

**AN EXPERT SYSTEM FOR
ANALYSIS OF GRAIN BIN LOADS**

by

Zhiping Ni

A thesis
presented to the University of Manitoba
in partial fulfilment of the
requirements for the degree of
Master of Science
in
Agricultural Engineering

Winnipeg, Manitoba

(c) Zhiping Ni , 1997



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-23441-X

**THE UNIVERSITY OF MANITOBA
FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION PAGE**

AN EXPERT SYSTEM FOR ANALYSIS OF GRAIN BIN LOADS

BY

ZHIPING NI

**A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University
of Manitoba in partial fulfillment of the requirements of the degree**

of

MASTER OF SCIENCE

Zhiping Ni

1997 (c)

Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to Dissertations Abstracts International to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

ABSTRACT

An expert system was developed as a design aid to assist in the design of grain storage bins. The expertise embodied in this expert system was extracted from Canadian Farm Building Code and ASAE EP433. This expert system, developed in Level5 Object expert system shell, consists of knowledge base, databases, and external programs. The knowledge base serves as overall integrator handling user interaction, accessing databases, passing parameters to external programs , and running plotter programs. Databases provide bin geometry and material properties. The external programs present the pressure distribution on screen.

Based on the code selected by user, this system selects design parameters and calculates static pressures and dynamic pressures exerted by stored material, then reduces those pressures to loads on specific components of a grain bin. This expert system can provide bin designers with a rapid and accurate means of determining loads applied to individual bin components.

ACKNOWLEDGEMENTS

I wish to express my sincerest appreciation to Dr. M.G. Britton, my major professor, for his encouragement and guidance during this study.

I would like to thank other members of my advisory committee, Dr. Q. Zhang , Dr. J. I. Glanville , and Dr. D. Polyzois, for their suggestion toward this study.

I would like to thank my wife Yuexian and my son Chong dearly for the patience and understanding when I was working on this thesis. They have taken on the inconveniences imposed by an absent husband and father without complaint.

LIST OF TABLES

<u>Table</u>		<u>Page</u>
4.1	Bolt pattern coefficients	25
4.2	Moisture content of CFBC	43
4.3	Material properties for EP433	44
4.4	Overpressure factors for CFBC	56

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
3.1	Basic structure of an expert system	17
4.1	Structural breakdown of a Level5 Object	27
4.2	Basic structure of this expert system	29
4.3	General flowchart of this expert system	30
4.4	Flowchart of CFBC subprogram	31
4.5	Flowchart of ASAE subprogram	32

TABLE OF CONTENTS

ABSTRACT

ACKNOWLEDGEMENTS

LIST OF TABLES

LIST OF FIGURES

<u>Chapter</u>	<u>Page</u>
1. INTRODUCTION	1
2. REVIEW OF LITERATURE	4
2.1. Loads Applied to Grain Bin Wall	4
2.1.1. Static Loads	4
2.1.2. Dynamic Loads and Thermal Loads	7
2.2. Codes and Standards	8
2.2.1. German Code	9
2.2.2. ASAE Engineering Practice	9
2.2.3. American Concrete Institute Standard	10
2.2.4. Canadian Farm Building Code	11
2.3. CAD in Agricultural Engineering Design	12
3. EXPERT SYSTEM	15
3.1. Introduction	15
3.2. Components of an Expert System	16
3.3. Expert System Development Tool	19
3.3.1. Expert System Shell	19
3.3.2. Level5 Object Expert System Shell	21

4. DEVELOPMENT OF THIS EXPERT SYSTEM	23
4.1. Knowledge Domain	23
4.2. Development Tool	26
4.3. System Architecture	28
4.4. Structure of This Expert System	29
4.4.1. Defining Class Structure	33
4.4.2. Defining Attribute Structure	33
4.4.3. Creating Display	34
5. TEST AND EVALUATION	72
5.1. Testing	72
5.2. Example Application for CFBC 1990	73
5.3. Evaluation of Level5 Object	92
5.3.1. Integrating Expert System Technology with Database Technology	92
5.3.2. Calling External Program from a Knowledge-Base System	93
5.3.3. Knowledge Base Management	97
6. CONCLUSIONS AND SUGGESTION FOR FURTHER DEVELOPMENT	99
LIST OF REFERENCE	102
<u>Appendix</u>	
A. Database for Bin Geometry	
B. Databse for Grain Properties	
C. List of External Program Code	

Chapter 1

Introduction

Many kinds of bulk solids structure have been used in farm for storage and handling of grain over the last century. These include different bin shapes both circular cross-section and rectangular cross-section, and various bin wall materials such as plywood, concrete, smooth steel, and corrugated steel. However, corrugated steel bins with circular cross-section represent the major component of the on-farm grain storage market (Britton and Zhang, 1989). Modern farming practices have increased production of cereal grains and other field crops. Increased grain production and high capacity handling systems have led to large-capacity storage bin. These large storage structures, where bin depth could exceed the diameter, were generally commercially manufactured storage structures which were durable, long lasting and simple to erect. This created considerable pressure on bin manufactures to optimize their designs because overdesign of large storage structures greatly affected the cost of the bin, and underdesign compromised safety.

Although statistics are not available, it has been estimated that about 1000 industrial and farm silos, bins or hopper fail, in one way or another, each year in North America (Jenkyn and Goodwill, 1987). In the meantime, the incidence of structure failure in metal grain bins seems to have increased in recent years. This is maybe a result of the trend to the large storage structure.

Designing structural components of a grain bin involves two steps. First, one must estimate pressures exerted by the stored material and second, select structural members capable of carrying these loads safely and economically. Accurate design of grain storage bins requires the ability to determine loads applied to the bin wall. Many failures are caused at least in part by underestimates of the forces exerted by the bulk solids (Trahair, 1985). To help ensure safe and better-quality bin structure, several countries have adopted codes and standard for bin design. Those most commonly discussed include the German Standard DIN 1055 (DIN, 1987), and American Concrete Institute 313 (ACT, 1983), EP433 (ASE, 1991), and Canadian Farm Building Code (NRC, 1990). Loads due to nature are neglected in most codes because they tend to be much smaller than loads due to grain. Most of these codes use Janssen's equation to predict material pressure. When using Janssen's equation to predict material pressure, the predicted pressure will depend on variables which include bulk density, the coefficient of friction on various surfaces and K value. These variables vary with the type of material stored, the methods of filling and emptying, and the conditions of storage. Therefore, the selection of appropriate variables is critical in the application of the design documents (Britton and Zhang, 1989). Although extensive research has been done on this area, the prediction of grain bin loads is still as much an art as a science. No real evidence can be presented to completely verify the accuracy of any one design theory. The selection of the design method and physical properties to be used is dependent on the personal experience and preference of the individual designer.

Several computer programs which are little more than automated calculations have been developed to predict grain bin loads. However, these programs contain few suggestions as to how

to pick the variables. Most programs give little or no guidance toward the development of optimum solutions.

The job of design could be made easier and more efficient if a computer aid was available that could help guide the designer from the selection of the code and material properties to load analysis.

Expert system is a recently created branch of computer programming which appears to have the potential judgmental decision making. Compared with a traditional program which is mainly dealt with large amounts of data, complex mathematical calculations, and graphical display, expert system which can closely mimic the human approach is well suited to solving problems that are very complex in nature, and generally involve uncertain facts and heuristic knowledge.

The objective of this project was to develop an expert system for analysis of grain bin loads and reduction to individual component loads. The knowledge base was focused on the Canadian Farm Building Code (1990) and ASAE EP433. Only the loads exerted by grain are dealt with in this project. It is intended that this expert system will provide designers with loads applied to specific bin components such as a sheet, a bolt, or a stiffener in order to optimize bin design. On the one hand, this expert system will provide the technical assistance necessary to raise the level of performance of a novice design to the level of that of an expert. On the other, the program can improve the productivity of an expert decision maker.

Chapter 2

Literature Review

Based on the feature of this project, three aspects of literature were reviewed:

1. Loads applied to grain bin walls
2. Codes and standards for the design of grain storage bins, and
3. CAD in agricultural engineering design

2.1 Loads Applied to Grain Bin Wall

2.1.1 Static Loads

Grain pressure theory has long been divided into shallow bin theory and deep bin theory, depending on the ratio of bin height to bin diameter. There are two basic theories for predicting lateral pressure exerted by stored material in shallow bins, i.e., Rankine's theory and Coulomb's theory (Ketchum,1919). Both of theory do not account for friction between the wall and stored material and lead to a lateral pressure linearly increased with depth. Based on Rankine's theory, the theory of equivalent fluid density was developed (shown in Eq.2.1).

$$L = w \times K \times H \times g \quad (2.1)$$

where:

L - lateral pressure

w - grain bulk density

K - ratio of lateral to vertical pressure

H - depth below grain surfaces

g - acceleration due to gravity

This approach was adopted by Canadian Farm Building Code before 1983 edition for predicting lateral pressure in shallow bins. The Rankine's theory was not applicable to bins with rough or corrugated walls because of the assumption of a frictionless wall. For bins with sloped walls, Coulomb's equation is recommended. Robert's experiment (Ketchum, 1919) confirmed there is no increase in bottom pressure after the grain has a depth more than twice the width of the bin. This revealed friction force between the wall and the grain existed. Based on this knowledge, Janssen (Ketchum, 1919) in 1895 developed a mathematical model to predict static loads applied to grain bin walls. This theory was developed by considering the equilibrium of a slice of grain bin wall. It accounts for the arching effect within the grain mass and transfers a part of the grain weight to the bin's wall. By assuming a constant density, w; coefficient of wall friction, μ , and a constant value of the ratio of lateral to vertical pressure, k, he was able to solve the differential equation to yield:

$$L = \frac{\omega \times R}{u} \times \left(1 - e^{-\frac{K \times \mu \times H}{R}} \right)$$

$$V = L / K \quad (2.2)$$

$$F = (\omega \times g \times H - V) \times R$$

where:

L - lateral pressure against the bin wall

V - vertical pressure acting on the bin floor or within the grain mass

F - vertical loads per unit of a wall perimeter due to friction

μ - coefficient of friction between the fill material and bin wall

K - ratio of lateral to vertical pressure

ω - unit weight of grain

R - hydraulic radiuses

H - depth below grain surfaces

g - acceleration due to gravity

Assuming the pressure on the wall as due to a wedge of grain between the wall and plane of rupture, Airy (1897) advanced a method for computing lateral and vertical loads of a bin. Airy's theory which considered both shallow bins and deep bins is basically an expansion of the Coulomb's theory. In his book, Ketchum (1919) indicated that both Janssen's solution and Airy's solution agree very closely with experiments. However, the Janssen's theory become the more widely used in view of its relative simplicity (Britton, 1969).

There are other theories for predicting the static loads in deep bins. One theory of worth mentioning here is Reimbert's theory. Using an empirical basis, Reimbert and Reimbert (1956) considered the ratio of lateral to vertical pressure to be a function of the height of the stored material and proposed a set of prediction equation for static loads. This theory is more compatible with real

storage than Janssen's and Airy' theories because of the consideration of a surcharge effect.

In order to use Janssen's equation properly, it is necessary to use valid values of the factors K and μ . In the derivation of his theory, Janssen (Ketchum,1919) assumed the pressure ratio K is a constant. However, some researchers have speculated that K is not a constant. Ketchum and Willians (Ketchum,1919), Kramer (1944) , and Lenczner(1963) all found that K increased with increasing depth of grain. On the other hand, Pleissner (Ketchum,1919) found that the pressure ratio decreased with depth. Jaky (1948) and Reimbert and Reimbert (1976) found K to be variable but following no simple pattern with increasing depth. Versavel and Britton (1986) found that K decreased with increased overpressure.

Design values for coefficient of friction of agricultural materials on various surfaces are not widely available. The coefficient of friction varies not only with each type of grain, but also with the experiment condition such as moisture content (Abdel-Sayed et al. 1985). Because of the variable surface and test methods, the tabulated values of coefficient of friction are of limited use (Versavel and Britton, 1986). The selection of the coefficient of friction will depend on the personal experience of individual designer.

2.1.2 Dynamic Load and Thermal Load

Bin failures have caused serious doubts on the scope and adequacy of static loads based on design theories (Britton and Zhang, 1989). Although a definition of dynamic effect was not given,

early researchers found the lateral pressure due to moving grain are materially increased on the side opposite the gate and slightly decreased on the side in which the gate is placed (Ketchum,1919). Experimental measurement of the dynamic stress in the model thin-walled, flat-bottomed, grain bins during centric discharge shows that no dynamic overpressure occurs in shallow bins ($H/D = 1.25$) while significant overpressure develops in deep bins ($H/D = 5.0$) (Manbeck et al. 1977). Dynamic loads during discharge have been identified to be the major cause of structural failures of grain storage bins (Jenike and Johanson, 1968). Although experimental work has revealed higher loads develop during emptying of grain storage bins, no coherent mechanistic explanation and adequate prediction theory have been advanced (Britton and Zhang, 1989). Therefore, in most modern design codes and standards, dynamic loads are simply predicted by multiplying static loads by overpressure factors.

Temperature drops result in a contraction of the steel wall to cause thermal loads. Although much experimental work has been carried out to determine the magnitudes of the thermal loads (Britton, 1973; Manbeck and Muzzelo, 1985; Blight, 1985), no practical prediction theory has been developed. Therefore, some design codes and standards recommend the use of thermal overpressure coefficients for predicting thermal loads in steel bins.

2.2 Codes and Standards

Bin failures tend to be the result of under estimates of the forces exerted by grain on the bin wall, eccentric loading and unloading problems, over estimating structural resistance of members,

or inaccurate structural performance characteristics of structural members (Trahair, 1985). To help ensure safety and better-quality bin structures, several countries have adopted codes and standards for bin design. These most commonly discussed include the German standard DIN 1055 (DIN 1987), the American Concrete Institute Standard 313 (ACI, 1983), ASAE Engineering Practice EP433 (ASAE,1991), and the Canadian Farm Building Code (NRCC,1990).

2.2.1 German Code

First published in 1964, the German Code DIN 1055 was based largely on earlier works by Pieper and Wenzel (Safarian,1985). This code was revised with supplementary provisions in 1977. Current code (DIN 1055, 1987) recommends the use of Janssen's equation for calculating the material pressure, and with modified values of the factors, K and μ , to calculate the dynamic loads. It specifies separate value of K and μ for filling and emptying as follows:

- 1) for the case of filling $k = 0.5$, $\mu = \tan(0.75 \times \phi)$; and
- 2) for the case of emptying $k = 1.0$, $\mu = \tan(0.6 \times \phi)$.

where: ϕ - the angle of internal friction of the stored grain.

In general case, the filling condition results in the large values for vertical pressure, and the emptying condition results in the large values for horizontal pressure(Abdel-Sayed et al., 1985).

2.2.2 ASAE Engineering Practice

First edition (EP433,1989) was adopted by ASAE in December 1988. This code was revised

editorially in February 1991 and approved by ANSI as an American National Standard in September 1991. In this practice, the static pressures are calculated by Janssen's equation. Based on the flow pattern, material movement in funnel flow bins ($H/D < 2$) occurs in a center core of the mass, and overpressures are not generated. However, the pressures in mass flow ($H/D \geq 2$) bins are greater than those static pressures predicted by Janssen's equation. This practice specifies that an overpressure factor of 1.4 may be used in mass flow. Based on the effect of the stationary grain along the bottom of the bin wall, a reduction in the overpressure factor is allowed within a distance of $D/4$ from the base of flat bottom bins. This practice specifies a maximum value of 834 kg/m^3 (52 lbs/ft^3) of bulk density instead of a various density of stored material, and suggests that the thermal pressure may be estimated as 8% of the static pressure for temperature declines of $10 \text{ }^\circ\text{C/h}$ and 15% for $20 \text{ }^\circ\text{C/h}$.

2.2.3 American Concrete Institute Standard

In 1977, the American Concrete Institute published its standard "Recommended Practice for Design and Construction of Concrete Bins, Silos, and Bunkers for Stored Granular Materials and Commentary". A revised edition (ACI, 1983) of this standard appeared in 1983. This code recommends the use of either Janssen or Reimbert's (1976) equations for calculating static pressure. When using these two equations, caution must be taken the dimension defined by each method. This code uses a slightly modified form of Janssen's equation for vertical friction force. Dynamic lateral pressure is obtained by simply multiplying the static pressure by the appreciated overpressure factor. This code suggests various overpressure factors (between 1.5 and 2.0) for bins of different height-

diameter ratios. The difference in the values of overpressure factors for two methods (Janssen's and Reimbert's) tends to bring the results into closer agreement, although the Reimbert's pressure will still be larger than the Janssen's lateral pressure in the upper position of shallow bin, and small in the lower portion of deep bin (Gaylord and Gaylord, 1985). The effect of buckling stresses is not considered in this code because it is only applicable to concrete bins.

2.2.4 Canadian Farm Build Code (CFBC)

Early Canadian Farm Building Code (NRCC, 1977) recommends the use of Janssen's equation for deep bins and equivalent fluid density equation for shallow bins which are defined that depth of grain is less than or equal to bin diameter. This code does not consider the effect of a surcharge, a regular occurrence in agricultural storage structures.

A revised edition (NRCC, 1983) appeared in 1983 recommended Rankine's equation for predicting lateral load for bins in which the ratio of grain depth to bin diameter is 0.75 or less. Under the condition of " heaped fill", the predicted loads are increased by a factor of 1.33. Although Rankine's theory does not account for friction between the wall and stored material, this code considers the vertical load to be equal to the lateral load multiplied by the coefficient of friction between the wall and stored material. Before 1983 code, nothing beyond the static loading condition is considered.

In the current Canadian code (NRCC, 1990), static loads in both deep and shallow bins with

vertical walls are predicted by Janssen's equation. This code specifies that an overpressure factor 1.4 to 1.6 may be used for central discharge and 2.0 to 2.5 for eccentric discharge. This code considers an increasing of 6% for the bulk density due to consolidation. It is applicable to variety of stored material and wall materials.

