(SC)(SCL) = Solar\ load\ through\ glass$
$q_3 = UA(t_b - t_{rc}) = Cooling\ load\ from\ partion, ceilings,\ floors$
$q_4 = N(sensible\ heat\ gain)CLF = People\ sensible\ heat\ load$
$q_5 = N(latent\ heat\ gain) = People\ latent\ heat\ load$
$q_6 = 3.41WF_{ul}F_{sa}(CLF) = Lights$
$q_7 = 2545PE_f(CLF) = Power$
$q_8 = (q_{is}F_{ua}F_{ra}CLF)/F_{fl} = Appliances\ sensible\ heat\ load$
$q_9 = q_{il}F_{ua} = Appliances\ latent\ heat\ load$
$q_{10} = 1.1Q(t_o - t_i) = ventilation\ and\ infiltration\ sensible\ heat\ load$
$q_{11} = 4840Q(W_o - W_i) = ventilation\ and\ infiltration\ latent\ heat\ load$
$q_{total} = \sum_1^9 q_i, \qquad i = 1,2,3.....9$

APPENDIX B

EXERPTS FROM THE EXPRSS MODEL

```
SCHEMA DESIGN_CONCEPT;

USE FROM Work_axis
      ();
USE FROM Work_Planes
      ();
USE FROM Work_Point
      ();
USE FROM Assembly_Construction
      ();

ENTITY BuildingType_designation;
      use_group : STRING;
      building_use : STRING;
      building_code : BuildingCode_Name;
END_ENTITY;

(*

Attribute definitions:

use_group:
building_use:
building_code:

CADREP_type

EXPRESS specification:

*)
ENTITY CADREP_type;
       : ;
      has_shape_file : Block_name;
END_ENTITY;

(*

Attribute definitions:

has_shape_file:

CAD_Reps

EXPRESS specification:

*)

ENTITY CAD_Reps
```

```
      ABSTRACT SUPERTYPE OF (External_Walls_Rep ANDOR Internal_Walls_Rep ANDOR
Roof_Rep ANDOR Floor_Rep ANDOR Window_Rep ANDOR Foundation_Rep ANDOR Door_Rep
ANDOR Site_Rep ANDOR Space_Rep ANDOR Slab_on_Grade_Rep ANDOR Column_Rep);
       : CADREP_type;
      ID : INTEGER;
END_ENTITY;


ENTITY Column_Rep
 SUBTYPE OF (CAD_Reps);
       : OPTIONAL ;
END_ENTITY;


ENTITY Construction_Reqiurements
 SUBTYPE OF (Requirements);
       : Construction_Type_Classification;
END_ENTITY;
(*
Attribute definitions:
:
Construction_Type

EXPRESS specification:

*)
ENTITY Construction_Type
 SUBTYPE OF (Explicit_attributes);
      name : Construction_Type_Name;
      is_a : Construction_Type_Classification;
END_ENTITY;

(*

Attribute definitions:
name:
is_a:
Criteria
EXPRESS specification:
*)
ENTITY Criteria
 SUBTYPE OF (Explicit_attributes);
      cost : NUMBER;
      life_cycle_cost : NUMBER;
      Thermal_load : NUMBER;
END_ENTITY;

ENTITY Design_Concept;
      has_derived_attribute : SET [1:?] OF Derived_Attributes;
      has_site : SET [1:1] OF Site_object;
      has_zones : SET [1:?] OF Zone;
END_ENTITY;

(*
```

```
Attribute definitions:

has_CADReps:

has_derived_attribute:

has_spaces:

has_explicit_attribute:

has_site:

has_zones:


Door_Rep

EXPRESS specification:

*)
ENTITY Door_Rep
 SUBTYPE OF (CAD_Reps);
       : OPTIONAL ;
END_ENTITY;

(*

Attribute definitions:

:

Explicit_attributes

EXPRESS specification:

*)
ENTITY Explicit_attributes
      ABSTRACT SUPERTYPE OF (Requirements ANDOR Building_type_designation ANDOR
Construction_Type ANDOR occupancy ANDOR design_team ANDOR location ANDOR
Criteria);
      has_ID : INTEGER;
END_ENTITY;
```

APPENDIX C

In this appendix we will present the EASYBUILD system through a detailed Example. We will use the example that was presented in chapter seven, for more elaboration. The first section of this appendix provides further explanation of how the EASYBUILD system was developed and how it works. The second section presents the example.

EASYBUILD

The EASYBUILD system is made up of three different modules, the Design Concept Definition Module, the Assembly Generation Module and the Database Module. Several files interact with each other to form the complete program. A list of the files used in EASYBUILD is shown in Table C.1. The stand-alone program part of the EASYBUILD system is developed in Visual Basic 5.0 (VB), while the other files are developed in different applications, each of which supports Visual Basic for Applications (VBA). This makes it easier to communicate between the different files and programs. In this section we will first present the different files that make up the system and then we will present the basic program flow.

