INTELLIGENT SELECTION OF GRINDING CONDITIONS

YAN LI

A thesis submitted in partial fulfilment of the
requirement of Liverpool John Moores University
for the degree of Doctor of Philosophy

September 1996

SOME DIAGRAMS

EXCLUDED ON

INSTRUCTION FROM

THE UNIVERSITY

Abstract

This thesis describes an investigation concerned with the development of an intelligent system for selection of grinding conditions. Based on a study of previous techniques for selection of grinding conditions, the study of grinding properties and the study of artificial intelligence techniques, the thesis proposes a multi-intelligent agent method for the selection of grinding conditions. The agents consist of case based reasoning, a neural network and rule based reasoning as well as the operator.

It is proposed that a neural network is an appropriate method for selecting grinding wheels because there is no model available for the wheel selection process. The neural network method of selecting the grinding wheel was investigated and an independent system was developed. The system was trained using data from catalogues. It was shown that the neural network was capable of learning the relationship between the wheel and the grinding process without a requirement for the knowledge engineer to fully understand the knowledge domain. A general neural network prototype system was developed to provide a tool for development of the grinding wheel selection system.

It is proposed that a case based reasoning method is appropriate for selection of grinding conditions because information for new technology, different processes and materials can be automatically incorporated into the system. A case based reasoning method for the selection of grinding conditions was developed. Techniques of case indexation, retrieval, modification, test and update were developed. The system was demonstrated to successfully learn new cases and make inferences for new problems.

A rule based reasoning agent for the approximate selection of grinding conditions was developed because of the lack of initial data for the case based reasoning system. The knowledge was based on previous research and data acquired from handbooks.

A blackboard method was used as a means of integrating the above systems as a multi-agent system. The system was developed and its performance evaluated by comparison with results from handbooks. The system works as expected and demonstrates the potential of using artificial intelligence for selection of grinding conditions.

Acknowledgements

The author would like to acknowledge the contributions of all of those who assisted towards the completion of this work. Particular gratitude is due to the director of studies, Professor W. Brian Rowe, for his invaluable guidance and encouragement throughout this work.

Thanks are also expressed to Professor Ben Mills, and Professor J. L. Moruzzi, for their help and advice.

Further thanks go to all members of staff in the Advanced Manufacturing Technology Research Laboratory for their practical help.

Acknowledgements are due to the Chinese National Education Committee for initial funding to visit the Liverpool John Moores University, and to the Liverpool John Moores University for providing the research funding necessary for me to undertake the doctoral research programme.

Finally I wish to thank my family and in particular my wife Wang Xiaoli, for their continued encouragement and support over past years.

Nomenclature

a	depth of cut
a_d	dressing depth
ACO	adaptive control with optimisation
AI	artificial intelligence
ANN	artificial neural network
A_r	mean chip aspect ratio
b	grinding width
b_c	grinding chip width
BP	error back propagation
c	specific heat capacity
C	heat flux distribution factor
CBR	case based reasoning
d_e	equivalent diameter
d_{eo}	equivalent diameter in an existing case
d_{en}	equivalent diameter in the Problem
d_{pj}	desired output of the ith node for pattern p
d_s	wheel diameter
d_w	workpiece diameter
d_{wo}	workpiece diameter in an existing case
d_{wn}	workpiece diameter in the Problem
e_c	specific grinding energy
E_p	square error function
E_a	root mean square error function
f	exponent
f_d	dressing lead
f_{do}	dressing lead in an existing case
f_{dn}	dressing lead in the Solution
FFN	feedforward network

$F_n^{'}$ specific normal grinding force

$F_t^{'}$ specific tangential grinding force

F_1 constant

F_2 constant

g exponent

G grinding ratio

G_1 constant

h_{eq} equivalent chip thickness

h_m maximum chip thickness

KBS knowledge based system

l_c chip length

l_d dynamic grit spacing

l_e real contact length

l_g geometric contact length

n_d number of dressing passes

net_j unshaped output of the jth node

o_j output of the jth node

o_{pj} actual output of the jth node for pattern p

P grinding power

R_a workpiece surface roughness

R_{ao} workpiece surface roughness in an existing case

R_{an} workpiece surface roughness in the Problem

RBR rule based reasoning

RMS root mean square

R_c Rockwell Hardness

R_w fraction of the total energy partitioned to the workpiece

R_1 constant

$Q_w^{'}$ grinding removal rate per unit wheel contact width

v_f wheel axis infeed speed

v_{fo} wheel axis infeed speed in an existing case

v_{fn} wheel axis infeed speed in the Solution

V_m mean grinding chip volume

v_w workpiece speed

v_{wo} workpiece speed in an existing case

v_{wn} workpiece speed in the Solution

v_s wheel speed

v_{so} wheel speed in an existing case

v_{sn} wheel speed in the Problem

x constant, exponent

T_d spark-out time

w_{ij} the weight of connection between the ith node in one layer and jth node in the next layer