2.3 CAD in Agricultural Engineering Design

Several computer programs were developed to predict grain bin loads. Jofriet and Negi (1988) developed a program to aid in the rapid determination of a pressure diagram and of the resulting hoop force. Based on the silo geometry, model of unloading operation, and the moisture content of the silage, the program carries out the various design calculations and presents a summary table of the output and a graph for pressure distribution. Moran (1991) developed an interaction computer program to predict the static design crop loads on shallow, circular grain storage structures. In accordance with the options and solution parameters specified by the user, the program selects the appropriate solution procedure. This program makes it possible to take more parameters into account than classical theory such as Coulomb's theory. A computer technique (Ross et. al., 1979) has been developed to estimate the wall pressures and average bulk density of materials in bins. The program uses Janssen's approach where the material properties and K value are allowed to vary within the bin as functions of the vertical pressure and the moisture content.

The programs mentioned above are based on well- defined domain knowledge. However, many engineering problems involve uncertain factor and heuristic knowledge. A knowledge-base

approach to problems solving is appropriate if the problem to be solved is not well-defined, the knowledge is incomplete or uncertain (Brzezinski, 1993).

Expert system is a recently created branch of computer programming where appear to have the potential judgmental decision making. The first commercial expert systems were developed in the early 1980's and they were the first successful real-world AI application (Turban and Watkins, 1988). Early expert systems were developed using a programming language such as LISP or PROLOG. Often they had to be run on Lisp-machines or other expensive hardware. This lead to a poor application of expert system in real world. With the emerge of expert system shell in the middle of eighties, expert system technology was becoming cheaper and more affordable. There have been successful agricultural-related applications of expert system reported in the literature (Doluschity and Schmisser, 1986). For this thesis, literature review will now focus on the application of expert system in relative design.

ASD developed by Watson and Brook (1990) is an expert system for the design of aeration system for flat grain storage. This expert system consists of interview, calculation, design drawing, and management recommendation components. ASD offers the capability of rapidly designing an aeration system and changing design guidelines to study the effects upon the design.

MET-X-Pert is an expert system for optimizing designs of metal-clad post-frame building (Gebremedhin et al., 1989). The novel part of the system was the development of a procedure for linking and sharing information between the conventional CAD program and the expert system.

First, a post-frame structure including diaphragm action was accomplished by using a FRAME program. Then, the solutions (stresses and deflections) obtained from the FRAME program were used by the knowledge base of the expert system. The expert system optimized the design based on specified rules. This system was directly developed in Turbo Pascal rather than a shell due to the limitations of the shells.

Embleton (1990) successfully embodied the design code and design routine of snow loads and wind loads into an expert system shell (PcPLUS). An external program was developed to perform mathematical operations that could not be handled in the shell environment. The database files of weather data were accessed through the PcPLUS-DOS link. Although this resulted in extra work for the system developer and slowed the speed of consultations, it provided a means to avoid shell limitations.

An expert system for the fire protection requirements of the National Building Code of Canada 1990 was developed by Olynick (1993). The expert system development tool used in this study was PcPLUS. This system can closely mimic the human approach used in the fire protection analysis. One of the important aspects of this study was the development of decision trees for each fire protection topic. The decision tree approach made programming relatively simple and effective.

Chapter 3

Expert System

3.1 Introduction

An expert system (ES) is a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice (Jackson, 1990). Expert system technology evolved as a result of many years of attempts by Artificial Intelligence (AI) researchers to create real-world AI application. The first commercial expert systems were developed in the early 1980's and they were the first successful real-world AI application (Brzezinski, 1993). These expert systems were usually developed in LISP, PROLOG, or other AI-special programming languages, and often they had to be run on LISP-machine or other expensive hardware. Dedicated hardware together with specialized programming languages which were difficult to interface to the external computing world, resulted in a poor application of expert system. In the middle of eighties, the first expert system development tools capable of running on personal computer appeared in the market place. This resulted in increasingly applications that were economically justifiable on inexpensive hardware.

Expert systems share some characteristics with traditional programs. Both kinds of programs reach conclusions. Similar to expert system to some extent, some traditional programs are interactive and communicative with users in a natural language. However, there are several fundamental

differences. Conventional computer programs are based on algorithms or clearly defined procedures that solve a problem directly. Development of a conventional application typically is based on a classic design - implement - test cycle (Brzezinski, 1993). Every aspect of the application must be defined in terms of a precise algorithm. This machine based calculation are very fast and accurate. However, user interaction is difficult. In contrast to conventional programming technology, Expert system techniques focus on representing and manipulating symbolic data. By encoding the knowledge of domain experts, expert system can simulate human approach to solve a real-world problem. Development of a knowledge-base system does not start with a complete design, instead incremental development and rapid prototype techniques are used. As opposed to a conventional system, knowledge is explicitly separated from the problem solving part of a knowledge-based system. This enables easier modification and maintenance of knowledge. Therefore, expert system technology is best suited to solving problems that are very complex in nature, and generally involve uncertain facts and heuristic knowledge. On the other hand, tasks involving large amounts of data, complex mathematical calculations, graphical display are better suited to conventional programming technology. However, real-world problems involve both types of tasks. In order for expert system technology to be useful in solving problems, it must be integrated with conventional programming technology. In recent years this requirement has been realized and various expert system packages have appeared on the market capable to varying degree of integrating with conventional programming technology.

3.2 Components of an Expert System

The knowledge base contains the basic knowledge for understanding, formulating, and solving the problem, including facts, rules, and methods.

The inference engine controls the execution of the system, and determines how to solve a particular problem. It uses the knowledge base to modify and expand the contents of working memory. Most expert systems are based on backward or forward chaining, and some are based on both rules and methods. In backward chaining, the system begins with the desired goal of the system and moves toward the requisite conditions to satisfy this goal. On the other hand, forward chaining uses the known conditions to work toward the desired goal of the consultation.

The user interface allows a user to communicate with the system and creates and uses a database for specific cases. Its basic function is to gather information from the user and present the results of a consultation session back to the user. The features of a user interface include data input, reporting, natural language modules, and graphics displays. These items determine the ease with which a user can learn and use a program as well as the ease of program development. Therefore, the capability of a user interface is the main factor by which the performance of an expert system is evaluated.

The working memory is a part of the knowledge base. It initially contains user specified facts, but as the system reasons with the data, facts may be added, modified, or deleted. The process concludes when the desired problem solving process ends.

The explanation subsystem provides the means for the expert system to explain to the user how it arrives at a particular conclusion or why a particular question is asked. It is important during the development and debugging of an expert system and consultation.

The knowledge acquisition subsystem is used for developing and modifying the knowledge base. It consists of a knowledge editor and an induction tool. The editor ensures that knowledge data are properly configured to the form required by the inference mechanisms of the system. Induction tools facilitate data transfer to the ES program from external information sources, e.g. database and analytical programs.

The above-mentioned six components are fundamental to all expert systems. Of the six components, only the knowledge base is problem specific. Early expert systems were developed using AI-specific programming language for six components. It soon became apparent that only the knowledge base was different for various problems, and all other components were the same in every expert system. These components or programs thus became known as shells.

3.3 Expert System Development Tool

3.3.1 Expert System Shell

The expert system shell is an expert system with an empty knowledge base. It consists of all the basic components needed to support an ES program but the problem specific knowledge.

Generally speaking, an expert system shell is application independent. Once constructed it can be reused in many applications. On the other hand, the knowledge base determines what problems an expert system will be able to solve.

By using the shell approach, the task of an expert system developer is to translate domain knowledge (from human experts, textbook, database, literature, and reference books, etc.) into the shell. Therefore, expert system shells greatly simplify and speed up the creation of an expert system, because they alleviate the programming problem and allow the developer to concentrate on building the knowledge base. There are many different types of expert system shells available now and each of them has its strengths and limitations. For example, some shells support only rule-based knowledge representation, such as VP-Expert, PcPLUS, EXSYS expert system shell. In this category some shells provide only backward chaining or only forward chaining inference mechanisms. Other shells support both rule-based and frame-based knowledge representation such as Level5 OBJECT and KES expert system shell. Shells vary from one to another with respect to knowledge base capacity, price, inference mechanisms used, facilities to make use of external information sources, graphics capabilities, and explanation facilities. Despite the wealth of knowledge accumulated about constructing expert system, choosing an appropriate tool for building a particular system remains a difficult yet crucial. A tool that in some sense well suits a particular problems area can facilitate the development process, shorten the development time, and lead to a finished product that performs with a high degree of efficiency. Four key shell characteristics should be reviewed when a shell is chosen (Embleton, 1990):

---- the ease of graphics incorporation

---- the screen formation features

---- the ease of knowledge base entry and management, and

---- the ease of external access to other programs and data files.

The selection of the shell should be based on the problem to be solved, the abilities of the developer, and the needs of the user.

3.3.2 Level5 OBJECT Expert System Shell

Level5 OBJECT, developed by Information Builders, Inc., is the expert system shell used to develop this expert system. It is a Microsoft window-based expert system shell which allows both rule-based and frame-based (object-oriented) knowledge representative.

Hardware requirements:

- * An IBM PC or 100% compatible with a 286 or above processor.
- * A hard disk drive with a minimum of 4MB free hard disk space.
- * At least 2MB of memory.

Software requirements:

- * Microsoft windows version 3.0.

The main features of Level5 OBJECT are:

- 1) Both rule-based and frame-based (object-oriented) knowledge representation.
- 2) English-like rule syntax.
- 3) Three different methods of integrating with database technology.
- 4) Dynamic data exchange.

- 5) Ability to call an external program or function from within a knowledge-base system.
- 6) Client-server architecture.
- 7) Very high-quality knowledge management facilities.

Chapter 4

Development of the Expert System

4.1 Knowledge Domain

The Canadian Farm Building Code (1990) and EP433 (ASAE, 1991) are the principal sources of knowledge used to build the present expert system. Both codes recommend the use of Janssen's equation to calculate static loads. CFBC (1990) specifies a central discharge overpressure factor range from 1.4 to 1.6 and an eccentric discharge overpressure factor between 2.0 and 2.5. EP433 specifies that an overpressure factor of 1.4 may be used in mass flow bins ($H/D \geq 2$). Because the corrugated steel grain bin is the domain system in the market, development efforts were restricted to corrugated steel bins with circular cross-sections.

For cylindrical bins, calculation of three "grain induced" pressure is important (Fankhauser, 1977). These pressures are:

1. Horizontal pressure creating hoop tension in the bin walls.
2. Vertical pressure acting on the cross-sectional area of the bin foundation or grain mass.
3. Vertical wall load introduced into bin side walls through friction between the grain mass and corrugated sheets.

Fankhauser suggested that the design of the bin shell was primarily dependent on the magnitude of

horizontal pressure and vertical wall load, and the vertical pressure was used for foundation analysis. Since the purpose of expert system related to bin shell design, attention was focused on horizontal pressure and vertical wall load. Those material pressures were determined by Janssen's equation. This expert system will reduce those pressures to the loads on specific bin components.

Horizontal pressure due to the stored material is assumed to be uniform around the circumference of the bin at any particular grain depth. Pressure is converted to hoop tension (T) at any specified grain depth based on the relationship:

$$T = p \times D / 2 \quad (4.1)$$

where

T = hoop tension at a specified grain depth (N/m)

p = horizontal pressure at the specified depth (N/m²)

D = bin diameter (m)

Hoop tension can then be specified as a tensile force in any given tier of bin sheets. This value can be used to specify the required wall sheet thickness at any vertical location in the bin wall. Bin sheet information with regard to the pitch of the corrugations allows hoop tension to be reduced further to the tensile force applied over the vertical dimension of a single corrugation. A table of bolt pattern coefficients (Table 4.1) was developed to define the number of bolts that would be available to carry this force (Ni et al., 1994). Shear force carried by a single bolt is specified as:

$$S = C \times T \times d \quad (4.2)$$

where:

S = shear force applied to each bolt (N)

T = hoop tension at a specified grain depth (N/m)

d = sheet pitch (m)

C = constant related to bolt pattern, given by Table 4.1.

Table 4.1. Bolt pattern coefficients

Bolt Pattern	C
one row of bolts with space equal to one corrugation pitch	1
one row of bolts with space equal to half of a corrugation pitch	2
two rows of bolts with space equal to one corrugation pitch	2
two rows of bolts with space equal to half of a corrugation pitch	4

It is assumed that vertical wall load will be transferred from the wall panel to the stiffener through bolted connections, and this load is considered to be taken entirely by the stiffeners

(Schott, 1990). The vertical load per stiffener is:

$$P = \pi \times D \times F / N \quad (4.3)$$

where:

P = vertical load in a single stiffener (N)

D = bin diameter (m)

F = vertical wall friction (N/m)

N = numbers of stiffeners around bin circumference.

4.2 Development tool

Level 5 Object, developed by Information Building Inc. (1990), is the expert system shell used to develop this design aid. Level5 Object is a Microsoft Window-based expert system shell which allows both rule-based and frame-based (object-oriented) knowledge representation. Level 5 Object was selected because of the object-oriented style which seems to adapt well to engineering design. This is a good learning tool for new developer to catch the idea how to quickly develop a prototype expert system.

Level5 Object is a hybrid application development tool that integrates object-oriented techniques and expert system technology with traditional, procedural programming. The structural breakdown of a Level5 Object is shown in Fig.4.1.

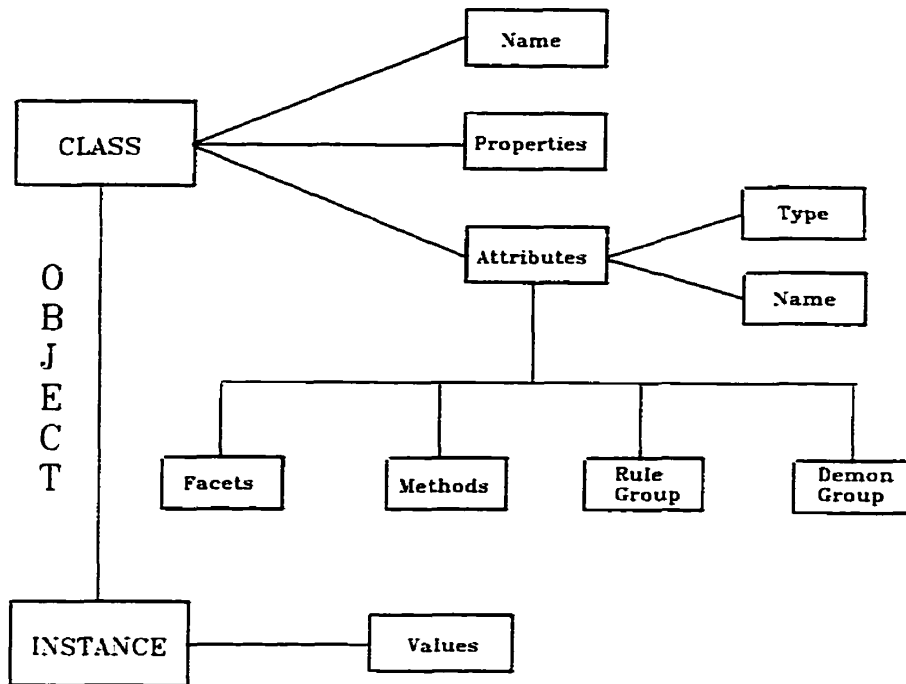


Fig.4.1. Structural breakdown of a Level5 Object

Objects are one of the basic structure components for Level5 knowledge representation. Objects have a class structure to define their characteristics and an instance to hold current data values from the knowledge base. In Level5, there are different kinds of objects, i.e. user-defined and system-defined class. User-defined objects are those you create during development and are specific to each individual application. Usually, at the beginning of development, knowledge engineer will define several classes in **Object Editor** based on the problem to be solved. System objects are a set of predefined objects that Level5 automatically creates in all application such as display, windows etc.. Level5 automatically creates system class when a new knowledge base is created. System class

allows you to control the inferencing and windowing environments, as well as displays, messages, database interaction. Each knowledge base also contains a single, predefined instance of the DOMAIN class created by Level5 which allows you to build applications without explicitly defining your own class. At an initial stage of this expert system development, a small prototype expert system was developed under DOMAIN class.

Within Level5 OBJECT, A class is defined by a collection of attributes, which represent the information contained within the object. Each attribute of a class has a specific attribute type. The Level5 attribute types are compound, multicomponent, simple, numeric, string, picture, rectangular, colour, time and interval. Each attribute can have when-needed and when-changed methods attached. Attributes of objects can be assigned values which can be accessed later in when-needed and when-changed methods. They can also be accessed in rules. Parts of the knowledge base in Level5 can be represented as rules. There are two types of rules: demons (forward-chaining rules) and rules (backward-chaining rules). Rules can use attributes of objects both in promises and conclusions.

4.3 System Architecture

The basic structure of this expert system is shown in Figure 4.2. The knowledge base serves as the overall system integrator handling user interaction, accessing database, passing parameters to external programs and running the plotter programs. Databases (dBASE III format) are used to store bin geometry and material properties. These databases are accessed by the knowledge base

directly. External programs, written in Pascal, are used to present the pressure distribution on the screen. Parameters determined by the knowledge base and those specified by the user are passed to external programs using ASCII files.