VISUAL BASIC FOR APPLICATIONS (VBA) PROJECTS

- *Spread Sheet files and VBA*
  - DesignConcept.xls
  - Assemblies.xls
  - Cool.xls, Thermal.xls (external thermal calculation spread-sheets [David, 1996 #103])
- *Visio VBA templates and stencils*
  - EASYBUILD Template, Template.vst
  - EASYBUILD stencil, Stencil.vss
- *AutoCad VBA*
  - EasyBuild.dvb
  - Project1.dvb
  - Drawingtemplate.dwg

STAND ALONE VB PROJECT

- EASYBUILD.vbp, VB Project
- Module1.bas, Modules that Handles Coordination
- VB Forms, MainForm.frm/frx, criteriaform.frm/frx, wieghtsform.frm/frx, spiderform.frm/frx, edit.frm/frx, assemblyform.frm/frx, externalwallform.frm/frx, projectform.frm/frx, options.frm/frx, about.frm/frx, splash.frm/frx, browser.frm/frx

Table C.1. EASYBUILD files

Files and Components of EASYBUILD

The stand-alone part of the EASYBUILD program works as the main interface for the three modules. It allows access to each of the other modules as well as maintaining the shared project information. Developed in VB, it is made up of the regular VB project files (*.vbw, *.vbp, etc….), forms and modules. The application is a single document interface application (SDI). The application is compiled as a standard exe application. There are no class modules since all the entity structures are handled in the Excel database. The Module1.bas handles the coordination of the various forms as well as the main menu. After starting a new project, the designer uses the menu and the toolbar items to draw his design concept in AutoCad. The various forms are also used to get the different data from the user such as the selection criteria and weights. When the forms are loaded they connect to the excel database and get some of the data from there to display it in the appropriate controls, as will be described later.

The excel database DesignConcept.xls and Assemblies.xls files are implementations of the product model developed in chapter five. These files handle the two schemas (*ASSEMBLY* and *DESIGN_CONCEPT* described earlier. The assemblies.xls is made of different tables for the various assembly types. Each table has the set of fields, which are the criteria and the properties of each assembly type. Also each assembly type table has the applicability field described earlier in chapter five. The assemblies.xls file also has the minimum, maximum, offset and scale shift values for each of the criteria associated with a particular assembly type. The assemblies.xls VBA project contains the forms and modules required in handling constraint specification and checking as well as being linked to an external spreadsheet for calculating the thermal load [Lord, 1996].

The DesignConcept.xls file on the other hand contains the information on the *DESIGN_CONCEPT* schema. This includes the different *CADREP* tables and their attributes like height, width, relational attributes to the assemblies.xls referring to the construction type used and to other *CADREPs*. The other tables in DesignConcept.xls are the criteria weights matrix, constraint tables and the performance calculation tables. The DesignConcept.xls also has the VBA project that calculates the criteria weights based on the analytical hierarchical process described earlier in chapter six. The VBA project also incorporates the selection algorithm, which reduces the number of assembly combination selections.

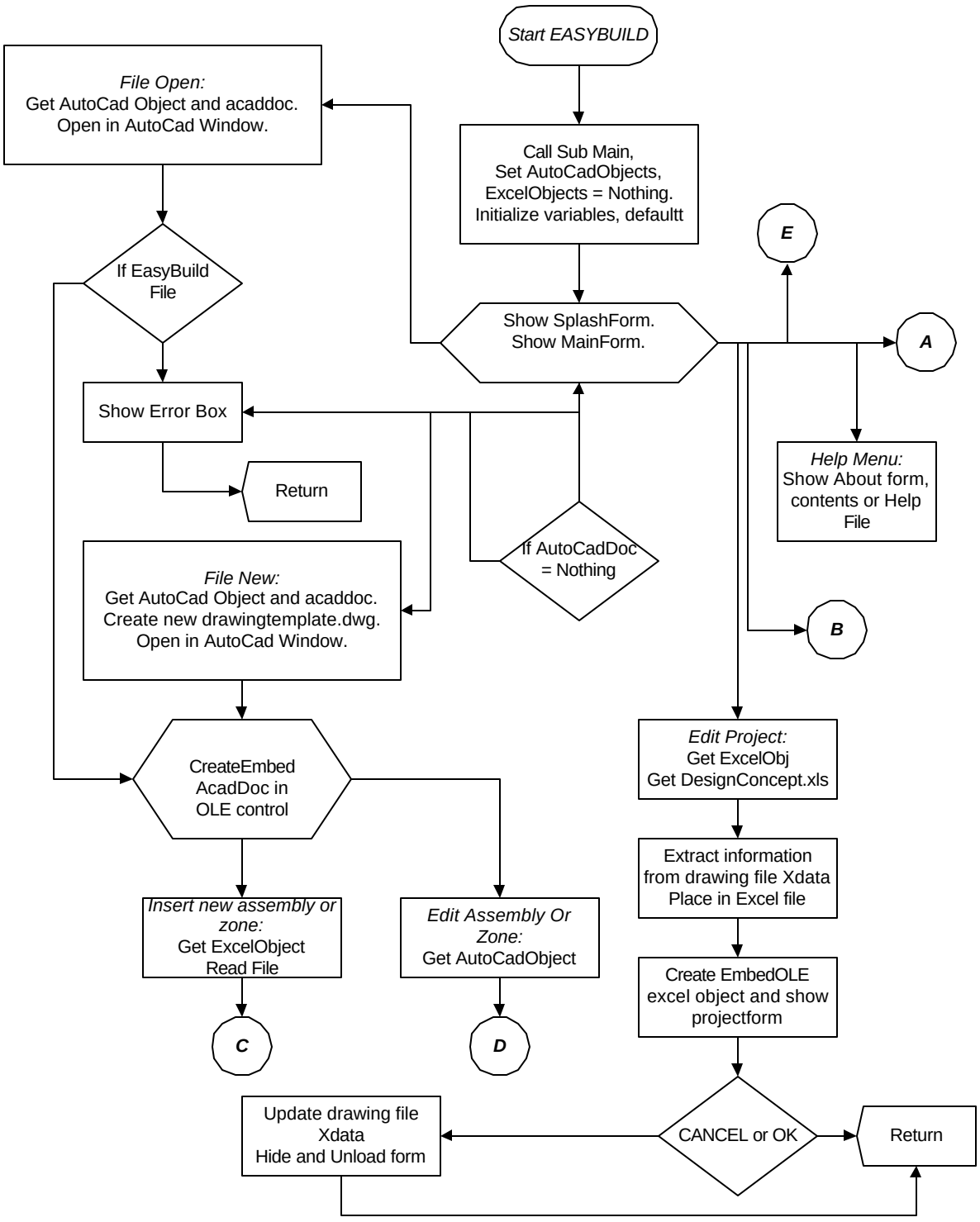


Figure C.1.(A), Basic Flow Chart of EASYBUILD

The DesignConcept.xls VBA project is also responsible for producing the different output plots and the different charts like the spider diagram, which lists the selected assemblies for the top performing solutions. The designer can specify number of solutions to be shown. The DesignConcept.xls is also linked to external spreadsheets for calculating the cooling load, as listed in Table C.1.