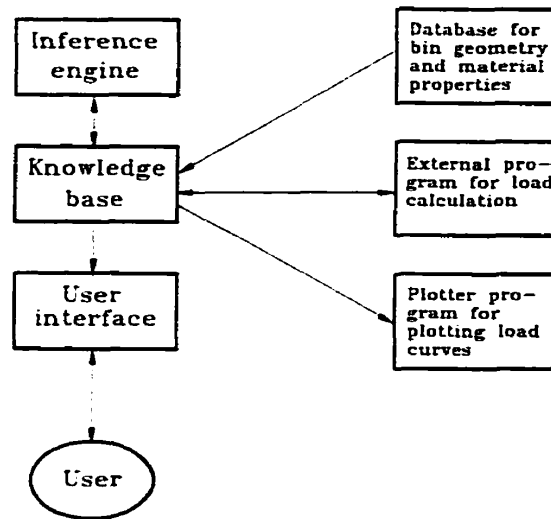


Fig.4.2. Basic structure of this expert system

4.4 Structure of the Bin Loads Analysis Program

The CFBC (NRCC, 1990) and EP433 (ASAE, 1991) recommend that the use of Janssen's equation for calculate the static pressures. When using Janssen's equation to predict material pressures, the predicted value for the material pressures will depend on variables which were specified. Important variables include the bulk density, coefficient of friction on various surface and K value. These variables vary with the type of material stored, the methods of filling and emptying, and conditions of storage. This expert system will help designer to pick up appropriate values. The key part of this ES is to reduce those pressures to the loads on specific component of a grain bin. The general flowchart of this program is shown in Fig.4.3. The flowchart of CFBC and ASAE

subprogram are shown in Fig.4.4. and Fig.4.5.

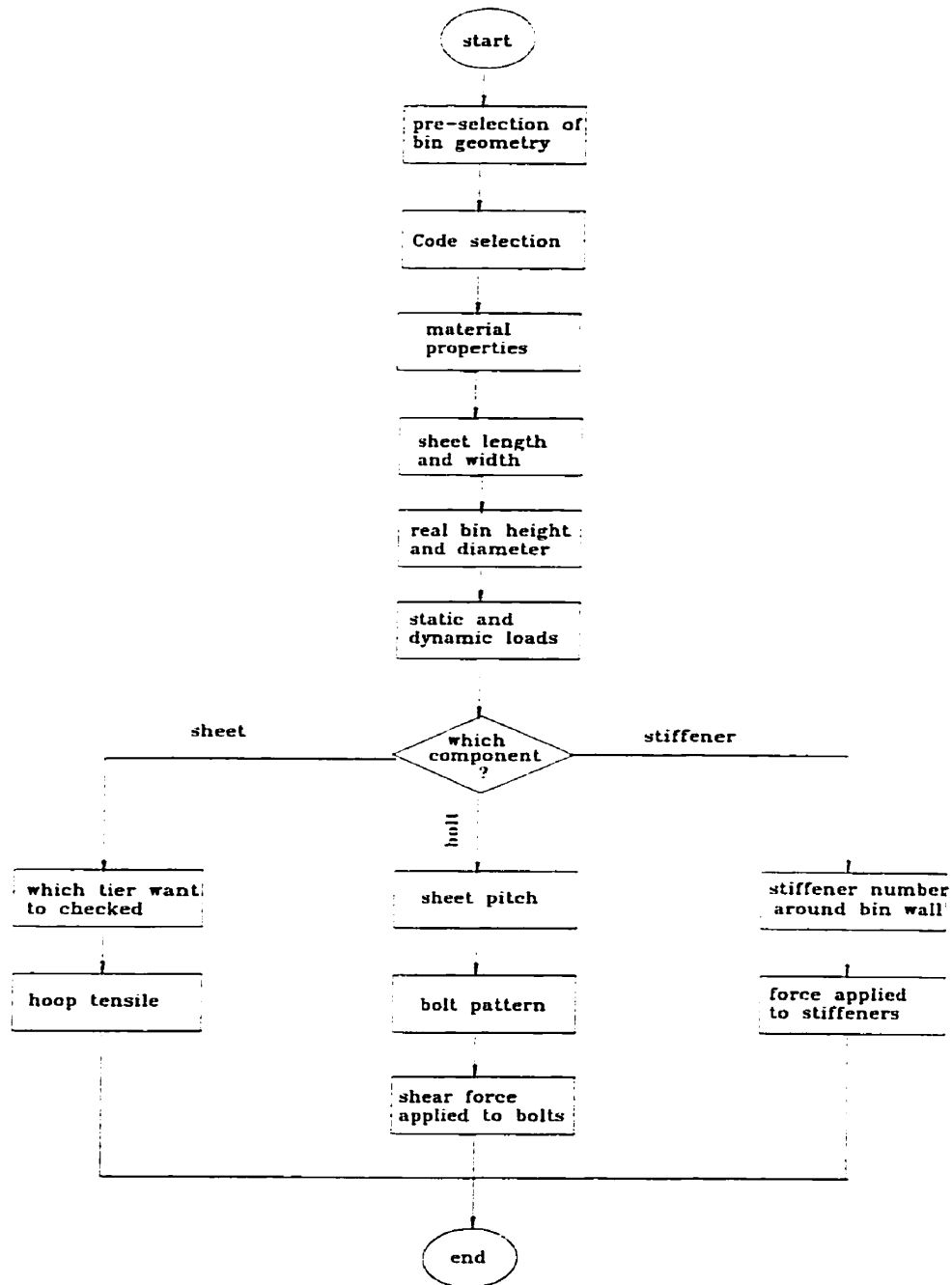


Fig.4.3. General flowchart of this expert system program

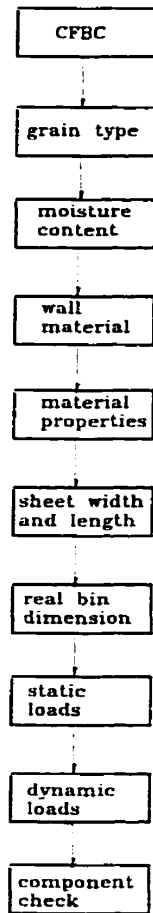


Fig.4.4. Flowchart of CFBC subprogram

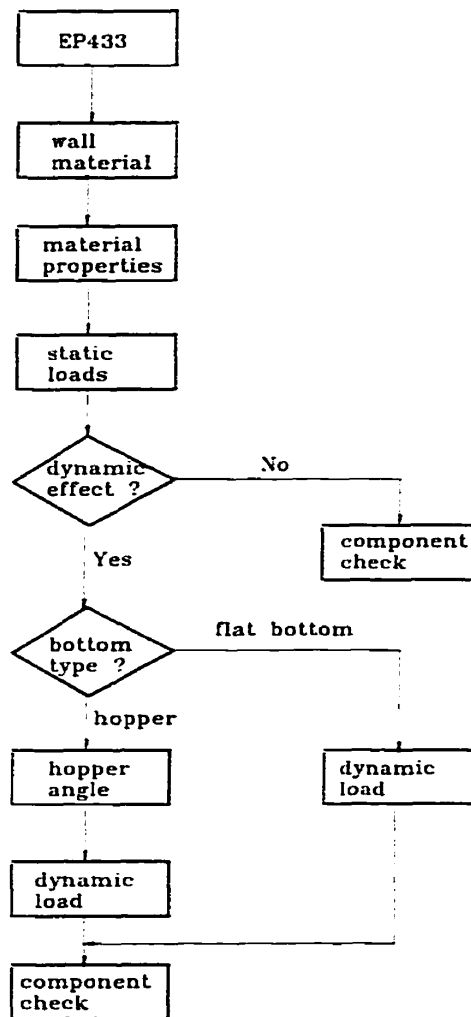


Fig.4.5. Flowchart of ASAE subprogram

4.4.1 Defining class structure

In Object Editor, five classes were created. They are:

- 1) ***Action display of CFBC***, which contains all the "action" to control display and inferencing of CFBC.
- 2) ***Action display of ASAE***, which contains all the "action" to control display and inferencing of ASAE.
- 3) ***Action display of load analysis***, which contains all the common "action" to control display and inferencing during load analysis.
- 4) ***CFBC*** , which consists of all the attributes stand for the parameters to calculate the loads in CFBC.
- 5) ***ASAE*** , which consists of all the attributes stand for the parameters to calculate the loads in ASAE.

4.4.2 Defining attribute structure

The attribute represents the information contained within the object. A knowledge base makes recommendation and conclusion as a result of the values Level5 obtains for the attribute of a class. The attribute of a class can be defined during the development of knowledge base.

The attributes of a class define the qualities of the class and the type of information

associated with the class. The following outlines the basic structure of Level5 attributes and the components.

A name ---- The name must be unique from all other attributes of the class. But different classes may have attributes with the same name.

A type ---- Each attribute of a class has a specific attribute type. The Level5 attribute types are compound, multicomound, simple, numeric, string, picture, rectangular, color, time, and interval.

Its facets ---- Each attribute can have many facets associated with it. Facets provide control over how the inference engines process and use attributes.

Its method ---- Each attribute can also have methods associated with it. Methods establish developer-defined procedures associated with each attribute.

4.4.3 Creating Display

1. Title Display

Display Editor allows developer to display a collection of objects used for prompt form, graphic or reports. Usually every expert system can have a Title Display, which will appear at the

start of a knowledge base session. This is achieved by assigning this display name to the attribute **Title Display** in the **Application System Class**.

In title display, a picturebox presented a grain storage bin and several pushbuttons were built. Picturebox can display a graphic bit-map image in a rectangular area. To read the bit-map data in from a disk resident file, it is necessary to assign the path and filename to the attribute filename. The disk resident file must be in BMP file format. The PRL (production rule language) below declares the instance of the picture system class.

```
INSTANCE Picturebox 1 ISA Picturebox  
WITH location := 0, 0, 500, 450.  
WITH clipped := TRUE  
WITH filename := "C:\L5o25\N\Title.BMP"
```

* Title.BMP is implement in the paintbrush of window 3.1.

Three pushbuttons (run, exit, and about) were created for different cases. During a knowledge base session, selecting a pushbutton changes the value of a simple attribute attached to this pushbutton in the context. Selecting a pushbutton can also cause a display to be sent to the current window or initiate an action. The PRL below declares the instance of the pushbutton **Run**.

```
INSTANCE Pushbutton  
WITH location := 391,339,461,364
```

WITH label := "Run"

WITH display attachment := Grain Amount Display

When the user selects a pushbutton, the value of the reference attribute changes. Assigning a display instance name (grain amount display) to the display attachment causes Level5 activated that display when the end user selects this pushbutton.

2. Grain Amount Display

To establish the bin geometry, two parameters must be determined. One is the grain amount to be stored in the bin and the other is the bulk density of stored grain. CFBC specifies a different grain bulk density due to variety of stored grain. A database (shown in appendix A) contained the relationship between the bin geometry (bin diameter and height) and the grain amount was established in Paradox for Window based on the lowest grain density - the barley which will lead to a conservation design. In this display, the user will be asked to enter the grain amount (Tons) in promptbox, which prompts the end user for an attribute's value. The end user can modify any current value displayed in the promptbox or enter a new value. The PRL below declares an instance of the promptbox system class.

INSTANCE promptbox ISA promptbox

WITH location := 410, 212, 467, 248

WITH justify IS left

WITH frame := TRUE

WITH show current := TRUE

WITH attachment := Grain Amount

A pushbutton **Continue** attached to attribute **Continues Weight display OF Action display of CFBC** was created. This attribute references a when-changed method to find qualified geometry bins through the database. Level5 Object can create a class corresponding to a given dBase file automatically. The class will inherit all attributes of the dB3 system class. The Database automatically generates the class structure of Level5 database objects from the external database structure. The PRL below declares the when-changed method to find qualified bins from a database file.

WITH Continues Weight display SIMPLE

WHEN CHANGED

BEGIN

FIND dB3 WEIGHT 1

WHERE weight of dB3 WEIGHT1 >= amount of grain

AND weight of dB3 WEIGHT1 <= amount of grain + 30

WHEN FOUND

MAKE qualified bin

WITH bin diameter := bin_diameter OF dB3 WEIGHT1

WITH bin height := bin_height OF dB3 WEIGHT1

WITH weight := weight of dB3 WEIGHT1

FIND END

visible OF main window := FALSE

output OF Main Window := qualified bin display

visible OF main window := TRUE

END

WHEN CHANGED is a method containing a sequence of procedural statements that Level5 executes when an attribute's value changes. A **WHEN CHANGED** block can include one or more of the statements listed below. These statements can occur in any order and can be nested within a single block. Level5 evaluates them from top to bottom. Whenever the user select a pushbutton, the when-changed method attached to this pushbutton will fire. **FIND command** compares the instance values of one or more classes to find those instances that meet conditions specified in a **WHERE clause**. When values are found satisfying the conditions specified in the **WHERE clause**, the specified **WHEN FOUND clause** is exerted. The **MAKE command** creates a new instance of a class (qualified bin) during the session, and **WITH clause** assign values which were found in a database to the attribute of the instance. The statement sends the **qualified bin display** to the main window and make the window visible.

3. Qualified Bin Display

A table was created to show all the eligible bins which are satisfied grain amounts. **Table system class** allows you to display the instance values of a class as data within a table. The PRL below declares an instance of this table system class

Class table

WITH attachment qualified bin

WITH columns REFERENCE COLLECTION

WITH heading SIMPLE

INIT TRUE

WITH heading height NUMERIC

WITH row height NUMERIC

WITH fill colour COLOUR

WITH column lines SIMPLE

INIT TRUE

WITH row lines SIMPLE

INIT TRUE

WITH frame SIMPLE

INIT TRUE

WITH selected SIMPLE

WITH double clicked SIMPLE

The attachment attribute references the qualified bin class whose instance values will appear in the table. One row in the table represents one instance of the attached class.

The simple attribute **selected** has a value of **TRUE** at run time when you select a row in a table. This attribute also sets the instance in the selected row as the current instance of its class. This tells Level5 Object when and what instance was selected in a table. If an end user forgets to select one item from the table, knowledge base will prompt the user to select one.

A pushbutton **Other** was created for the purpose when users want to enter other bin geometry other than one of the tables.

4. Code Selection Display

In this display, a radiobutton group was attached to a compound attribute **Code Name** . Compound attribute represents an attribute that assumes a single values from a logically related group of symbolic values. The **CFBC, ASAE, ACI, DIN 1055** are designated as the values of **Code Name**. ACI and DIN 1055 are included for the purpose of further development. Radiobutton group displays a group of radiobuttons which represent the possible values of a compound attribute. Radiobuttons appear in groups, and are used to respond to questions where the choices are mutually exclusive. The PRL system of the radiobutton system class is:

```
INSTANCE radiobutton ISA radiobutton group  
WITH location := 179, 115, 363, 262  
WITH pen colour := 0, 0, 255  
WITH fill colour := 192, 192, 192  
WITH frame := FALSE  
WITH label := "Code Name"  
WITH show current := TRUE  
WITH attachment := Code Name
```

A pushbutton attached to the when-changed method **Continues code display** was created in this display. If the user selects CFBC, this when-changed method fires, and **Grain Type Display** will appear. If the user selects ASAE, this when-changed method also fires, and **Wall Material of ASAE Display** will appear. The PRL below declares this method.

```
WITH Continues code display SIMPLE  
WHEN CHANGED
```

```

BEGIN
IF Code name IS CFBC 1990 THEN
BEGIN
visible OF main window := FALSE
output OF subwindow1 := Grain type display of CFBC
visible OF subwindow1 := TRUE
END
IF Code name IS EP433 1991 THEN
BEGIN
visible OF main window := FALSE
output OF subwindow1 := wall material display of ASAE
visible OF subwindow1 := TRUE
END
END

```

5. a. Grain Type OF CFBC Display

In CFBC, the value of bulk density, coefficient of friction and K value will depend on grain type, moisture content and bin wall material. Grain type includes Wheat, Barley, Shelled Corn, Soybeans, Flaxseed, and Canola. A table was created for displaying grain type. Different grain will hold different moisture content. Therefore, knowledge base should present suitable moisture content corresponding to the grain type selected by the user. This is accomplished by following when-changed method.

WITH Continues grain type selection SIMPLE

```

WHEN CHANGED

BEGIN

IF select OF table grain type = TRUE THEN

BEGIN

load MC OF Action display of CFBC := TRUE

visible OF subwindow1 := FALSE

output OF subwindow1 := Moisture Content of CFBC display

visible OF subwindow1 := TRUE

END

ELSE

BEGIN

text OF validationMessage := "Please select a grain from the table before selecting Continue pushbutton"

AskValidationMessage

END

END

```

When the user chooses the **Continue** pushbutton which attached to above when-changed method after selecting a grain, the when-changed method **Load MC OF Action display of CFBC** will fire.

The PRL below declares this when-changed method.

```

WITH Load MC SIMPLE

WHEN CHANGED

BEGIN

FORGET M_C list

FIND M_C list

WHERE grain name OF M_C = grain name OF grain

```

```

WHEN FOUND
MAKE M_C list
WITH M_C := MC OF M_C
FIND END
END

```

FORGET command removes values of attribute and sets its confidence to undetermined. **Where** clause matches the grain type defined by user with that of the **M_C Class**. Then **MAKE command** lists the moisture content in **M_C list** class. The instance of **M_C** class was created at editing time (shown in Table 4.2).

Table 4.2. Moisture content of CFBC

Grain name	Moisture Content
Wheat	11
Wheat	13
Barley	11
Barley	13
Shelled Corn	11
Shelled Corn	16
Soybeans	11
Flaxseed	9
Flaxseed	11.5
Canola	9
Canola	12.5

5. b. Wall Material of ASAE Display

In EP433, a maximum of 834 kg/m³ is recommended for the bulk density of any free-flowing grain, and other material properties only depend on wall material (shown in table 4.3.).

Table 4.3. Material properties for EP433

Wall Material	μ	K
Smooth Steel	0.30	0.5
Concrete	0.40	0.5
Corrugate Steel	0.37	0.5

A radiobutton group is attached to a compound attribute **Wall Material** , which consists of Steel, Concrete, and Corrugate Steel. A pushbutton **Continue** attached to following when-changed method was created in this display to find material properties (μ , K).

```
WITH Continue WM of ASAE SIMPLE
BEGIN
PURSUE K OF qualified grain
visible OF main window := FALSE
output OF main window := Grain Property Display
visible OF main widow := TRUE
END
```

PURSUE command allows you to invoke a backward-chaining inference engine from any point within an application. This command tells Level5 Object to follow the search order for a specified attribute (**K OF qualified grain**) until a value is obtained for the attribute. The material properties (μ , K) were determined by following when-needed method.

```

ATTRIBUTE K OF qualified grain NUMERIC
WHEN NEEDED
BEGIN
IF Wall Material IS Smooth Steel THEN
BEGIN
Density OF Qualified Grain := 834
Coefficient of friction OF Qualified Grain := 0.30
K OF Qualified Grain := 0.5
END
IF Wall Material IS Concrete THEN
BEGIN
Density OF Qualified Grain := 834
Coefficient of friction OF Qualified Grain := 0.40
K OF Qualified Grain := 0.5
END
IF Wall Material IS Corrugate Steel THEN
BEGIN
Density OF Qualified Grain := 834
Coefficient of friction OF Qualified Grain := 0.37
K OF Qualified Grain := 0.5
END
END

```

6. Moisture Content of CFBC Display

A table attached to **M_C** list class was created to display the moisture content corresponding to selected grain. The user can pick up one moisture content based on his/her own experience, and click **Continue** pushbutton to go over **Wall Material OF CFBC Display**.

7. Wall Material of CFBC Display

Wall Material is a compound attribute, which consists of **Smooth Steel, Corrugate Steel, Plywood, and Concrete**. A Radiobutton attached to **Wall Material** attribute was created. The user

can select one of wall materials. A when-changed method is attached to **Continue** pushbutton to determine material properties such as bulk density, coefficient of friction, and K value. The PRL below declares this when-changed method.

```
WITH Continue WM OF CFBC selection SIMPLE  
BEGIN  
FIND dB3 Property1  
WHERE grain_type OF Property1 = grain name OF grain  
AND mois_cont OF Property1 = MC OF M_C list  
AND wall_mate OF Property1 = material name OF Wall Material  
WHEN FIND  
MAKE Qualified Grain  
WITH Density := density OF Property1  
WITH Coefficient of friction := coef_fric OF Property1  
With K := K OF Property1  
FIND END  
visible OF subwindow1 := FALSE  
output OF main window := Grain Property Display  
visible OF main window := TRUE  
END
```

In this when-changed method, knowledge base will match the grain type, moisture content, and wall material specified by user with dB3 Property1 which stored the material properties (shown in Appendix B). **MAKE command** will create the instance of **Qualified Grain** class.

8. Grain Property Display

The material properties (w , μ , K) will be presented to the user in promptboxes, which allow the user to modify any current value or enter a new value. In the mean time, help screen will provide the user with some information about the influence some factors on material properties by selecting **Help** pushbutton. Many research work has done in this area. In help screen, the user can get summary information about how to pick up a suitable value. Also, the user can optionally select

Change pushbutton if he/she want to change material properties according to his/her own experience.

9. Sheet Dimension Display

As mentioned before, this expert system will focus on the corrugated steel bins which stand for most of on-farm grain storage bins. Sheet length and sheet width were asked to recalculate suggested bin dimensions on the basis of "full" sheets. Storage capacity will be recalculated for each set of suggested dimensions. The user can enter this two dimensions or pick it up from the table which stand for several sheets used in WESTEEL. The PRL below declares when-changed method for recalculating bin dimension and storage capability.

```
WITH Continues sheet display SIMPLE
WHEN CHANGED
BEGIN
IF bin height OF qualified bin / sheet width = INT (bin height OF qualified bin / sheet width) THEN
Tier Number := bin height OF qualified bin / sheet width
ELSE
Tier Number := INT (bin height OF qualified bin / sheet width + 1 )
BEGIN
FORGET Table1
i := 0
WHERE ( i < Tier Number )
BEGIN
i := i + 1
MAKE Table1
WITH Tire Number := i
END
Bin Height := Tier Number × sheet width
Bin Diameter := INT ( bin diameter OF qualified bin × π / sheet width + 1 ) × sheet length / π
Grain Amount := ( π × SQR (Bin Diameter/2) × Bin Height × density OF qualified grain × 1.06 / 1000
Grain Amount := ROUND (Grain Amount)
output OF subwindow2 := Real bin display
```



```

visible OF subwindow2 := TRUE
END
END

```

MAKE command creates the instance of **Table1** class. A **Display** pushbutton attached to above when-changed method was created to display real bin dimension and storage capability. A **Close** pushbutton lets user return to main window from subwindow2. A **Continue** pushbutton attached to **continues sheet dimension display** was created to continue knowledge session. This when-changed method will lead to the load data and curve display.

10. Static Load and Curve Display

Until now, knowledge base has obtained all the information for determining static loads. Therefore, the user can select to see load data or view load curves.

If the user selects **see load data Yes**, following when-changed method **Continues load data yes display** will exert.

```

WITH Continues load data yes display SIMPLE
WHEN CHANGED
BEGIN
IF Code name IS CFBC 1990 THEN
BEGIN
PURSUE Sta_Hori_Pressure [1] OF CFBC
FORGET Table2
i := 1
WHILE ( i < Tier Number + 1 )
BEGIN
i := i + 1
MAKE Table2
WITH A := Height [i] OF CFBC
WITH B := Sta_Hori_Pressure [i] OF CFBC / 1000
WITH C := Vert_Pressure [i] OF CFBC / 1000
WITH D := Vert_Friction [i] OF CFBC / 1000

```

```

END
output OF subwindow3 := Static Load display of CFBC
visible OF subwindow3 := TRUE
END

IF Code name IS EP433 1991 THEN
BEGIN
PURSUE Sta_Hori_Pressure [1] OF ASAE
FORGET Table3
i := 1
WHILE ( i < Tier Number + 1 )
BEGIN
i := i + 1
MAKE Table3
WITH A := Height [i] OF ASAE
WITH B := Sta_Hori_Pressure [i] OF ASAE / 1000
WITH C := Vert_Pressure [i] OF ASAE / 1000
WITH D := Shear_Stresses [i] OF ASAE / 1000
WITH E := Vert_Wall_Load [i] OF ASAE / 1000
END
output OF subwindow3 := Static Load display of ASAE
visible OF subwindow3 := TRUE
END
END

```

Sta_Hori_Pressure OF CFBC is an array type attribute, which can be declared in **Attribute Type Dialog**. The value of array elements can be assigned at run time from methods, rules, or demons. The attribute **Sta_Hori_Pressure OF CFBC** and **Sta_Hori_Pressure OF ASAE** respectively reference a **when_needed** method which specifies a procedure that Level5 uses when determining an attribute value. When you declare a when-needed method for an attribute, Level5 Object automatically replaces the default search order list (C, W, R, Q, D) with a search order list containing just when-needed. The PRL below declares the when-needed method to determined **Sta_Hori_Pressure OF CFBC**.

```

ATTRIBUTE Sta_Hori_Pressure OF CFBC NUMERIC
ARRAY SIZE 100
WHEN NEEDED
BEGIN
h := 0
i := 1
WHILE (h <= Tier Number × Sheet Width)
BEGIN
Sta_Hori_Pressure [i] OF CFBC := 1.06 × 9.81 × Density OF qualified grain × R / Coefficient of friction OF
qualified grain × ( 1 - EXP (-K OF qualified grain × Coefficient OF
qualified grain × h / R ))
Vert_Pressure [i] OF CFBC := Sta_Hori_Pressure [i] OF CFBC × K OF qualified grain
Vert_Friction [i] OF CFBC := 1.06 × 9.81 × Density OF qualified grain × R × ( h - R / (K OF qualified grain
× Coefficient of friction OF qualified grain) + R / (K OF qualified grain ×
Coefficient of friction OF qualified grain × EXP (-K OF qualified grain ×
Coefficient OF qualified grain × h / R )))
Height[i] OF CFBC := i - 1
h := h + sheet width
i := i + 1
END
END

```

The static loads of ASAE are determined by another when-needed method similar to the above when-needed method to determine static loads of CFBC.

When finishing to pursue **Sta_Hori_Pressure**, **MAKE command** creates the instance of **Table2** class and sends the values of **Height**, **Sta_Hori_Pressure**, **Vert_Pressure**, and **Vert_Friction** to Numeric attribute A, B, C, and D. The statement sends static load display to subwindow3 and make the window visible. A table attached to **table2** class was created to display static loads in subwindow3. A pushbutton **Close** is attached to a when-changed method to close this subwindow.

If the user selects **view load curve Yes**, when-changed method **Display Curves** exerts. The PRL below declares this method.

```

WITH Display Curves SIMPLE
WHEN CHANGED
BEGIN
IF Code Name IS CFBC 1990 THEN
BEGIN
filename OF file 1 := "C:\15o25\mi\height.txt"
action OF file 1 IS open new := TRUE
write line OF file 1 := TO STRING(Tier Number)
action OF file 1 IS close := TRUE
filename OF file 2 := "C:\15o25\mi\Cstaload.txt"
action OF file 2 IS open new := TRUE
FOR ( i :=1 TO Tier Number + 1)
BEGIN
write line OF file 2 := TO STRING(Sta_Hori_Pressure[i] OF CFBC)
write line OF file 2 := TO STRING(Vert_Pressure[i] OF CFBC)
write line OF file 2 := TO STRING(Vert_Friction[i] OF CFBC)
END
action OF file 2 IS close := TRUE
ESTABLISH "IPU, EXTERN, C:\BP\NI\Cstaload.exe"
END
IF Code Name IS EP433 1991 THEN
BEGIN
filename OF file 1 := "C:\15o25\mi\height.txt"
action OF file 1 IS open new := TRUE
write line OF file 1 := TO STRING(Tier Number)
action OF file 1 IS close := TRUE
filename OF file 2 := "C:\15o25\mi\Astaload.txt"
action OF file 2 IS open new := TRUE
FOR ( i:=1 TO Tier Number + 1)
BEGIN
write line OF file 2 := TO STRING(Sta_Hori_Pressure[i] OF ASAE)
write line OF file 2 := TO STRING(Vert_Pressure[i] OF ASAE)
write line OF file 2 := TO STRING(Shear_Stress[i] OF ASAE)
write line OF file 2 := TO STRING(Vert_Wall Load [i] OF ASAE)

```

```

END
action OF file 2 IS close := TRUE
ESTABLISH "IPU, EXTERN, C:\BP\NT\Astsload.exe"
END
END

```

This method is used for calling an external program which is used to display load curves. Parameters to the external program are passed through ASCII files. Level5 Object creates an ASCII file by setting the action OF a file attribute to open new, writes the parameters to the file by **To STRING** function which converts a numeric attribute type to a string, and closes the file by setting the action OF a file attribute to close. Next, Level5 Object calls the external program (shown in Appendix C), written in Pascal, using **ESTABLISH command**. The external program will display static load curves on the screen.

A pushbutton **Continue** is attached to following when-changed method for examining dynamic effect.

```

WITH Continues static load display SIMPLE
WHEN CHANGED
BEGIN
IF Code Name IS CFBC 1990 THEN
BEGIN
visible OF main window := FALSE
output OF main window := dynamic effect prompt OF CFBC display
visible OF main window := TRUE
END
IF Code Name IS EP433 1991
AND Bin Height / Bin Diameter >= 2 THEN
BEGIN
visible OF main window := FALSE
output OF main window := dynamic effect prompt OF ASAE display
visible OF main window := TRUE
END

```

```

IF Code Name IS EP433 1991
AND Bin Height / Bin Diameter < 2 THEN
BEGIN
visible OF main window := FALSE
output OF main window := component check display
visible OF main window := TRUE
END
END

```

11. a. Dynamic Effect Prompt of CFBC Display

According to CFBC, it is necessary to check dynamic horizontal pressure. In this display, a textbox was created to provide the user with some information about dynamic effect during discharge. A pushbutton **Continue** leads to the user to examine dynamic loads.

11. b. Dynamic Effect Prompt of ASAE Display

According to EP433, the dynamic effect must be considered in mass flow bins ($H/D \geq 2$). The overpressure factor differs by bin bottom type, ie. flat bottom bin and hopper bin. A pushbutton **Continue** is attached to bottom type display OF ASAE.

12. a. Dynamic Load and Curve Display

Like static loads, the user can select **see dynamic load data** or **view load curves**. If user selecting **see load data Yes**, the when-changed method **Continues Dy Load Data Yes Display** fires. The PRL below declares this method.

```

WITH Continues Dy Load Yes Display SIMPLE
WHEN CHANGED
BEGIN
IF Code Name IS CFBC 1990 THEN

```

```

BEGIN
PURSUE Dyn_Hori_Pressure [1] OF CFBC
FORGET Table4
i:= 1
WHILE ( i < Tier Number + 1 )
BEGIN
i := i + 1
MAKE Table4
WITH A:= Height[i] OF CFBC
WITH B := Sta_Hori_Pressure [i] OF CFBC / 1000
WITH C := Dyn_Hori_Pressure [i] OF CFBC / 1000
END
output OF subwindow3 := Dynamic Load Display
visible OF subwindow3 := TRUE
END

```

```

IF Code Name IS EP 433 1991 AND Bin Type OF ASAE IS Flat Bottom Bin THEN
BEGIN
PURSUE Dyn_Hori_Pressure flat [1] OF ASAE
FORGET Table4
i:= 1
WHILE ( i < Tier Number + 1 )
BEGIN
i := i + 1
MAKE Table4
WITH A:= Height[i] OF ASAE
WITH B := Sta_Hori_Pressure [i] OF ASAE / 1000
WITH C := Dyn_Hori_Pressure flat [i] OF CFBC / 1000
END
output OF subwindow3 := Dynamic Load Display
visible OF subwindow3 := TRUE
END

```

```

IF Code Name IS EP433 1991 AND Bin Type OF ASAE IS Hopper Bin THEN
BEGIN
PURSUE Dyn_Hori_Pressure hopper [1] OF ASAE

```

```

FORGET Table4
i:= 1
WHILE ( i < Tier Number + 1 )
BEGIN
i := i + 1
MAKE Table4
WITH A:= Height[i] OF ASAE
WITH B := Sta_Hori_Pressure [i] OF ASAE / 1000
WITH C := Dyn_Hori_Pressure hopper [i] OF CFBC / 1000
END
output OF subwindow3 := Dynamic Load Display
visible OF subwindow3 := TRUE
END
END

```

The dynamic horizontal pressure in CFBC is determined by following when-needed method:

```

ATTRIBUTE Dyn_Hori_Pressure OF CFBC NUMERIC
ARRAY SIZE 100
WHEN NEEDED
BEGIN
i := 1
WHILE ( i <= Tier Number + 1 )
BEGIN
Dyn_Hori_Pressure [i] OF CFBC := Overpressure_Factor × Sta_Hori_Pressure [i] OF CFBC
Height [i] OF CFBC := i - 1
i := i + 1
END
END

```

The Overpressure Factor in CFBC is shown in Table 4.4 .

Table 4.4. Overpressure Factors for CFBC

Overpressure Factor for Stored Grain		
Grain stored	Overpressure factor	
	$H/4R \leq 2.5 \cdot u$	$H/4R > 5 \cdot u$
Cereal grains, shelled corn, soybeans and canola	1.0	1.4
Flaxseed and canola	1.0	1.6

Overpressure factor is determined by following Rule Group

Rule 1.

**IF Bin Height / Bin Diameter $\leq 2.5 \times$ Coefficient of Friction OF qualified grain
THEN Overpressure_factor := 1**

Rule 2.

**IF Bin Height / Bin Diameter $\geq 5 \times$ Coefficient of Friction OF qualified grain
AND grain name OF grain = "Flaxseed"
THEN Overpressure_factor := 1.6**

Rule 3.

**IF Bin Height / Bin Diameter $\geq 5 \times$ Coefficient of Friction OF qualified grain
AND grain name OF grain \neq "Flaxseed"
THEN Overpressure_factor := 1.4**

Rule 4.

**IF Bin Height / Bin Diameter $> 2.5 \times$ Coefficient of Friction OF qualified grain
IF Bin Height / Bin Diameter $< 5 \times$ Coefficient of Friction OF qualified grain
AND grain name OF grain \neq "Flaxseed"**


```

ARRAY SIZE 100
WHEN NEEDED
BEGIN
  i := 1
  WHILE ( i <= Tier Number + 1 )
    BEGIN
      Dyn_Hori_Pressure hopper [i] OF ASAE := 1.4 × Sta_Hori_Pressure [i] OF ASAE
      i := i + 1
    END
  END

```

IF the user selects **view load curve Yes**, the following when-changed method fires. The PRL below declares this method.

```

WITH Continues dynamic load curve display SIMPLE
WHEN CHANGE
BEGIN
  IF Code Name IS CFBC 1990 THEN
    BEGIN
      filename OF file 3 := "C:\15o25\mf\height.txt"
      action OF file 3 IS open new := TRUE
      write line OF file 3 := TO STRING (Tier Number)
      action OF file 3 IS close := TRUE
      filename OF file 4 := "C:\15o25\mf\dyload.txt"
      action OF file 4 IS open new := TRUE
      FOR (i:= 1 TO Tier Number + 1)
        BEGIN
          write line OF file 4 := TO STRING (Sta_Hori_Pressure[i] OF CFBC)
          write line OF file 4 := TO STRING (Dy_Hori_Pressure[i] OF CFBC)
        END
      action OF file 3 IS close := TRUE
      ESTABLISH "IPU, EXTERN, C:\BP\mf\Cdyload.exe"
    END

IF Code Name IS EP433 1991 AND Bin Type IS Hopper Bin THEN
  BEGIN

```

```

filename OF file 3 := "C:\15o25\n\height.txt"
action OF file 3 IS open new := TRUE
write line OF file 3 := TO STRING (Tier Number)
action OF file 3 IS close := TRUE
filename OF file 4 := "C:\15o25\n\dyload.txt"
action OF file 4 IS open new := TRUE
FOR (i:= 1 TO Tier Number + 1)
BEGIN
write line OF file 4 := TO STRING (Sta_Hori_Pressure hopper [i] OF ASAE)
write line OF file 4 := TO STRING (Dy_Hori_Pressure hopper [i] OF ASAE)
END
action OF file 3 IS close := TRUE
ESTABLISH "IPU, EXTERN, C:\BP\n\Cdyload.exe"
END

```

```

IF Code Name IS EP433 1991 AND Bin Type IS Flat Bottom Bin THEN
BEGIN
filename OF file 3 := "C:\15o25\n\height.txt"
action OF file 3 IS open new := TRUE
write line OF file 3 := TO STRING (Tier Number)
action OF file 3 IS close := TRUE
filename OF file 4 := "C:\15o25\n\dyload.txt"
action OF file 4 IS open new := TRUE
FOR (i:= 1 TO Tier Number + 1)
BEGIN
write line OF file 4 := TO STRING (Sta_Hori_Pressure flat [i] OF ASAE)
write line OF file 4 := TO STRING (Dy_Hori_Pressure flat [i] OF ASAE)
END
action OF file 3 IS close := TRUE
ESTABLISH "IPU, EXTERN, C:\BP\n\Cdyload.exe"
END
END

```

TO STRING function sends the value of other numeric types to a file by converting the value to string. When finishing bin load consultation, knowledge base will prompt end user to continue the

load analysis session.