The drawingtemplate.dwg is the AutoCad Drawing template setup for EASYBUILD projects. The template provides contains the CADREP types for the different assemblies with the different work features described in chapter six attached as Xdata to blocks. The template also contains the different attributes of the design concept, like the location, occupancy etc… , also attached as Xdata to the drawing document itself. This template is a read only file that is used as the basis for any new projects. The project information is directly linked to the DesignConcept.xls file, and the user can edit the project information from the menu of the stand-alone program. AutoCad also has two VBA projects. The first Easybuild.dvb is responsible for reading the file generated by the Assembly Generation Definition Module (AGDM) produced by Visio, and performing the *Place* operation as described in chapter six. The second VBA project, project.dvb, is responsible for reading the work features names form Xdata attached to the blocks and linking it to the forms of the Visio files.

The Easybuild.vst is the Visio template file used for the AGDM. The template and the Easybuild.vss stencil contain the master shapes that make up the different operations required. Only the place operation is currently functional. The VBA project here is used to get the different parameters for the operations and generate the text file to be read by AutoCad to generate the assembly detail. By dragging and dropping the operation the user is shown a form, which is used to input the parameters of the operation. In the next section we will describe the basic flow of the program.

Basic Flow of the Program

The stand-alone program forms the starting point for an assembly selection and generation session using the system. The basic Flowchart for the EASYBUILD system is shown in Figure C.1(A) and, C.1(B). The start up object is the *sub main* procedure of the main form. The splash and main form are first shown. The user then can either open an existing project file or start a new project.

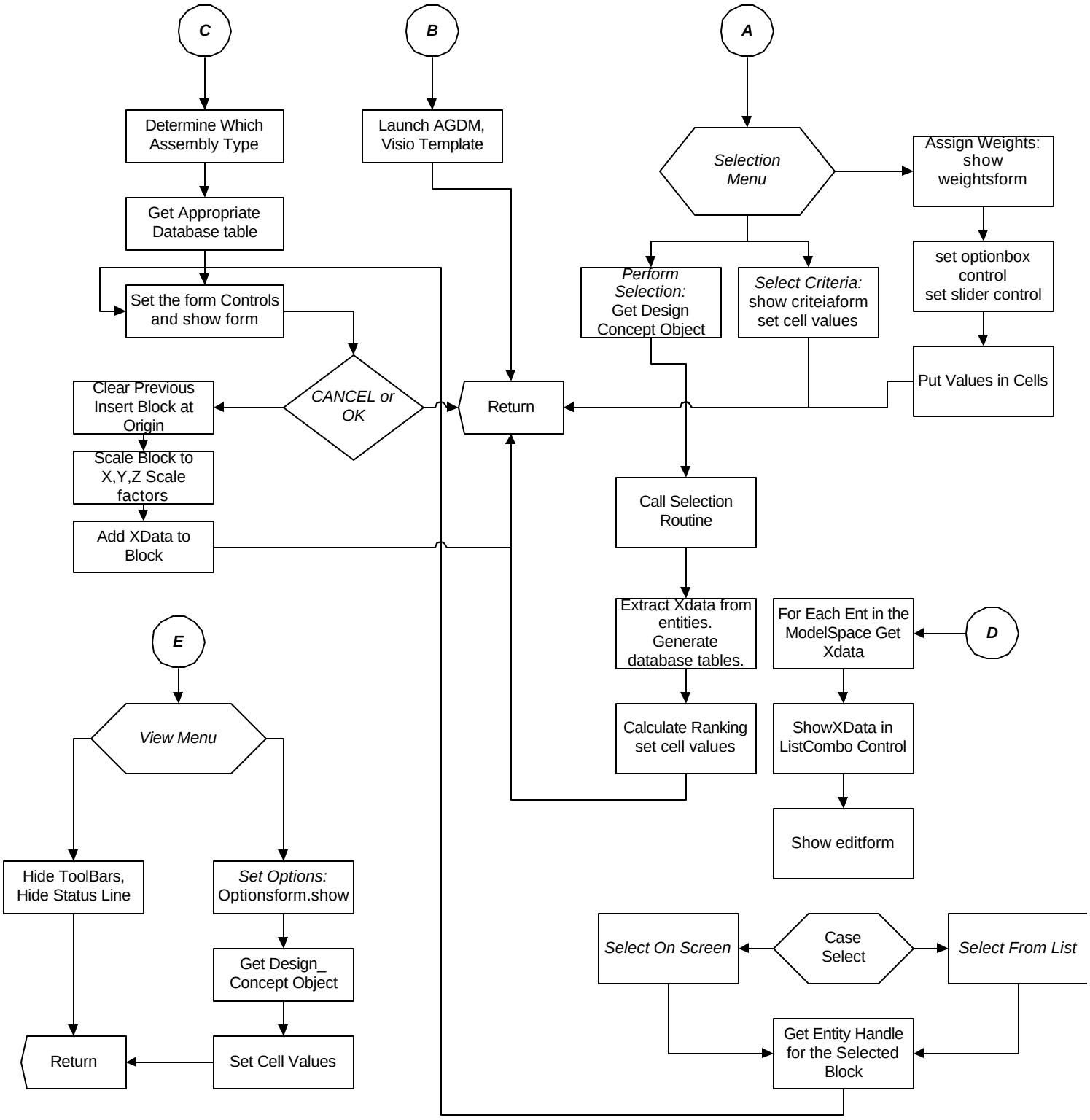


Figure C.1.(B), Basic Flow Chart of EASYBUILD

The files are AutoCad files based on the drawingtemplate.dwg template which has the different data attached to the drawing entities as described above. If the designer opens an existing file and chooses to edit it, the data is read from the graphical entities and displayed in the appropriate forms. A check is carried out to verify that the file in consideration is an based on the template.

After the file is opened the user can start drawing or editing the assemblies to form the abstract design concept. The designer can use the toolbar provided or the menu items to insert a CADREP in the drawing template. The VB project communicates with the database to get the necessary information to display in the different controls on the assembly forms. Before the form is shown and after it is loaded, the database file is opened in the background and the appropriate data is read from the database table and inserted in the list box control.

When the designer inserts an assembly CADREP, the appropriate block is inserted at the origin in the AutoCad read only template file. The designer can then rotate, translate or scale the block to form the design. The different graphical attributes like the width, length, or (x,y,z) location are attached as XData. This means that they are maintained by AutoCad, and can be accessed anytime to transfer to the database.

The designer is also asked to select the criteria and set the criteria weights (or load an old saved set of criteria and weights). The different weights are set in the designconcept.xls automatically and the select routine is called from the database to perform the selection. Then the various output charts and the spider diagrams are generated automatically.

A Developed Example

The EASYBUILD system was developed to aid the designers in the selection and generation of the assembly constructions for a given design concept. In order to present the system, we will discuss in this section an example on the use of the EASYBUILD system. The process of automated selection of the construction assemblies is divided into three main steps; first, the definition and input of the building design concept, and secondly definition of the considered performance criteria and their importance weights. Thirdly, performing the automated selection and review of the results. In the first part of this section, we will present the first step; building design concept definition. The following part will present the second step and finally we will present some of the results of the use of the system.

A. Building Definition

- *Drawing the building in EASYBUILD/AutoCAD Interface*

The first step, as we said, is drawing the building. This is done in EASYBUILD by choosing the required assembly type button on the toolbar or the menu command. This prompts the user with the different information required to draw the assembly like the width and the height and if desired the construction type to be used or any user constraints on the assembly constructions to be used in the case of the external walls. For example, the design concept shown in Figure C.2. is drawn using the various assemblies like the column, foundations, roofs, external walls, etc… The building is a small storage and office space in a physical plant. The building location and other attributes like the occupancy and use are entered by the designer. These will be used in calculating the different criteria and constraints.

- *Editing the Assemblies, setting user-constraints on external wall assemblies*

EASYBUILD also allows for the assemblies to be selected from the AutoCad file and the information attached to the CADREP block can be edited. For example for the design concept shown in Figure C.2., a user defined constraint on the external wall was defined by the designer; the finishing material desired for the external wall assemblies was defined to be plaster.

B. Criteria and Weights

- *Selecting the criteria*

After the building is drawn and the various information about the building is entered, the designer needs to define the "best" solution for the design concept. For example, this building shown will house a critical piece of equipment that produces a high level of noise. This means that STC rating of the external wall assemblies is an important factor, along with the resilience of the used walls and floors. Also the cost of the entire building is to be kept at a minimum and the cooling load are also to be minimized. This set of criteria can be saved and be restored for latter use in similar building situations in the future.