12. b. Bottom Type of ASAE Display

A radiobox attached to the compound attribute **Bin Type** was created in this display. The **Bin Type** consists of **Hopper Bin** and **Flat Bottom Bin**. A pushbutton **Continue** was attached to following when-changed method **Continues bin type display**.

```
WITH Continues bin type display SIMPLE
BEGIN
IF Bin Type OF ASAE IS Flat Bottom Bin THEN
BEGIN
visible OF main window := FALSE
output OF main window := Dynamic load and curve display
visible OF main window := TRUE
END
IF Bin Type OF ASAE IS Hopper Bin THEN
BEGIN
visible OF main window := FALSE
output OF main window := Hopper angle display
visible OF main window := TRUE
END
END
```

If the user selects flat bottom bin to be examined, the **Dynamic load and curve display** will appear. Like dynamic load and curve display OF CFBC, user can see dynamic load data or view load curves.

12.b.1 Hopper Angle Display

If the user selects a hopper bin to be examined, the hopper angle display will appear. A promptbox attached to hopper angle was created in this display. A picturebox will display a hopper

to show the user which angle is needed. A pushbutton **Continue** is attached to **Dy_Hori_Pressure** hopper **OF ASAE** display.

12.b.2 Dy_Hori_Pressure hopper OF ASAE Display

Four pushbuttons were created in this display for seeing dynamic load data and curves on bin body and hopper respectively. The dynamic load applied on bin body is similar to that of CFBC. The PRL below declares a when-changed method to display dynamic loads applied to hopper.

```
WITH Continues dynamic data at hopper display SIMPLE  
WHEN CHANGED  
BEGIN  
PURSUE Normal Pressure [1] OF ASAE  
FORGET Table5  
i := 0  
WHILE ( i < Count OF ASAE )  
BEGIN  
i := i + 1  
MAKE Table5  
WITH A := Height [i] OF ASAE  
WITH B := Normal Pressure [i] OF ASAE / 1000  
WITH C := Tangential Stress [i] OF ASAE / 1000  
END  
END
```

The Normal Pressure is determined by following when-needed method.

```
WITH Normal Pressure NUMERIC  
ARRAY SIZE 100  
WHEN NEEDED  
BEGIN  
h := 0  
i := 1
```

```

WHILE ( h <= Hopper height OF ASAE )
BEGIN
Normal Pressure [i] OF ASAE := ( 1 + (Hopper height OF ASAE - h ) / Hopper height OF ASAE × 0.4)
× ( 1.08 × 9.81 × 834 × R / Coefficient of friction OF ASAE × (1 - EXP
(- K OF ASAE × Coefficient of friction OF ASAE × ( h + Bin Height ) / R
)) × SQR ( COS ( Hopper angle OF ASAE × π / 180 )) / K OF ASAE + SQR
( SIN (Hopper angle OF ASAE × π / 180 )))
Tangential Stress [i] OF ASAE := Coefficient of friction OF ASAE × Normal Pressure [i] OF ASAE
Height [i] OF ASAE := h
h := h + 0.2
i := i + 1
END
Count OF ASAE := i + 1
END

```

The Hopper height is determined by following when-needed method.

```

WITH Hopper height NUMERIC
WHEN NEEDED
BEGIN
Hopper height OF ASAE := Bin Diameter / 2 × TAN (Hopper angle OF ASAE × π / 180)
END

```

The method to display load curve on hopper is same as the curve display mentioned before.

13. Component Check Display

A radiobox attached to the compound attribute **component** was created in this display. The component consists of **sheet, bolt, and stiffener**. The user can select one for examination. A pushbutton **Continue** is attached to following when-changed method **Continues component check display**. The PRL below declares this method.

```

WITH Continues component check display SIMPLE

```

```

WHEN CHANGED
BEGIN
IF Component IS Sheet THEN
BEGIN
visible OF main window := FALSE
output OF main window := Sheet analysis display
visible OF main window := TRUE
END
IF Component IS Bolt THEN
BEGIN
visible OF main window := FALSE
output OF main window := Bolt analysis display
visible OF main window := TRUE
END
IF Component IS Stiffener THEN
BEGIN
visible OF main window := FALSE
output OF main window := Stiffener analysis display
visible OF main window := TRUE
END
END

```

Based on the user selection, knowledge base will send different display to the user.

If sheet is selected to be examined, the sheet analysis display will appear.

14. Sheet Analysis Display

A table attached to **Tier Number**, which was determined in sheet dimension display, was created. Tier number is defined as the first tier from bin top. The user can select a specific tier for examining. A valuebox by the table will display the depth from the top of grain to the top of selected tier (**Depth from grain top**). The pushbutton **Continue** is attached to when-changed method **Continues sheet analysis display**. The PRL below declares this method:

WITH Continues Sheet Analysis display SIMPLE
WHEN CHANGE
PURSUE Hoop tensile at sheet bottom
output OF subwindow4 := Hoop Tensile Display
visible OF subwindow4 := TRUE

The value of hoop tensile at sheet bottom is determined by following when-need method.

Attribute Hoop tensile at sheet bottom NUMERIC

WHEN NEEDED

BEGIN

IF Code Name IS CFBC 1990 THEN

BEGIN

Hoop tensile at sheet bottom := Dyn_Hori_Pressure at sheet bottom of CFBC × Bin Diameter / 2

Hoop tensile at sheet middle := Dyn_Hori_Pressure at sheet middle of CFBC × Bin Diameter / 2

Hoop tensile at sheet top := Dyn_Hori_Pressure at sheet top of CFBC × Bin Diameter / 2

END

IF Code Name IS EP433 1991 AND Bin Height / Bin Diameter \geq 2 THEN

BEGIN

Hoop tensile at sheet bottom := Dyn_Hori_Pressure at sheet bottom of ASAE × Bin Diameter / 2

Hoop tensile at sheet middle := Dyn_Hori_Pressure at sheet middle of ASAE × Bin Diameter / 2

Hoop tensile at sheet top := Dyn_Hori_Pressure at sheet top of ASAE × Bin Diameter / 2

END

IF Code Name IS EP433 1991 AND Bin Height / Bin Diameter < 2

BEGIN

Hoop tensile at sheet bottom := Sta_Hori_Pressure at sheet bottom of ASAE × Bin Diameter / 2

Hoop tensile at sheet middle := Sta_Hori_Pressure at sheet middle of ASAE × Bin Diameter / 2

Hoop tensile at sheet top := Sta_Hori_Pressure at sheet top of ASAE × Bin Diameter / 2

END

END

Attribute Dyn_hori_pressure at sheet bottom NUMERIC

WHEN NEEDED

BEGIN

Dyn_Hori_Pressure at sheet bottom of CFBC := Overpressure_factor × 1.06 × 9.81 × Density OF qualified grain × R / Coefficient of friction OF qualified grain × (1 - EXP (-K OF qualified grain × Coefficient OF qualified grain × Depth from grain top / R))/1000

Dyn_Hori_Pressure at sheet middle of CFBC := Overpressure_factor × 1.06 × 9.81 × Density OF qualified grain × R / Coefficient of friction OF qualified grain × (1 - EXP (-K OF qualified grain × Coefficient OF qualified grain × (Depth from grain top - sheet width/2) / R))/1000

Dyn_Hori_Pressure at sheet top of CFBC := Overpressure_factor × 1.06 × 9.81 × Density OF qualified grain × R / Coefficient of friction OF qualified grain × (1 - EXP (-K OF qualified grain × Coefficient OF qualified grain × (Depth from grain top - sheet width) / R))/1000

END

Three valueboxes were created to display hoop tensile at sheet bottom, middle, and top respectively in **Hoop tensile display**. The designer can pick up one of the values to determine sheet thickness based on his/her own experience. A **Close** pushbutton in hoop tensile display lets user return to sheet analysis display. A **Go Back** pushbutton in sheet analysis display allows user return to component check display.

If the user selects **bolt** to be examined in Component Check Display, the bolt analysis display will appear.

15. Bolt Analysis Display

A promptbox attached to sheet pitch was created for the requirement of determining the bolt space. A when-changed method is used to check the user whether the sheet pitch entered is satisfied the following requirement:

$$\text{Sheet Width} / \text{Sheet Pitch} = \text{Integer}$$

The PRL below declares this method

```

ATTRIBUTE Sheet Pitch NUMERIC
WHEN CHANGED
BEGIN
IF Sheet Width / Sheet Pitch <> INT (Sheet Width / Sheet Pitch) THEN
BEGIN
text OF ValidationMessage := "please enter another sheet pitch to make sure that sheet width
divided by sheet pitch is an Integer"
ASK ValidationMessage
END
END

```

As sheet analysis display, the user can select any tier for examining. A pushbutton **Continue** is attached to following when-changed method.

```

WITH Continues bolt analysis display SIMPLE
WHEN CHANGED
BEGIN
PURSUE tensile force at sheet bottom
visible OF main window := FALSE
output OF main window := Bolt Pattern Display
visible OF main window := TRUE
END

```

The tensile force at sheet bottom is determined by following when-needed method:

```

ATTRIBUTE tensile force at sheet bottom NUMERIC
WHEN NEEDED
BEGIN
tensile force at sheet bottom := Hoop tensile at sheet bottom × sheet width
END

```

A Radiobutton attached to compound attribute **Bolt Pattern** was created. Usually, there are four kinds of bolt patterns used in corrugated sheet bins, i.e.

- 1) one row of bolts with space equals to one corrugation pitch
- 2) one row of bolts with space equals to half of corrugation pitch
- 3) two rows of bolts with space equals to one corrugation pitch
- 4) two rows of bolts with space equals to half of corrugation pitch

A **Display** pushbutton attached to following when-changed method to display shear force applied to each bolt in bolt force display. The PRL below declares this method.

```

WITH Continues bolt display Display SIMPLE
WHEN CHANGED
BEGIN
PURSUE Shear Force in Each Bolt
output OF subwindow4 := Bolt Force Display
visible OF subwindow4 := TRUE
END

```

The shear force in each bolt will be determined by following when-needed method.

```

Attribute shear force in each bolt NUMERIC
BEGIN
shear force in each bolt := tensile force at sheet bottom / bolt number
END

```

The bolt number will be determined by following when-needed method.

```

ATTRIBUTE Bolt Number NUMERIC
WHEN NEEDED
BEGIN
IF Bolt Pattern IS one row of bolts with space equals to one corrugation pitch THEN
BEGIN
Bolt number := sheet width / sheet pitch
END
IF Bolt Pattern IS one row of bolts with space equals to half of corrugation pitch

```

```

OR Bolt Pattern IS two rows of bolts with space equals to one corrugation pitch THEN
BEGIN
Bolt number := 2 × sheet width / sheet pitch
END
IF Bolt Pattern IS two rows of bolts with space equals to half of corrugation pitch THEN
BEGIN
Bolt number := 4 × sheet width / sheet pitch
END
END

```

A pushbutton **Continue** attached to following when-changed method was created to determine bolt diameter based on the shear force applied to each bolt.

```

WITH Continue bolt pattern display SIMPLE
WHEN CHANGED
BEGIN
visible OF main window := FALSE
output OF main window := Bolt Material Display
visible OF main window := TRUE
END

```

16. Bolt Material Display

A Radiobutton attached to compound attribute **Bolt Material** was created in this display. This compound attribute consists of **Grade1, Grade2, Grade5, Grade7, and Grade8**. The user can select one kind of bolt material. A **Continue** pushbutton was attached to following when-changed method.

```

WITH Continues bolt material display SIMPLE
WHEN CHANGED
BEGIN
PURSUE Bolt Diameter
output OF subwindow4 := Bolt Diameter Display
visible OF subwindow4 := TRUE

```

END

The bolt diameter will be determined by following when-needed method:

```
ATTRIBUTE Bolt Diameter NUMERIC  
BEGIN  
Bolt Diameter := SQUR ( 4 × shear force in each bolt / ( 0.62 × tensile strength × 1000 × 0.145 ) × 1000  
END
```

The tensile strength will be determined by following rule group.

```
RULE 6:  
IF Bolt Material IS SAE Grade 1 THEN  
Tensile Strength := 60
```

```
RULE 7:  
IF Bolt Material IS SAE Grade 2 THEN  
Tensile Strength := 74
```

```
RULE 8:  
IF Bolt Material IS SAE Grade 5 THEN  
Tensile Strength := 120
```

```
RULE 9:  
IF Bolt Material IS SAE Grade 7 THEN  
Tensile Strength := 133
```

```
RULE 10:  
IF Bolt Material IS SAE Grade 8 THEN  
Tensile Strength := 150
```

A **Go Back** pushbutton in bolt material display allows user return to component check display.

If the user select **stiffener** to be checked in component check display, the stiffener analysis

display will appear.

17. Stiffener Analysis Display

A promptbox attached to attribute **stiffener number around circumference** was created. A pushbutton **Continue** was attached to following when-changed method.

```
WITH Continues stiffener display SIMPLE  
WHEN CHANGED  
BEGIN  
PURSUE Force applied to each stiffener  
output OF subwindow4 := Stiffener Force Display  
visible OF subwindow4 := TRUE  
END
```

The stiffener force will be determined by following when-needed method.

```
Attribute force applied to each stiffener NUMERIC  
WHEN NEEDED  
BEGIN  
IF Code Name IS CFBC 1990 THEN  
BEGIN  
force applied to each stiffener := ( $\pi \times$  Bin Diameter / Number of Stiffener)  $\times$  Vert_Fric_Pressure [ Tier  
Number OF Table1] OF CFBC / 1000  
END  
IF Code Name IS ASAE THEN  
BEGIN  
force applied to stiffener := ( $\pi \times$  Bin Diameter / Number of Stiffener)  $\times$  Vert_Wall_Load [ Tier Number OF  
Table1 | OF ASAE / 1000  
END  
END
```

A **Continue** pushbutton in stiffener analysis display is attached to **Output** display to present

the results of load consultation.

18. Output Display

Output display from the expert system includes tensile force in bin sheets, shear force on individual bolt, and compression force on each stiffener. Loads are provided for each tier of bin sheets over the full wall height.

Chapter 5

Test and Evaluation

5.1. Testing

The obvious objective of testing is to ensure that the expert system gives the correct results and prompts presented in the order which would mimic a human expert. Testing cannot be strictly separated as a phase in the development procedure, rather, it occurs continuously throughout development. As each new piece of knowledge is learned from the expert, it is added to the system and the expert must then perform a detailed evaluation of the system. Expert system tools are designed to complement this incremental development approach. Through testing, any errors or bias should be removed from the system. The reliability of this expert system has been evaluated for the Canadian Farm Building Code(1990) and EP433(1991). This was accomplished by comparing system output with hand calculations for a series of assumed conditions. In all cases, the expert system results agree with the hand calculations.

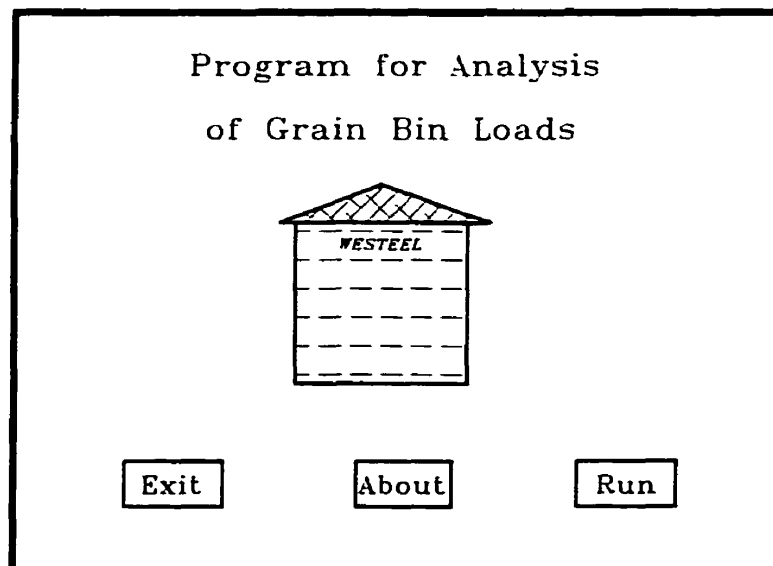
As well, data from the design office at Westeel were compared to system output. The error of lateral pressure between Westeel and this system is less than 6%, and error of vertical friction loads is less than 10%.

This program was tested extensively by the expert system developer and operated without any apparent computer error. The domain expert, Dr. M.G.Britton, who is an expert experienced in the design of grain storage bins, also tested the program and was satisfied with the user interface and the accuracy of results.