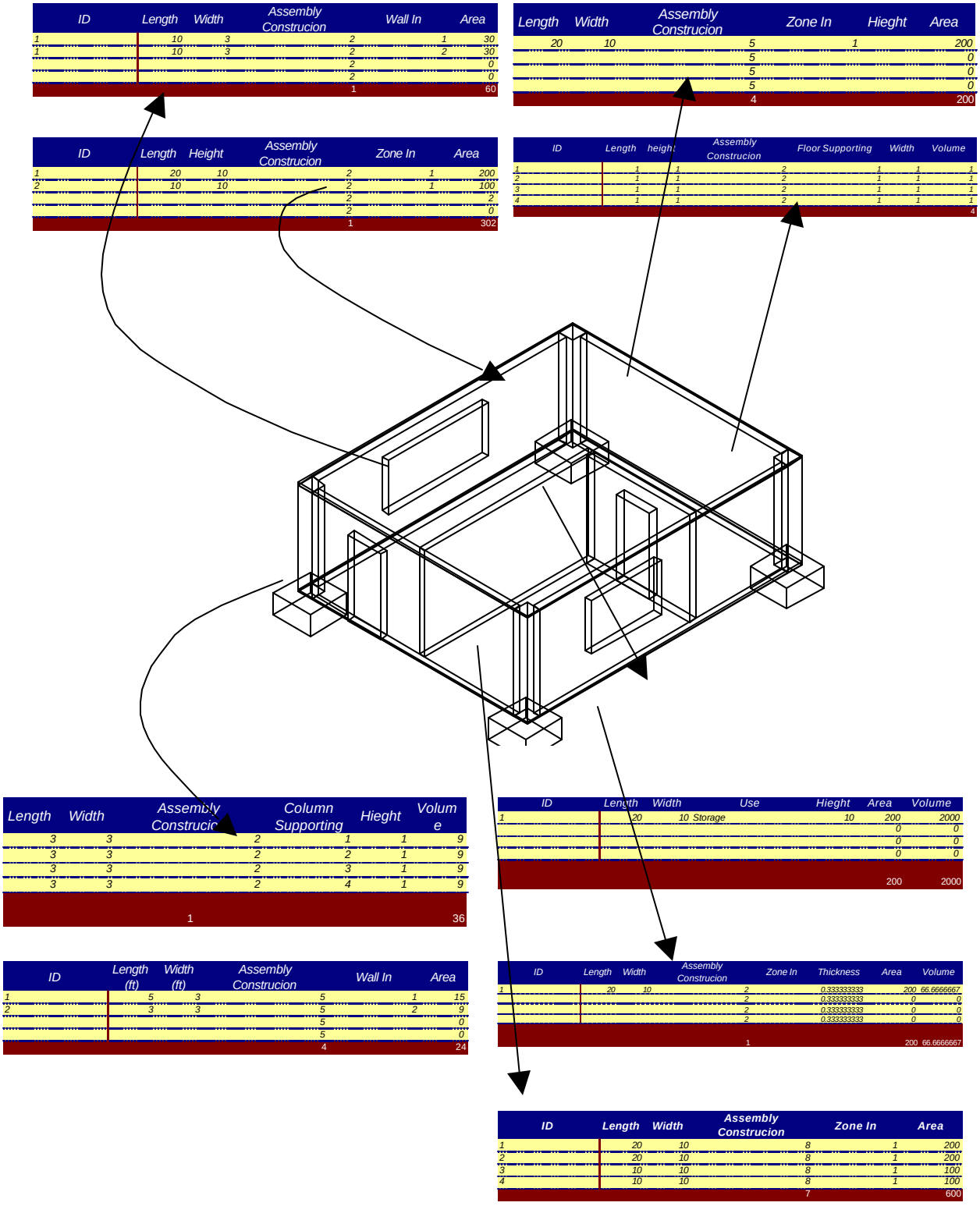


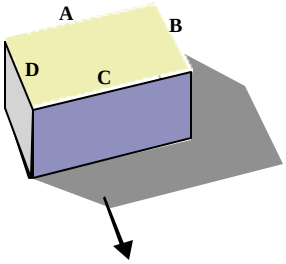
Figure C.2. The Building Model and the generated tables in the database

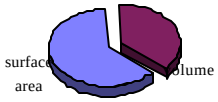
Building Data — Rectilinear

5/23/99 21:08

Storage
Project
Nassar

"C" wall orientation:	N	[N E S W]
"A" dimension:	20	m
"B" dimension:	10	m
story height:	3	m
stories:	1	
number of occupants:	2	min
number of occupants:	3	max
electric power density:	3	W/m² min
electric power density:	5	W/m² max





ground floor area:	200	m²	gross wall area: 180 m²
total floor area:	200	m²	window, % of floor: 2%
total roof area:	200	m²	slab perimeter: 60 m
opaque roof area	200	m²	Surface Area/Volume: 0.63 ratio
volume:	600	m³	max. persons / floor area: 2 persons/10 m²

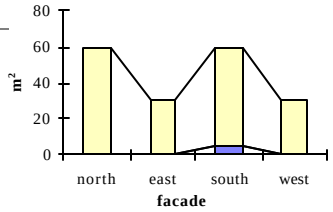
elements

orientation	item	dim1 length	dim2 height	dim3 number	subtotal m²	total m²	glazing / total surface ratio [0 to 1]	totals n	e	s	w
South	"A" Wall	20	3		60					56	
	Glazing	2	1	2		4	.07			4	
	Opaque Wall (wall - window)					56					
West	"B" Wall	10	3		30						30
	Glazing	0	0	2		0	.00				0
	Opaque Wall (wall - window)					30					
North	"C" Wall	20	3		60			60			
	Glazing	0	0	10		0	.00	0			
	Opaque Wall (wall - window)					60					
East	"D" Wall	10	3		30				30		
	Glazing	0	0	2		0	.00		0		
	Opaque Wall (wall - window)					30					
	Skylight	0	0	0		0	.00				

Total Opaque Wall Area:	346
Total Vertical Glazing Area:	4
Ratio of Window to Total Wall Area:	.02

north	glazing	0 m²
east	glazing	0
south	glazing	4
west	glazing	0
opaque north wall		60
opaque east wall		30
opaque south wall		56
opaque west wall		30

glazing-to-opaque wall area



glazing dimensions

		A	B	C	D	summary
north	length			0		0
	height			0		0
	number			10		10
east	length				0	0
	height				0	0
	number				2	2
south	length	2				2
	height	1				1
	number	2				2
west	length		0			0
	height		0			0
	number		2			2

Figure C.3. The Building Data in the Thermal Calculation Spread-Sheet [using Lord, 1996]

- *Setting the criteria weights calculation method*

The default weight for calculating the criteria weights in the EASYBUILD system is the simple percentage method. However, there is another way for calculating the criteria weights in the system, as described in chapter six; the analytical hierarchical process (AHP). The AHP is a pair-wise comparison method as was discussed earlier and usually gives more accurate results but requires slightly more effort for setting. The designer can select which method to use for calculating the weights. So for example, for the building shown the first method was selected.

- *Setting the criteria weights*

Although, the relative importance of the selected criteria is not always stable, i.e. have predefined fixed values, it is important to set an initial set of criteria weights. The designer can enter this set of weights or, similar to the criteria, the design can restore from a file a saved set of weights for a similar previous design situation. The EASYBUILD system starts out with a default set weights were equal importance is assigned to each criteria. So for example, in the building shown the default set of criteria was used as an initial estimate.

- *Setting the criteria-combination method*

The relative importance of the criteria is also influenced by the criteria combination method. The default method in EASYBUILD for considering multi criteria is the addition method. However there are other methods for calculating the combined criteria weights, which can be set using the EASYBUILD system as was described in chapter six. Each method is used for a specific situation and can yield different results, as described earlier. The designer selects the method based on his/her preference. So for example, for the shown building the default method was selected.

	Type	Name	Initial Weight	O_k
Criteria One	Design_Concept	Cooling Load	*0.2*	*3*
Criteria Two	Design_Concept	Material Cost	*0.2*	*0*
Criteria Three	Assembly, External Wall		*0.2*	*2*
Criteria Four	Assembly, Floor	Resilience	*0.2*	*2*
Criteria Five	Assembly, External Wall	STC rating	*0.2*	*3*

Table C.2. The Criteria Data for the Example Building

C. Selection

- *Automated Selection*

The search for the best criteria is carried out in EASYBUILD using the procedure defined in chapter six. This procedure reduces the number of assembly constructions to be considered and therefore reduces the run time. For example, based on the defined building and the design concept criteria and weights, an initial solution was selected. The output results for this initial solution is shown in Figure C.4. The output shows the spider diagram for the set of selected best solutions. In this diagram the different normalized criteria performances (in percentages) are plotted across the different axis. Each solution is plotted as a separate data series. The overall quality of a solution is similar to the *weighed* area under the data series curve for the particular solution. EASYBUILD also provides pair-wise criteria plots for the best solutions as shown in Figure C.5. The plots show the different solutions plotted as data points across two axes. Each of the two axes represents the normalized performance of a solution with respect to one criterion. These different solutions are a result of varying the criteria weights with respect to each other. These allow the designer to view how the solutions perform with respect to each based on changing the importance weights of the criteria.

Another output of the system is the solution performance distribution (significance of range, variance in performance) as seen in Figure C.6. The user can also then try different other runs as will be described in the next step.

- *Modifying Input*

The EASYBUILD system can also be used to consider varying the criteria and then performing automated selection again, for example selection based one assembly-based criterion The designer for example can perform an automated selection based on a criterion that only affects one assembly type. For example in the shown building, the designer might want to select the external wall assembly with the highest STC rating. The EASYBUILD system selects the best assembly, which also does not violate any of the predefined or set constraints. The selected assembly was external wall assembly number 7: "*12in brick, 0.5 plaster on each side*". This is the best assembly construction that satisfies the user defined constraint of plaster finish and other predefined constraints like minimum fire rating based on Code. Note that the rest of the solution set is unaffected.

Another example would be selection based on one design_concept criterion which requires the EASYBUILD system to look for the best assembly from respect of the building as a whole. For example, selecting the external wall assembly for the minimum cooling load. Other possible modifications are,

- Searching for one or more assembly criteria
- Searching for one or more design_concept criteria
- Searching for a tradeoff between two or more assembly-based criteria
- Searching for a tradeoff between two or more design_concept criteria
- Searching for a tradeoff between a generic and assembly-based criteria