5.2. Example Application for CFBC 1990

The following example application illustrates the required inputs and outputs to run this expert system.

After starting program, the user is shown a title screen. the user clicks < **Run** > to continue.



The user is then asked to enter the grain amount (T) to be stored. After entering grain amount (in this example, the grain amount to be stored in the bin is 180 Tons), the user clicks < **Continue** > button.

How many tons of grain do you want to store in the bin ?

180

Exit Continue

The program then accesses the dBase file in which bin geometry was stored , and sends several qualified bins to screen. The user is asked to pick up one from the table. The user can get some information about how to select bin geometry by clicking < **Help** > button. Optionally, the user can click < **Other** > button to enter his/her own bin geometry. After selecting bin geometry, the user clicks < **Continue** > button to continue.

There are several geometric bins which satisfy your need, please select one. If you want to try different geometric bin, please press "Other" button.

Bin Diameter (m)	Bin Height (m)	Weight (T)
6	7	185
6	8	208
7	5	149

Other

Exit

Help

Continue

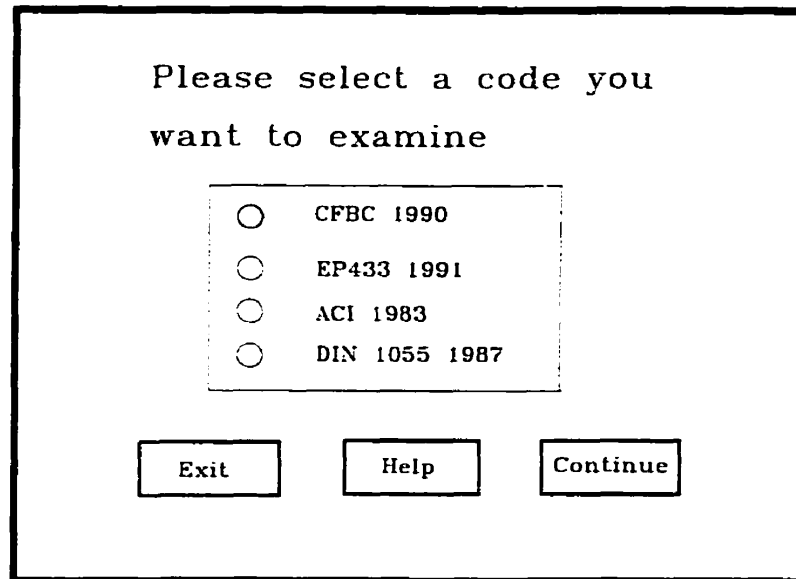
Next, the user is asked whether or not want to examine rectangular bins. The user clicks < NO > in this example to code selection display.

This is based on the calculation of circular bins. Would you like to try rectangular bins ?

Yes

No

In code selection display, the user is asked to select one code to be examined. The user selects CFBC 1990 and clicks < **Continue** > button.



Please select a code you want to examine

<input type="radio"/>	CFBC 1990
<input type="radio"/>	EP433 1991
<input type="radio"/>	ACI 1983
<input type="radio"/>	DIN 1055 1987

Exit

Help

Continue

Then the user is asked to select one type of grains to be stored in the bin. The user selects **Wheat** in this example and clicks < **Continue** > button.

Next, the user is asked to pick up a moisture content from list. In this example, the user pick up a 11 of moisture content, and clicks < **Continue** > button.

The user is then asked to select one wall material. After selecting **corrugated steel**, the user clicks < **Continue** > button.

Please select a grain from list

Wheat
Barley
Shelled Corn
Soybeans
Flaxseed
Canola

Exit

Continue

Please select a moisture
content (%) from list

11
13

Exit

Continue

Please select a wall material from list

Smooth Steel
Corrug. Steel
Plywood
Concrete

The program then accesses the dBase file in which material properties were stored, and sends the material properties to the user.

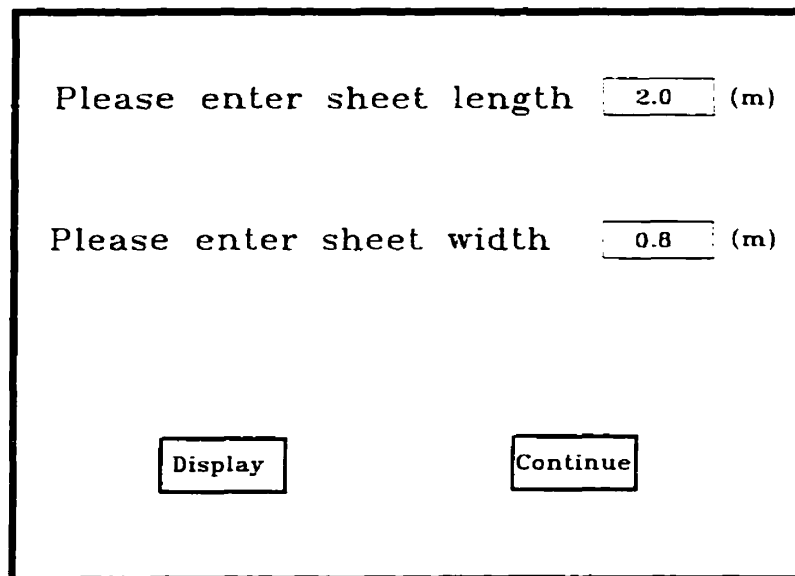
According to CFBC (1990) , the properties of selected grain is shown as follows :

<i>Coefficient of friction</i>	<input type="text" value="0.35"/>
<i>K - value</i>	<input type="text" value="0.6"/>
<i>Bulk density</i>	<input type="text" value="770"/>

Would you like to change the properties of selected grain ?

At this point, the user is asked if a different value would be chosen. This option allows the user enter a new value of material properties based on his/her own experience. In this example, the user clicks < NO > button.

Next, the user is asked to enter a sheet length and sheet width. After entering this two dimension, the user clicks < Display > button to display real bin geometry in subwindow.



Please enter sheet length (m)

Please enter sheet width (m)

The real bin diameter is (m)

The real bin height is (m)

The amount of grain to be stored in this bin is (T)

The user clicks < **Close** > button to close subwindow and returns to main window. Then user clicks < **Continue** > button to next screen.

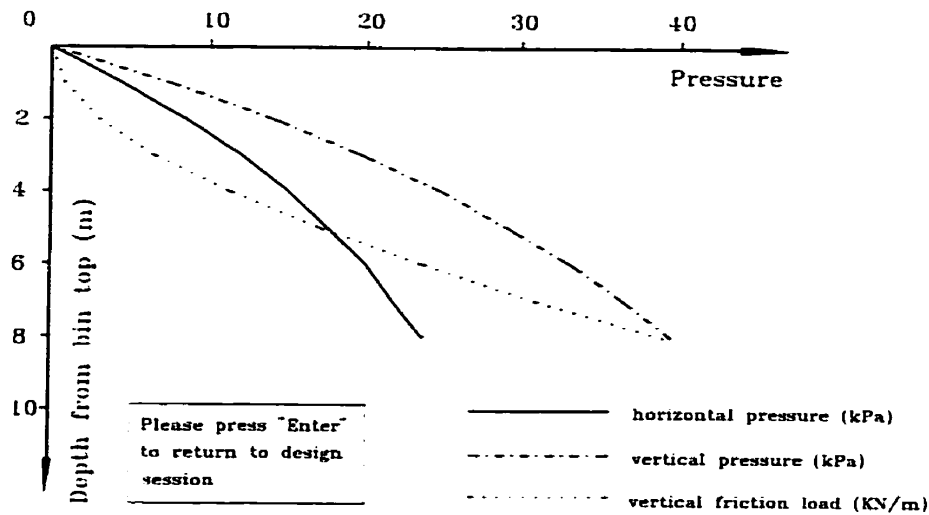
Would you like to see load data ?

Would you like to view load curves ?

In static load display screen, if user clicks < Yes > to see load data, the subwindow appears to display static loads in table.

Tier Number	Sta_Hori_Pressure (kPa)	Vert_pressure (kPa)	Vert_friction (KN/m)
1	3.64	6.07	0.51
2	6.92	11.54	2.00
3	9.86	16.47	4.36
4	12.54	20.90	7.51
5	14.93	24.88	11.36
6	17.08	28.47	15.85
7	19.01	31.69	20.91
8	20.76	34.60	26.48
9	22.32	37.21	32.52
10	23.73	39.56	38.97

By clicking < Close > button, the user will be brought to main window. If user clicks < Yes > to view load curves, the knowledge base calls external program, passes the parameters to external program. External program will send load curves to the screen. The user presses < Enter > to return to consultation.



Then the user clicks < **Continue** > button in load display to continue dynamic effect examination.

According to CFBC (1990),
It is necessary to examine
dynamic effect during
discharge

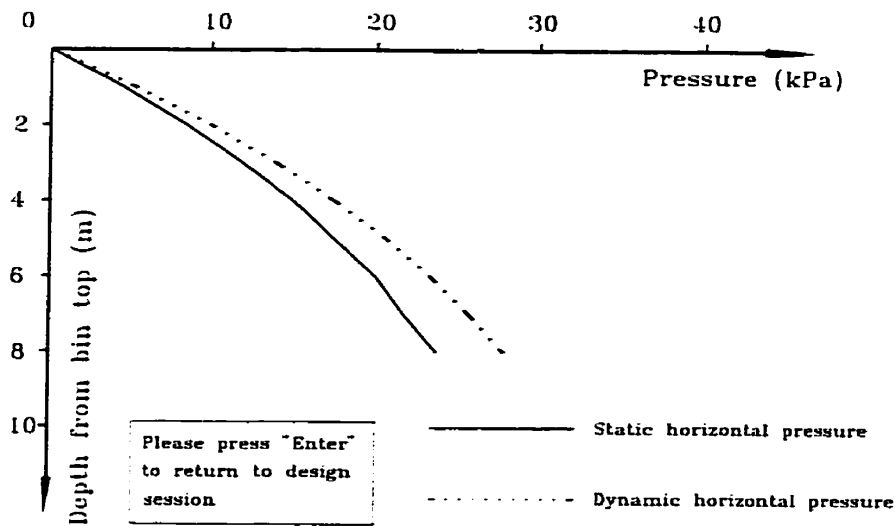
Exit Continue

The user can obtain some information on dynamic effect during discharge from dynamic effect prompt display.

Like static load display, the user can select to see dynamic load data or to view dynamic load curves in dynamic load display. If user clicks < Yes > to see load data, a table which presents the comparison dynamic horizontal pressure with static horizontal pressure will be shown in subwindow.

Tier Number	Static horizontal pressure (kPa)	Dynamic horizontal pressure (kPa)
1	3.84	4.82
2	6.92	8.13
3	9.88	11.60
4	12.54	14.72
5	14.93	17.53
6	17.08	20.06
7	19.01	22.33
8	20.76	24.38
9	22.32	26.22
10	23.73	27.88

If the user clicks < Yes > to view load curves, the program calls external program to display dynamic load curves.



Next, the user is asked whether a eccentric discharge will be examined. In this example, the user clicks < **NO** > button to continue.

The user is then asked to select one kind of components to be examined.

Please select an item which you want to examine ?

Sheet
 Bolt
 Stiffener

If the user selects sheet to be checked , the sheet analysis display appears after user clicks < Continue > button. In this display, the user is asked to select one tier of sheets. If user picks up tier No.6 to be examined, the depth (4.80 m) from bin top to the bottom of selected sheet will be presented in promptbox beside table. After user clicks < Display > button, the hoop tensile force will be shown in following subwindow.

Please select a tier you want to examine

Tier No.
1
2
3
4
5
6
7
8
9
10

Depth from bin top to selected sheet bottom

4.8 (m)

Help Display Go Back

The < Go Back > button in sheet analysis display lets user to return to component check display.

The hoop tensile force per unit height of bin wall

At sheet bottom	<u>63.86</u>	
At sheet middle	<u>59.94</u>	(KN/m)
At sheet top	<u>55.81</u>	

Note: The hoop tensile force can be used to determine sheet thickness

If user select bolt to be checked, the bolt analysis display will appear. The user is asked to enter sheet pitch. In this example, a 0.1 (m) of sheet pitch is entered.

Please enter sheet pitch (m) 0.1

Please select a tier you want to check

Tier No.	Depth from bin top to selected sheet bottom
1	
2	
3	
4	
5	
<u>6</u>	<u>4.8</u> (m)
7	
8	
9	
10	

After picking up a tier, the user clicks < **Continue** > button to continue. Next, the user is asked to select a bolt pattern. The user can get graphic help about bolt pattern by clicking < **Help** > button. In this example, the bolt pattern is set to two rows of bolts with bolt space equal to half of one corrugation pitch. The user then clicks < **Display** > for presenting bolt numbers for each vertical joint and shear force applied to each bolt in subwindow.

The tensile force (hoop tensile * sheet width) applied to this sheet is

51 (KN)

Please select bolt pattern

- one row of bolts with space equal to one corrugation pitch
- one row of bolts with space equal to half of a corrugation pitch
- two rows of bolts with space equal to one corrugation pitch
- two rows of bolts with space equal to half of a corrugation pitch

The number of bolts in each vertical joint is

The shear force applied to each bolt is

(KN)

After clicking < **Continue** > button, the user is asked to select a kind of bolt materials. In this example, the user selects **SAE Grade5** and clicks < **Display**) button to get bolt diameter.

Please select a bolt material

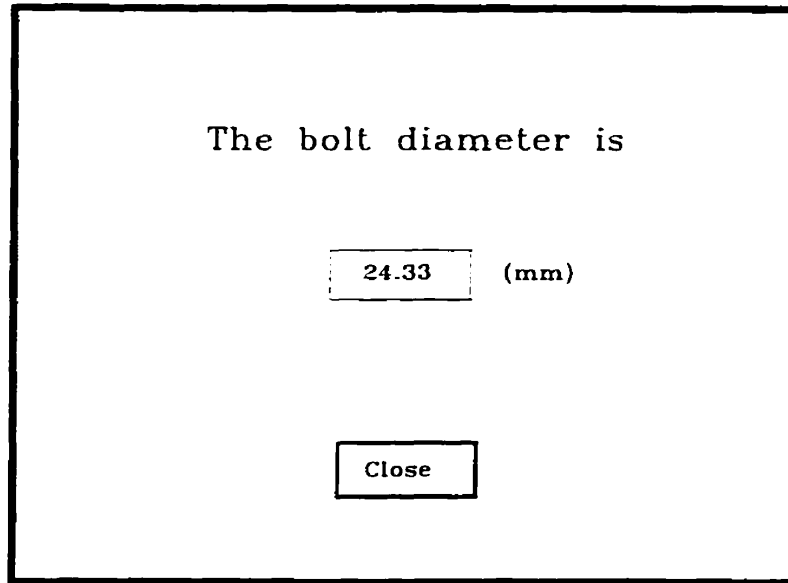
SAE Grade 1

SAE Grade 2

SAE Grade 5

SAE Grade 7

SAE Grade 8



The < **Go Back** > button lets user to return to component check display again.

Last, if user selects stiffener to be examined, the stiffener check display will appear. The user is asked to enter the amount of stiffeners around bin circumference. The user enters ten stiffeners in this example and picks up tier No.6 to be checked, then clicks < **Display** > button to display compression force applied to the select stiffener.

Then, the user clicks <**Continue**> button to see the output of load consultation.

How many stiffeners are there
around bin circumference ?

10

Please select a tier you want
to check

Tier No.
1
2
3
4
5
6
7
8
9
10

Depth from bin top to
selected sheet bottom

4.8

Continue

Display

Go Back

The compression force applied to
selected stiffener is

31.70 (KN)

Note: Assume "Column" length equals to bolt space
we can determine the dimension of this stiffener

Close

Output of Load Consultation

Tier Number	6
Tensile Force at Sheet Bottom (KN/m)	63.86
Shear Force on Each Bolt (KN)	1.59
Compression Force on Each Stiffener (KN)	31.70

Exit

Restart

5.3. Evaluation of Level5 Object

5.3.1. Integrating Expert System Technology with Database Technology

The advantage of integration of expert system with database technology is enormous. It can analyze the data stored in a database, create results, and store the results in a database so that other system can process it further.

The Level5 Object supports two database systems: dBase III and Focus. It also offers a more generic database interface through SQL (Structural Query Language) and third-part subsystem for remote database.

Level5 Object implements its dBase interface using an object-oriented approach. The dB3 system class provides direct read / write access to dBase III databases. The structure of the database file is brought into LEVEL5 by creating an instance of the dB3 system class.

In order to use a database file, a class corresponding to that file must be created. All attributes of that class, both attribute names and type, must correspond to the field of the dBase file. Level5 Object can create a class corresponding to a given dBase file automatically in Database Editor. The class will inherit all attributes of the dB3 system class.

Once a database has been incorporated as an object in a knowledge base, you can access

its data, and develop rules, demons and methods. The attributes of the dBase-derived classes can be used both in rules and methods within Level5 Object.

The Level5 Object's implementation of dBase interface is quite complete and is both easy to understand and use. Its object-oriented approach integrates well with the rest of the system. Most function needed for efficient database processing are supported but are some limitations in the dBase interface.

- 1) It requires the use of one particular database system - dBase.
- 2) Only 10 files can be open simultaneously and when index files are used only six files can be open.

However, the ability to integrate databases in Level5 is enough for developing the expert systems with limited database processing.

5.3.2. Calling external programs from a knowledge-base system

Expert system tools typically are very limited in math and graphic functions and those functions that they do have are very inefficient when compared to conventional programming languages. By allowing the expert system shell to call an external program or function, many of the above limitations can be eliminated. Allowing calls to external functions can significantly increase the capability and the performance of an expert system.

Two levels of implementation are possible in Level5 Object:

- calling an external program without further communication;

- calling an external program with the parameters and results passing.

The first method is very easy to use and does not require any special support from a caller other than the ability to start an external program. The second method is much more flexible and useful but also much more difficult to implement, especially when the parameters and results are passed through memory as opposed to disk files.

Level5 Object provides two commands for external programs

- **ACTIVATE**: used when an external program is to be called only once;
- **ESTABLISH**: used when an external program will be called more than once (it will stay in memory after the first call).

Two types of programs can be called from Level5 Object: **EXTERN** and **SERVER**.

EXTERN program is an external program that is called with optional command-line parameters and no further communication exists between the external program and LEVEL5 Object. It can be any MS Dos or MS window application. **SERVER** is an external program that can receive parameters from LEVER Object and pass back the results upon termination.

5.3.2.1 Calling an EXTERN program

An **EXTERN** program does not have to be written specifically to communicate with Level5 Object. Any program that runs under MS Dos or MS Window can be executed as an **EXTERN** program. The following is an example of a call to an **EXTERN** program.

WITH Load Display SIMPLE

WHEN CHANGED

BEGIN

filename OF file1 := "C:\15o25\ni\variable.txt"

action OF file1 IS open new := TRUE

write line OF file1 := TO STRING (Bulk density)

write line OF file1 := TO STRING (Coefficient of friction)

write line OF file1 := TO STRING (K_value)

write line OF file1 := TO STRING (Bin Diameter)

write line OF file1 := TO STRING (Bin Height)

action OF file 1 IS close := TRUE

ESTABLISH "IPU, EXTERN, C:\BP\ni\load.exe

filename OF file2 := " C:\15o25\ni\Count.txt"

action OF file2 IS open old := TRUE

read line OF file2 := TRUE

Count := TO NUMERIC (current line OF file2)

action OF file2 IS close := TRUE

filename OF file3 := " C:\15o25\ni\load.txt"

action OF file3 IS open old := TRUE

read line OF file3 := TRUE

FOR (i := i TO Count)

BEGIN


```

Sta_Hori_pressure [i] := TO NUMERIC ( current line OF file3 )

END

action OF file3 IS close := TRUE

END

```

After user clicks the **Display** button which attached to when-changed method " **Load Display**", Level5 Object calls the external program, passing the parameters to it through a text file. The external program reads the parameters from the file, calculates the static loads, and passes the results (static loads) back to Level5 Object through another text file.

5.3.2.2. Calling a SERVER program

A **SERVER** program is written specifically to communicate with Level5 Object. It calls functions that read the values of attributes in **SEND** statement and that writes values to **RECEIVE** statements. A **SERVER** program must be a MS Windows program. The following is an example of a call to a **SERVER** program from Level5 Object:

```

ACTIVATE "IPU, SERVER, C:\I5o25\ Prog.exe"

SEND name OF employer

SEND employee.sex IS male

RECEIVE salary OF employee

```

The **ACTIVATE** command starts the **SERVER** program Prog.exe and establishes communication between the server and Level5 Object. Next, Level5 Object sends two parameters to the server using **SEND** commands which should have the corresponding l5-read statements. After the last **SEND** command Level5 Object yields control to the server. After finishing the processing, the server returns the result to Level5 Object using one of l5-write statements. Level5 Object reads the result using the **RECEIVE** command. At this point the server terminates and control is returned to Level5 Object. If a Level5 Object application does not use the **RECEIVE** command, the server should use l5-quit statement to inform Level5 Object that it can continue processing.

Through the development of this expert system, it is clear that all the functionality necessary to call an external program is provided by Level5 Object. Calling **EXTERN** program is much easier and does not impose any special requirements on the external program. Parameters can be passed on the command-line, through ASCII files, or through dBase files. If passing parameters through the files is not efficient enough then **SERVER** program must be used. Writing a **SERVER** program imposes special requirements on the program and can be cumbersome but it allows passing parameters and results efficiently through memory.

5.3.3. Knowledge Base Management

Knowledge base management facilities become increasingly important when a knowledge base becomes larger and more complex. Level5 Object provides easy-to-use and well-designed

editors for the following elements of a knowledge base: rules, methods, objects and attributes. All editors are well integrated and easily accessed from menus and icon bars.

Level5 Object provides a history facility. It allows a user to store a history of a session to a text file in a readable format. The history shows all actions performed by Level5 Object's inference engine. The history facility is a very useful debugging and testing tool.

Level5 Object offers explanation facility through the expanded facet that can be attached to attributes. The expanded facet associates an attribute with a display window that can contain both textual and graphical explanation of the attribute, for example it can explain why the system is trying to determine a value of the attribute.

Level5 Object allows exporting a knowledge base to a text file. The knowledge base is stored in a readable text format - it is translated to Level5 Object's PRL (Production Rule Language). It is also possible to import a previously exported knowledge base. This allows developer to port Level5 Object-based applications to other hardware platforms.

Level5 Object offers a very-quality development tools when compared with other expert system shells, for example PcPLUS. It provides a wide range of tools which should satisfy most developers. Its extensive support for user interface allows quick development of attractive prototype system.

Chapter 6

Conclusions and Suggestion for Further Development

From the experience gained through the process of developing this expert system, and from the performance of the present system, the following conclusions are drawn:

1. An expert system was successfully developed to analyze grain bin pressures and reduce those pressures to loads on specific components of a grain bin. This expert system consists of knowledge base, database files, and external programs.
2. The information both from published source and unpublished heuristics of the domain experts was effectively captured into a frame-based expert system shell. Level5 Object expert system shell was found to be an acceptable platform upon which to develop this type of design aid.
3. This expert system was developed to provided bin designers with a rapid and accurate means of determining loads applied to individual bin components. The applicability of this technology to engineering design specific to steel grain bins has been confirmed.
4. Preliminary evaluation of the system indicated that its performance was satisfactory in

addressing the domain targeted. The system has been evaluated against conventional design calculations based on codes, In all trial, expert system results agree with the hand calculation. However, some areas of the knowledge base need to be improved a large domain addressed to provide solutions of better quality so that it may be developed for use. The following recommendations are given for further expansion of this expert system:

1. The standard database file including sheet dimensions should be established for user selection.
2. Although the corrugate steel grain bin is the domain system in the market, the smooth steel bin is still needed by farmer. Therefore, the expansion of this system for smooth steel bins is valuable. It can be easily done by adding some domain knowledge to knowledge base.
3. The principal loads for bin design come from of the action of the stored material. However, bins will be designed to resist other applicable loads such as dead load, snow loads, wind or earthquake loads, and thermal loads. For a practical program, the generalization of those loads must be considered. The determination of those loads is more arts than science. The bottleneck for developing is knowledge acquisition from domain expert.
4. The present system only can predict loads applied to specific components of a grain bin. The further study should expend this system to a practical system for optimizing the selection of structural members capable of carrying loads safely and economically. For example the further system should provide the user with sheet thickness, bolt and stiffener dimension for a selected

tier.

REFERENCES

- Abdel-Sayed, G., F. Monasa, and W. Siddall. 1985. Cold-formed steel farm structures Part I: Grain Bins. *J. of Struc. Eng.* 111(10): 2065-2089.
- ACI. 1983. Recommended practice for design and construction of concrete bins, silos and bunkers for storing granular materials and commentary. ACI 313, American Concrete Institute, Detroit, MI.
- Airy, W. 1897. The pressure of grain. *Proceedings of the Institute of Civil Engineers.* 131 : 347-358.
- ASAE. 1991. Loads exerted by free-flowing grain on bins. ASAE Standards. EP433. American Society of Agricultural Engineering. St. Joseph, MI. 509-512.
- Blight, G.E. 1985. Temperature changes affect overpressures in steel bins. *International Journal of Bulk Solids Storage in Silos.* 1(3): 1-7.
- Britton, M.G. 1969. Lateral pressures in deep bulk fertilizer storage bins. Unpublished M.Sc. Thesis, The University of Manitoba.
- Britton, M.G. 1973. Strain on deep bin walls due to ambient temperature decrease. Ph.D. Thesis. Texas A & M. College Station, TX, USA.
- Britton, M.G. and Q. Zhang. 1989. State-of-art in the design of grain storage structures. ASAE Paper No. 89-4535.
- Brzezinski, A. 1993. An evaluation of techniques and tools for integrating knowledge-base and conventional-computing systems. Unpublished M.Sc. Thesis. The University of Manitoba.
- CFBC. 1990. Canadian Farm Building Code. 1990. NRCC No. 21312, National Research Council of Canada. Ottawa, Canada.
- DIN. 1987. Design loads for buildings: loads on silos. DIN 1055. German Standard. Blatt 6. Deutsche Normen, Berlin.
- Doluschitz, R. and W.E. Schmisser. 1987. Expert system : application to agricultural and farm management. *Computers and Electronics in Agricultural.* 2(1) : 173-182.
- Embleton, K.M. 1990. An expert system approach to establish design snow and wind loads.

Unpublished M.Sc. Thesis, The University of Manitoba.

Fankhauser, C.D. 1977. Analysis and design of corrugated grain tank. ASAE Paper No: 77-4502.

Gebremedhin, K.G., S.S. Jagdale and R. Gupta. 1989. An expert system for optimizing computer aided design of post frame buildings. *Applied Engineering in Agriculture*. 5(3):447-452.

Gaylord, E.H. and C.N. Gaylord. 1985. Design of steel bins for storage of bulk solids. Prentice-Hill. New Jersey.

Hough, B.K. 1957. Basic Soils Engineering. New York, NY. The Ronald Press Co.

Information Buildings Inc. 1990. Level5 Object User's Guide. New York. NY 10001.

Jackson, P. 1990. Introduction to expert system. Saint Louis, Missouri. Addison-Wesley.

Jaky, J. 1948. Pressure in silos. Proceedings of second international conference on soil mechanics and foundation engineering, Rotterdam. 1 : 103-107.

Jenkyn, K.T. and D. J. Goodwill. 1987. Silo failures: lessons to be learned. *Engineering Digest*. September 1987. 17-22.

Jenike, A.W. and J.R. Johanson. 1968. Bin loads. *Journal of Structural Division, Proceeding of ASCE*, 94(ST4): 1011-1041.

Jofriet, J. C. and S.C. Negi. 1988. Tower silo design: Principles and computer implementation. *J. Agric. Eng. Res.* (40): 9-21.

Ketchum, M.S. 1919. The design of wells, bins and grain elevators. Third edition. Book Co., New York. NY.

Kramer, H.A. 1944. Factors inferencing the design of bulk bins for rough rice. *Agricultural Engineering* 25: 463-466.

Lenczner, D. 1963. An investigation into the behaviour of sand in a model silo. *The Structural Engineering*. 41(12): 389-398.

Manbeck, H.B., M.G. Goyal, G.L. Nelson and M.G. Singh. 1977. Dynamic overpressure in model bins during emptying. ASAE Paper No. 77-4505.

Manbeck, H.B. and L.M. Muzzelo. 1985. Measurement of thermally induced pressure in a model grain bin. *Transaction of the ASAE*. 28(4):1253-1258.

- Moran, P. 1991. Static design loads on circular plan retaining walls for granular materials. *J. Agric. Engng. Res.* 48: 145-152.
- Ni, Z., M.G. Britton and Q. Zhang. 1994. An expert system for analysis of grain bin loads. ASAE Paper No. 94-4026.
- Olynick, D.M. 1993. An expert system for the fire protection requirements of the national building code of Canada 1990. Unpublished M.Sc. Thesis, The University of Manitoba.
- Reimbert, M. and A. Reimbert. 1956. *Silos traite theorique et pratique*. Edition Eyrollas, Paris.
- Reimbert, M. and A. Reimbert. 1976. *Silos: theory and practice*. Clauathal, Germany: Trans Tech Publications.
- Reimbert, M. and A. Reimbert. 1987. *Silos - theory and practice*, 2nd Edition in English. Lavoisier Publishing, New York, NY, USA.
- Ross, I.J., T.C. Gridges, O.J. Loewer and J.N. Walker. 1979. Grain bin loads as affected by grain moisture content and vertical pressure. *Transactions of ASAE*. 26(4): 592-597.
- Safarian, S.S. and E.C. Harris. 1985. *Design and construction of silos and bunkers*. Van Nostrand Reinhold Company Inc.
- Schott, R.W. 1990. Vertical strength examination of grain bin stiffeners. Unpublished M.Sc. Thesis. The University of Manitoba.
- Trahair, N.S. 1985. Structural design criteria for steel bins and silos. *Int. J. Bulk Solids Storage in Silos* 1(2). 11-20.
- Turban, E. and P. Watkins. 1988. *Applied expert system*. Amsterdam.
- Versavel, P.A. and M.G. Britton. 1986. Interaction of bulk wheat with bin wall configuration in model bins. *ASAE*. 29(2) : 533 - 537.
- Watson, D.G. and R.C. 1990. Aeration system design for flat grain storages with an expert system. *ASAE* 6(2) : 183-188.

Appendix A. Database for bin geometry

Bin Diameter (m)	Bin Height (m)	Stored Capability (Ton)
4.0	3.0	38
4.0	4.0	48
4.0	5.0	58
4.0	6.0	68
4.0	7.0	79
4.0	8.0	89
4.0	9.0	99
4.0	10.0	109
5.0	3.0	61
5.0	4.0	77
5.0	5.0	93
5.0	6.0	110
5.0	7.0	126
5.0	8.0	142
5.0	9.0	158
5.0	10	174
6.0	4.0	115
6.0	5.0	138
6.0	6.0	162
6.0	7.0	185
6.0	8.0	208
6.0	9.0	237
6.0	10	254
7.0	4.0	162
7.0	5.0	194
7.0	6.0	225
7.0	7.0	256
7.0	8.0	288
7.0	9.0	319
7.0	10	351
8.0	4.0	218
8.0	5.0	260
8.0	6.0	301
8.0	7.0	342
8.0	8.0	383
8.0	9.0	424
8.0	10	465

Appendix B. Database for Grain Properties

Grain	MC (%)	WM	μ	K	w (kg/m ³)
Wheat	11.0	Smooth Steel	0.10	0.4	770
Wheat	11.0	Corrug. Steel	0.35	0.6	770
Wheat	11.0	Plywood	0.30	0.6	770
Wheat	11.0	Concrete	0.35	0.6	770
Wheat	13.0	Smooth Steel	0.25	0.4	770
Wheat	13.0	Corrug. Steel	0.35	0.6	770
Wheat	13.0	Plywood	0.30	0.6	770
Wheat	13.0	Concrete	0.35	0.6	770
Barley	11.0	Smooth Steel	0.10	0.4	620
Barley	11.0	Corrug. Steel	0.35	0.6	620
Barley	11.0	Plywood	0.30	0.6	620
Barley	11.0	Concrete	0.35	0.6	620
Barley	13.0	Smooth Steel	0.25	0.4	620
Barley	13.0	Corrug. Steel	0.35	0.6	620
Barley	13.0	Plywood	0.30	0.6	620
Barley	13.0	Concrete	0.35	0.6	620
Shelled Corn	11.0	Smooth Steel	0.20	0.4	720
Shelled Corn	11.0	Corrug. Steel	0.35	0.6	720
Shelled Corn	11.0	Plywood	0.30	0.6	720
Shelled Corn	1.0	Concrete	0.35	0.6	720
Shelled Corn	16.0	Smooth Steel	0.35	0.4	720
Shelled Corn	16.0	Corrug. Steel	0.35	0.6	720
Shelled Corn	16.0	Plywood	0.45	0.6	720
Shelled Corn	16.0	Concrete	0.60	0.6	720
Soybeans	11.0	Smooth Steel	0.20	0.4	770
Soybeans	11.0	Corrug. Steel	0.35	0.6	770
Soybeans	11.0	Plywood	0.30	0.6	770
Soybeans	11.0	Concrete	0.50	0.6	770
Flaxseed	9.0	Smooth Steel	0.20	0.4	770
Flaxseed	9.0	Corrug. Steel	0.35	0.6	770
Flaxseed	9.0	Plywood	0.35	0.6	770
Flaxseed	9.0	Concrete	0.35	0.6	770
Flaxseed	11.5	Smooth Steel	0.25	0.4	770

Flaxseed	11.5	Corrug. Steel	0.35	0.6	770
Flaxseed	11.5	Plywood	0.40	0.6	770
Flaxsed	11.5	Concrete	0.45	0.6	770
Canola	9.0	Smooth Steel	0.20	0.4	640
Canola	9.0	Corrug. Steel	0.35	0.6	640
Canola	9.0	Plywood	0.35	0.6	640
Canola	9.0	Concrete	0.35	0.6	640
Canola	12.5	Smooth Steel	0.25	0.4	640
Canola	12.5	Corrug. Steel	0.35	0.6	640
Canola	12.5	Plywood	0.35	0.6	640
Canola	12.5	Concrete	0.35	0.6	640

Appendix C. List of External Program Code

Program for displaying static loads of CFBC 1990

Program Loads;

Uses Crt, Graph;

Var

k, Tier_Number, Height : Integer;

Static_hori_pressure, Static_vert_pressure, Static_vert_friction : Real;

tf, ef : Text;

a,b,c,d : ARRAY[0..30] OF Word;

Procedure Graphic ;

Var

grDriver, grMode, grError : Integer;

i, x0, x1, y0, y1 : Integer;

factor_x, factor_y : Real;

Pressure, Height : String [4];

Begin

grDriver := Detect;

InitGraph(grDriver, grMode, 'C:\bp\bgi');

SetViewPort(50, 50, 200, 150, Clipoff); { set origin to (50, 50) }

SetLineStyle(0, 0, 3);

SetBkColor (1);

x0 := 0;

x1 := 300;

Y0 := 0;

Y1 := 300;

factor_x := (x1-x0) / (c[k]/1000);

factor_y := (250) / Tier_Number;