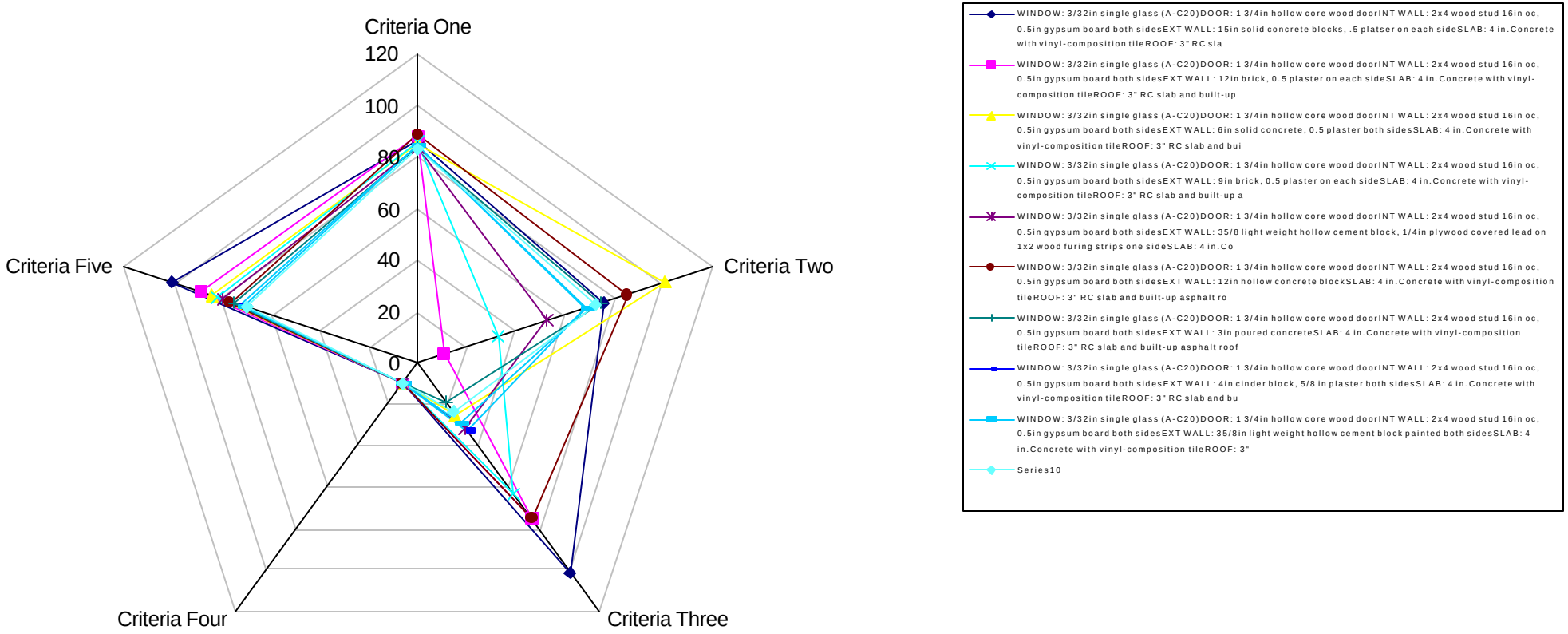


Figure C.4. The Spider Diagram showing the selected best set of solutions

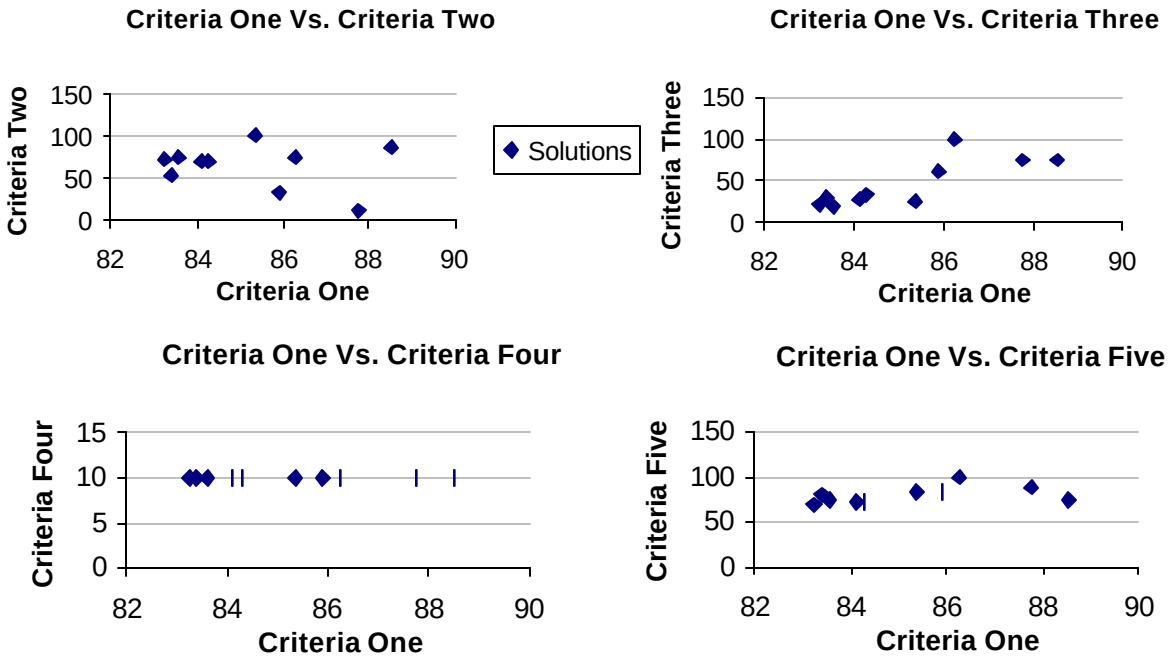


Figure C.5. Pair-wise Criteria Plots

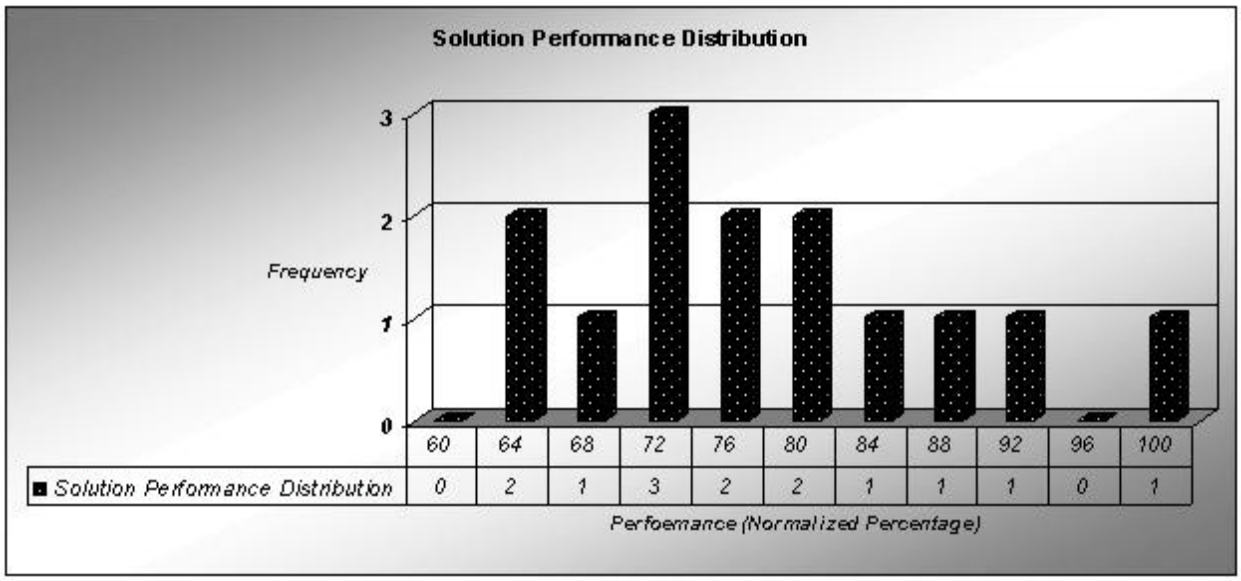


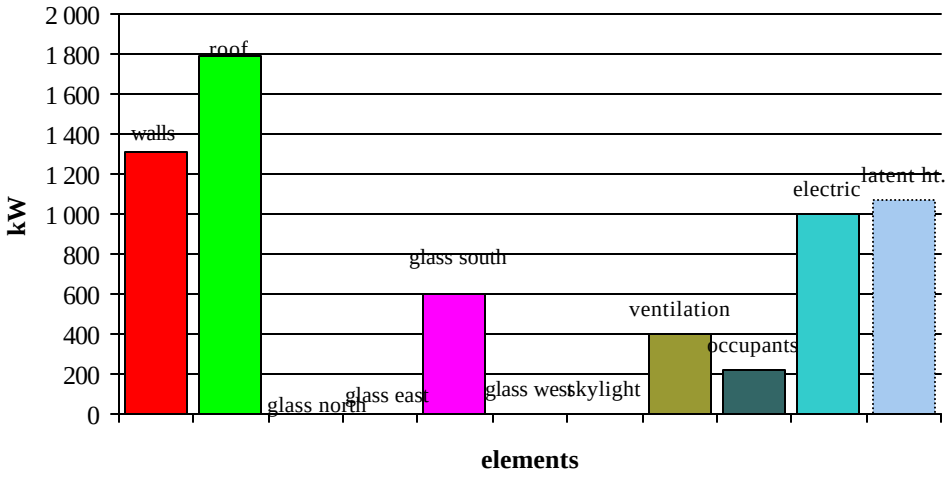
Figure C.6. Performance Distribution for the Various Solutions

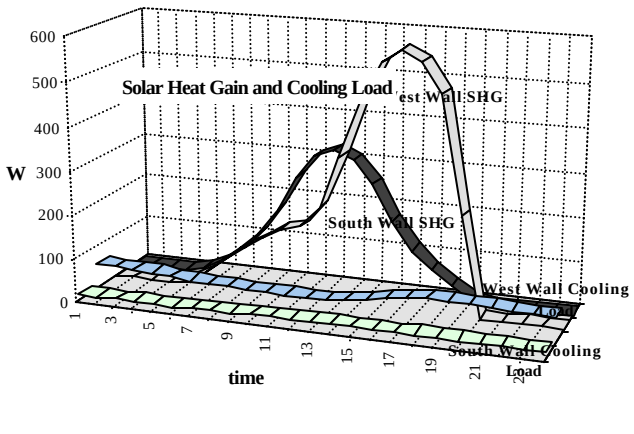
Peak Heat Gain — Storage

5/23/99 21:08 — M.E.E.B., p. 247 ff. — Project: Nassar

summer outdoor design temperature: 34 °C
daily temperature range: 15 °C = High

item	Coefficient or U-factor		Quantity	units	D.C.L.F./ D.E.T.D.	notes	PEAK GAIN W		M.E.E.B. reference
Walls	0.37		346	m²	10.3	frame	1 310	20%	Table 5.19, p. 249
Roof	0.41		200	m²	21.6	blt.up-dk	1 793	28%	Table 5.19
Glass, north			0	m²	82	roller shade	0	0%	Table 5.20
Glass, east			0	m²	280	naked	0	0%	Table 5.20
Glass south			4	m²	151	bare	604	9%	Table 5.20
Glass, west			0	m²	280	single	0	0%	Table 5.20
Skylight		x	0	m²	454	double	0	0%	Table 5.20
L/s Ventilation	8	x	3	people	16.7	*W/(L/s)*	401	6%	
watts per Occupant	75		3	people			225	4%	Table 5.22, p. 252
Maximum Electric Power Density:			5	W/m²			1 000	16%	Table 5.8 B
Latent =	20%		of sensible			latent	1 066	17%	see p. 257
-to-air heat exchanger?	0%		efficiency			sensible	5 332	83%	
						Peak Heat Gains	6	**kW TOTAL**	