OutTextXY (x0, y0-20, '0');

```

For i := 1 To Round (c[k]/1000 +10) Do                                { make x axis and y axis }
If (i/10) = i Div 10 Then
  Begin
    Str(i, Pressure);
    MoveTo(Round(i * factor_x), y0);
    LineTo(Round(i * factor_x), -5);
    OutTextXY (Round(x0 + i * factor_x) - 5, Y0 - 20, Pressure);
  End;
For i := 1 To Tier_Number + 1 Do
  Begin
    Str(i, Height);
    MoveTo(x0, Round(i * factor_y));
    LineTo(5, Round(i * factor_y));
    OutTextXY (x0 - 20, Round(Y0 + i * factor_y -3), Height);
  End;
SetColor( Red );
For i := 0 To Tier_Number -1 Do
  Begin
    Line( Round(a[i] * factor_x / 1000), Round(b[i] * factor_y), Round( a[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  End;
SetColor( Green );
For i:= 0 To Tier_Number -1 Do
  Begin
    Line( Round(c[i] * factor_x / 1000), Round(b[i] * factor_y), Round( c[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  End;
SetColor(Yellow);
For i:= 0 To Tier Number -1 Do
  Begin
    Line( Round(d[i] * factor_x / 1000), Round(b[i] * factor_y), Round( d[i+1] * factor_x /1000) , Round(b[i+1] *
factor_y ));
  end;
SetColor( White );
MoveTo ( 0, 0 );
LineTo (440, 0);
MoveTo ( 0, 0 );

```

```

LineTo (0, 360);
MoveTo (410, 0);
LineTo (410, 3);
LineTo (440, 0);
LineTo (410, -3);
LineTo (410, 0);
MoveTo (-3, 330);
LineTo (3, 330);
LineTo (0, 360);
LineTo (-3, 330);
SetColor( White );
SetTextStyle( 0 , 0 , 1 );
OutTextXY(360, 20, 'Pressures ' );
SetTextStyle( 0 , 1 , 1 );
OutTextXY(20, 220, 'Tier Number ' );
SetTextStyle( 0, 0, 1);
SetColor( Red );
MoveTo( 250, y1 + 60 );
LineTo( 310, y1 + 60 );
SetColor( Green );
MoveTo( 250, y1 + 80 );
LineTo( 310, y1 + 80 );
SetColor (Yellow);
MoveTo(250, y1 + 100);
LineTo(310, y1 + 100);
SetColor( White );
SetTextStyle( 0, 0, 1 );
OutTextXY(320, y1 + 55, 'Static Horizontal Pressure (kPa) ');
OutTextXY(320, y1 + 75, 'Static Vertical Pressure (kPa) ');
OutTextXY(320, y1 + 95, 'Static Vertical Friction (KN/m)');
SetFillStyle (SolidFill, 3);
Bar(10, y1 + 65, 230, Y1 + 105); {make an area for text}
SetColor (Yellow);
SetTextStyle(0, 0, 1);
OutTextXY( 15, Y1 + 70, ' Please press "Enter" to' );
OutTextXY( 15, Y1 + 90, ' return to design session');
Readln;

```

```

    CloseGraph;
End;

Begin
    Assign (ef, 'c:\15o25\mi\height.txt');
    Reset (ef);
    Read (ef, Tier_Number);
    Close (ef);
    Assign (tf, 'c:\15o25\mi\staload.txt');
    Reset (tf);
    For Height := 0 To Tier_Number Do
        Begin
            Read (tf, Static_hori_pressure, Static_vert_pressure, Static_vert_friction);
            a[Height] := Round (Static_hori_pressure);
            b[Height] := Height;
            c[Height] := Round (Static_vert_pressure);
            d[Height] := Round (Static_vert_friction);
        End;
    Close (tf);
    k := Tier_Number;
    Graphic;
End.

```

Program for displaying dynamic horizontal pressure of CFBC 1990

```

Program Loads;
Uses Crt, Graph;

```

```

Var
    k, Tier_Number, Height : Integer;

```



```
Static_hori_pressure, Dynamic_hori_pressure : Real;  
tf, ef : Text;  
a,b,c : ARRAY[0..30] OF Word;
```

```
Procedure Graphic ;
```

```
Var
```

```
grDriver, grMode, grError : Integer;  
i, x0, x1, y0, y1 : Integer;  
factor_x, factor_y : Real;  
Pressure, Height : String [4];
```

```
Begin
```

```
grDriver := Detect;  
InitGraph( grDriver, grMode, 'C:\bp\bgi' );  
SetViewPort(50, 50, 200, 150, Clipoff);  
SetLineStyle(0, 0, 3);  
SetBkColor (1);  
x0 := 0;  
x1 := 300;  
Y0 := 0;  
Y1 := 300;  
factor_x := (x1-x0) / (c[k]/1000);  
factor_y := (250) / Tier_Number;  
OutTextXY (x0, y0-20, '0');  
For i := 1 To Round (c[k]/1000 +10) Do  
If (i/10) = i Div 10 Then  
Begin  
Str(i, Pressure);  
MoveTo(Round(i * factor_x), y0);  
LineTo(Round(i * factor_x), -5);  
OutTextXY (Round(x0 + i * factor_x) - 5, Y0 - 20, Pressure);  
End;  
For i := 1 To Tier_Number + 1 Do  
Begin  
Str(i, Height);  
MoveTo(x0, Round(i * factor_y));  
LineTo(5, Round(i * factor_y));  
OutTextXY (x0 - 20, Round(Y0 + i * factor_y -3), Height);
```

```

    End;
SetColor( Red );
For i := 0 To Tier_Number -1 Do
Begin
Line( Round(a[i] * factor_x / 1000), Round(b[i] * factor_y), Round( a[i+1] * factor_x / 1000) ,Round(b[i+1] *
factor_y ));
End;
SetColor( Green );
For i:= 0 To Tier_Number -1 Do
Begin
Line( Round(c[i] * factor_x / 1000), Round(b[i] * factor_y), Round( c[i+1] * factor_x / 1000) ,Round(b[i+1] *
factor_y ));
End;
SetColor( White );
MoveTo ( 0, 0 );
LineTo (440, 0);
MoveTo ( 0, 0 );
LineTo (0, 360);
MoveTo (410, 0);
LineTo (410, 3);
LineTo (440, 0);
LineTo (410, -3);
LineTo (410, 0);
MoveTo (-3, 330);
LineTo (3, 330);
LineTo (0, 360);
LineTo (-3, 330);
SetColor( White );
SetTextStyle( 0 , 0 , 1 );
OutTextXY(360, 20, 'Pressures ' );
SetTextStyle( 0 , 1 , 1 );
OutTextXY(20, 220, 'Tier Number ' );
SetTextStyle( 0, 0, 1);
SetColor( Red );
MoveTo( 250, y1 + 60 );
LineTo( 310, y1 + 60 );
SetColor( Green );

```

```

MoveTo( 250, y1 + 90 );
LineTo( 310, y1 + 90 );
SetColor( White );
SetTextStyle( 0, 0, 1 );
OutTextXY(320, y1 + 55, 'Static Horizontal Pressure (kPa)');
OutTextXY(320, y1 + 85, 'Dynamic Horizontal Pressure (kPa)');
SetFillStyle (SolidFill, 3);
Bar(10, y1 + 55, 230, Y1 + 95);
SetColor (Yellow);
SetTextStyle(0, 0, 1);
OutTextXY( 15, Y1 + 60, ' Please press "Enter" to' );
OutTextXY( 15, Y1 + 80, ' return to design session');
Readln;
CloseGraph;
End;

```

Begin

```

Assign (ef, 'c:\15o25\ni\height.txt');
Reset (ef);
Read (ef, Tier_Number);
Close (ef);
Assign (tf, 'c:\15o25\ni\dyload.txt');
Reset (tf);
For Height := 0 To Tier_Number Do
Begin
Read (tf, Static_hori_pressure, Dynamic_hori_pressure);
a[Height] := Round (Static_hori_pressure);
b[Height] := Height;
c[Height] := Round (Dynamic_hori_pressure);
End;
Close (tf);
k := Tier Number ;
Graphic;
End.

```

Program for displaying static loads of EP433 1991

Program Loads;

Uses Crt, Graph;

Var

k, Tier_Number, Height : Integer;

Static_hori_pressure, Static_vert_pressure, Shear_stress, Vertical_wall_pressure : Real;

tf, ef : Text;

a,b,c,d : ARRAY[0..30] OF Word;

e : ARRAY[0..30] OF LongInt;

Procedure Graphic ;

Var

grDriver, grMode, grError : Integer;

i, x0, x1, y0, y1 : Integer;

factor_x, factor_y : Real;

Pressure, Height : String [4];

Begin

grDriver := Detect;

InitGraph(grDriver, grMode, 'C:\bp\bgi');

SetViewPort(50, 50, 200, 150, Clipoff);

SetLineStyle(0, 0, 3);

SetBkColor (1);

x0 := 0;

x1 := 300;

Y0 := 0;

Y1 := 300;

factor_x := (x1-x0) / (e[k]/1000);

factor_y := (250) / Tier_Number;

OutTextXY (x0, y0-20, '0');

For i := 1 To Round (e[k]/1000 +10) Do

If (i/10) = i Div 10 Then

```

Begin
  Str(i, Pressure);
  MoveTo(Round(i * factor_x), y0);
  LineTo(Round(i * factor_x), -5);
  OutTextXY (Round(x0 + i * factor_x) - 5, Y0 - 20, Pressure);
End;
For i := 1 To Tier_Number + 1 Do
  Begin
    Str(i, Height);
    MoveTo(x0, Round(i * factor_y));
    LineTo(5, Round(i * factor_y));
    OutTextXY (x0 - 20, Round(Y0 + i * factor_y - 3), Height);
  End;
SetColor( Red );
For i := 0 To Tier_Number - 1 Do
  Begin
    Line( Round(a[i] * factor_x / 1000), Round(b[i] * factor_y), Round( a[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  End;
SetColor( Green );
For i:= 0 To Tier_Number - 1 Do
  Begin
    Line( Round(c[i] * factor_x / 1000), Round(b[i] * factor_y), Round( c[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  End;
SetColor(Yellow);
For i:= 0 To Tier_Number - 1 Do
  Begin
    Line( Round(d[i] * factor_x / 1000), Round(b[i] * factor_y), Round( d[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  end;
SetColor(11);
For i:= 0 To Tier_Number - 1 Do
  Begin
    Line( Round(e[i] * factor_x / 1000), Round(b[i] * factor_y), Round( e[i+1] * factor_x / 1000) , Round(b[i+1] *
factor_y ));
  end;
end;

```

```
SetColor( White );
MoveTo ( 0, 0 );
LineTo (440, 0);
MoveTo ( 0, 0 );
LineTo (0, 360);
MoveTo (410, 0);
LineTo (410, 3);
LineTo (440, 0);
LineTo (410, -3);
LineTo (410, 0);
MoveTo (-3, 330);
LineTo (3, 330);
LineTo (0, 360);
LineTo (-3, 330);
SetColor( White );
SetTextStyle( 0, 0, 1 );
OutTextXY(360, 20, 'Pressures ' );
SetTextStyle( 0, 1, 1 );
OutTextXY(20, 220, 'Tier Number ' );
SetTextStyle( 0, 0, 1);
SetColor( Red );
MoveTo( 250, y1 + 60 );
LineTo( 310, y1 + 60 );
SetColor( Green );
MoveTo( 250, y1 + 80 );
LineTo( 310, y1 + 80 );
SetColor (Yellow);
MoveTo(250, y1 + 100);
LineTo(310, y1 + 100);
SetColor (11);
MoveTo(250, y1 + 120);
LineTo(310, y1 + 120);
SetColor( White );
SetTextStyle( 0, 0, 1 );
OutTextXY(320, y1 + 55, 'Static Horizontal Pressure (kPa) ' );
OutTextXY(320, y1 + 75, 'Static Vertical Pressure (kPa) ' );
OutTextXY(320, y1 + 95, 'Shear Stress (KPa)');
```

```
OutTextXY(320, y1 + 115, 'Vertical Wall Friction (KN/m)');
SetFillStyle (SolidFill, 3);
Bar(10, y1 + 75, 230, Y1 + 115);
SetColor (Yellow);
SetTextStyle(0, 0, 1);
OutTextXY( 15, Y1 + 80, ' Please press "Enter" to ' );
OutTextXY( 15, Y1 + 100, ' return to design session');
Readln;
CloseGraph;
End;
```

Begin

```
Assign (ef, 'c:\15o25\ni\height.txt');
Reset (ef);
Read (ef, Tier_Number);
Close (ef);
Assign (tf, 'c:\15o25\ni\Astaload.txt');
Reset (tf);
For Height := 0 To Tier_Number Do
Begin
Read (tf, Static_hori_pressure, Static_vert_pressure, Shear_stress, Vertical_wall_pressure);
a[Height] := Round (Static_hori_pressure);
b[Height] := Height;
c[Height] := Round (Static_vert_pressure);
d[Height] := Round (Shear_stress);
e[Height] := Round (Vertical_wall_pressure);
End;
Close (tf);
k := Tier_Number;
Graphic;
End.
```

Program for display dynamic loads applied to the hopper of EP433 1991

Program Hopper_Loads;

Uses Crt, Graph;

Var

Depth, Hopper_height, Normal_pressure, Tangential_stress : Real;

tf, ef : Text;

a,c : ARRAY[0..100] OF LongInt;

b : ARRAY[0..100] OF Real;

Count, i, k : Integer;

x0, x1, y0, y1 : Word;

factor_x, factor_y : Real;

Pressure, Height : String [4];

Procedure Graphic ;

Var

grDriver, grMode, grError : Integer;

i,j : Integer;

x0, x1, y0, y1 : integer;

factor_x, factor_y : Real;

Pressure, Height : String [4];

Begin

grDriver := Detect;

InitGraph(grDriver, grMode, 'C:\bp\bgi');

SetViewPort(50,50, 200,150, Clipoff);

SetLineStyle (0, 0, 3);

SetBkColor (1);

X0 := 0;

x1 := 300;

Y0 := 0;

Y1 := 300;

factor_x := (x1 - x0) / (a[1]/1000) ;

factor_y := (y1 - y0) / Hopper_height;


```

OutTextXY (x0, y0 - 20, '0');
For i := 1 To Round (a[1]/1000 + 10) Do
If (i / 10) = i Div 10 Then
  Begin
    Str(i, Pressure);
    MoveTo(Round(i * factor_x), y0);
    LineTo(Round(i * factor_x), -5);
    OutTextXY (Round(x0 + i * factor_x) - 5, Y0 - 20, Pressure);
  End;
If (Hopper_height <= 2.0) Then
For i := 1 To Round( Hopper_height * 10 + 1) Do
If (i / 2) = i Div 2 Then
  Begin
    Str(i/10 : 3 : 1, Height);
    MoveTo(x0, Round(i/10 * factor_y));
    LineTo( 5, Round(i/10 * factor_y));
    OutTextXY (X0 + 20, Round(y0 + i/10 * factor_y - 3) , Height);
  End;
If (Hopper_height > 2.0) Then
For i := 1 To Round(Hopper_height *10 + 4) Do
If (i / 5) = i Div 5 Then
  Begin
    Str(i/10 : 3 : 1, Height);
    MoveTo(x0, Round(i/10 * factor_y));
    LineTo( 5, Round(i/10 * factor_y));
    OutTextXY (X0 + 20, Round(y0 + i/10 * factor_y - 3) , Height);
  End;
SetColor( Red );
For j := 1 To k-1 Do
  Begin
    Line( Round(a[j] * factor_x / 1000), Round(b[j] * factor_y) , Round(a[j+1] * factor_x / 1000), Round(b[j+1] *
factor_y));
  End;
SetColor( Yellow );
For j := 1 To k-1 Do
  Begin
    Line( Round(c[j] * factor_x / 1000), Round(b[j] * factor_y), Round(c[j+1] * factor_x /1000), Round(b[j+1] *

```

```

factor_y));
End;
SetColor( White );
MoveTo ( 0, 0 );
LineTo (440, 0);
MoveTo ( 0, 0 );
LineTo (0, 360);
SetColor ( White );
MoveTo (410, 0);
LineTo (410, 3);
LineTo (440, 0);
LineTo (410, -3);
LineTo (410, 0);
MoveTo (-3, 330);
LineTo (3, 330);
LineTo (0, 360);
LineTo (-3,330);
SetColor( White );
SetTextStyle( 0 , 0 , 1 );
OutTextXY(360, 20, 'Pressures ' );
SetTextStyle( 0 , 1 , 1 );
OutTextXY(-15, 160, 'Height from hopper top (m) ' );
SetTextStyle ( 0, 0, 1);
SetColor( Yellow );
MoveTo( 250, y1 + 60);
LineTo(310, y1 + 60);
SetColor( Red );
MoveTo( 250, y1 + 90);
LineTo( 310, y1 + 90);
SetTextStyle( 0, 0, 1 );
SetColor(White);
OutTextXY(320, y1 + 55, 'Dynamic Tangential Stress (kPa) ' );
OutTextXY(320, Y1 + 85, 'Dynamic Normal Pressure (kPa) ' );
SetFillStyle (SolidFill, 3);
Bar(10, y1 + 55, 230, Y1+95);
SetColor (Yellow);
SetTextStyle(0, 0, 1);

```

```
OutTextXY(15, Y1 + 60, 'Please press "Enter" to');  
OutTextXY(15, y1 + 80, 'return to design session');  
CloseGraph;  
End;
```

```
Begin
```

```
Assign (ef, 'c:\15o25\ni\count.txt');  
Reset (ef);  
Read (ef, Count, Hopper_height);  
Close (ef);  
Assign (tf, 'c:\15o25\ni\A.dyload.txt');  
Reset (tf);  
i := 1;  
While i <= Count Do  
Begin  
Read (tf, Depth, Normal_pressure, Tangential_stress);  
a[i] := Round (Normal_pressure);  
b[i] := Depth;  
c[i] := Round (Tangential_stress);  
i := i + 1 ;  
End;  
Close (tf);  
k := i - 1;  
Graphic;  
End.
```