[Based on indoor design temperature 24 °C]							32	W/m²	
							31	m²/kW	





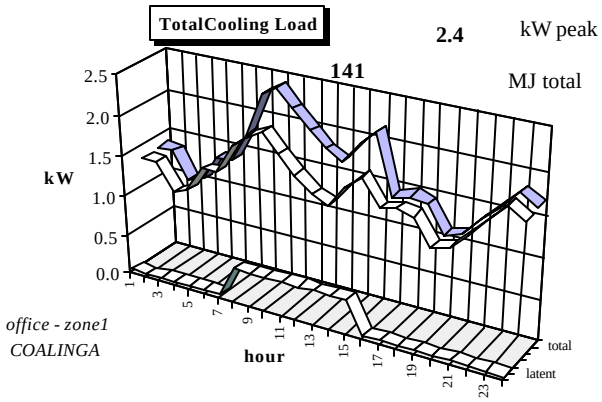


Figure C.7. Results of the Thermal Calculation Spread-sheet [using Lord, 1996]

Biography

Khaled Nassar was born on the 6th of October 1971. After going to the Gezira Language School in Cairo, he joined Kuwait English School in Kuwait City where he received his General Certificate of Education (GCE). Khaled then joined Cairo University's Architectural Engineering Program where he graduated in 1992. Upon his graduation Khaled joined the staff of the Department of Architecture Engineering as a teaching assistant and later as a lecturer. He also started a design build firm, where he designed and built several residential and commercial projects. In 1995, he received his Masters of Science Degree in Architecture Engineering with a thesis on multi-criteria optimization in architectural design. He joined the department of building construction at Virginia Tech in 1996 and started working on this dissertation. Khaled joined Hico Inc., a specialty contractor in Christiansburg Virginia in the summer of 1997. In 1999 Khaled finished his dissertation and accepted am assistant professor position at the School of Engineering of Roger Williams University in Bristol Rhode Island, where he currently lives with his wife and son.