α momentum factor

δ_{pj} error term for pattern p on node j

η learning rate

θ_m^* critical maximum workpiece temperature

κ thermal conductivity

ρ density

Contents

	Page
Abstract	i
Acknowledgements	ii
Nomenclature	iii
Chapter 1 Introduction	1
1.1 The Significance of the Investigation	1
1.2 Aims and Objectives	2
1.3 Scope of the Investigation	3
Chapter 2 Review of Approaches to Selection of Grinding Conditions	5
2.1 Introduction	5
2.2 Data Retrieval Methods	5
2.3 Empirical Model Methods	6
2.4 Artificial Intelligence Methods	8
2.4.1 Grinding wheel selection	10
2.4.2 Grinding parameter selection	12
2.5 Discussion	17
Chapter 3 Study of Grinding Conditions	19
3.1 The Basic Grinding Variables	19
3.2 The Requirements to Process Variables and Output Variables	21
3.3 The Description of the Grinding Problem	22
3.4 The Grinding Conditions	23
3.4.1 The grinding wheel	23
3.4.2 The dressing variables	24
3.4.3 The grinding variables	26
3.5 The Grinding Conditions Selected by Selection System	29
3.6 Conclusions	29

Chapter 4 Strategy and Methodology for Selection of Grinding Conditions 30

4.1 Artificial Intelligence Techniques in Engineering 30

4.1.1 Rule based reasoning 30

4.1.2 Artificial neural network 32

4.1.3 Case based reasoning 33

4.1.4 Comparison of different approaches 34

4.1.5 The hybrid intelligent system 35

4.2 The Strategy for Selection of Grinding Conditions 36

4.3 The Software Approach and Tools for Development of the System 39

Chapter 5 The Prototype Neural Network System 41

5.1 The Multilayer Feedforward Neural Network with the Error Back Propagation Learning Algorithm 41

5.1.1 An artificial neuron 41

5.1.2 The multilayer feedforward network 42

5.1.3 The error back propagation learning algorithm 43

5.2 The Structure of the Prototype 47

5.3 The Executive Modules of the System 48

5.3.1 Training module 48

5.3.2 Test module 50

5.3.3 Application module 51

5.4 The Prototype System Development Environment 52

5.4.1 File editor 53

5.4.2 System setup window 54

5.4.3 Training process monitoring and control 55

5.4.4 Test window 57

5.5 The Prototype System Test and Evaluation 57

5.6 Local Minima Problem of the Back Propagation Algorithm 62

5.7 Summary 62

Chapter 6 Grinding Wheel Selection Using a Neural Network 63

6.1 The Structure of the Wheel Selection System 64

6.1.1 The system input 64

6.1.2 The system output 66

6.1.3 Encoding 66

6.2 The Neural Network Topology Architecture 70

6.3 The Training of the Neural Network 71

6.4 Network Test 74

6.5 Alternative Wheel Selection 76

6.6 The CBN Wheel 77

6.7 The Implementation of the Wheel Selection System 78

6.7.1 The system configuration 78

6.7.2 The system user interface 78

6.8 Application Example 82

6.9 Conclusions 82

Chapter 7 The Selection of Grinding Conditions Using Case Based Reasoning 84

7.1 The Case Based Reasoning Approach 84

7.2 Case Representation 86

7.3 Case Collection 88

7.4 Indexation 90

7.5 Information Retrieval 95

7.5.1 Use of a part number 95

7.5.2 Using the Index 95

7.6 Modification of a Case 98

7.7 Evaluation of the Solution 101

7.8 An Example 103

Chapter 8 The Selection of Grinding Conditions Using Rule Based Reasoning 106

8.1 The Architecture of Rule Based Reasoning for Selection of Grinding Conditions 106

8.2 Rule Base for Grinding Conditions 107

8.3 Inference Engine 112

Chapter 9 The Development of A Multi-Agent System for the Selection of Grinding Conditions 115

9.1 Introduction 115

9.2 The System Structure and Principle 115

9.2.1 The knowledge agents 116

9.2.2 The blackboard 116

9.2.3 The control mechanism 117

9.3 The Implementation of the Multi-agent System 119

9.4 System Input Module 120

9.4.1 Process information input 120

9.4.2 Workpiece information input 121

9.4.3 Workpiece material database 122

9.5 System Select Module 126

9.5.1 Wheel selection 126

9.5.2 Grinding condition selection 126

9.6 Case Database Management 129

9.7 Application Example 130

Chapter 10 The Evaluation of the Strategy and the System 133

10.1 Consideration of Evaluation Techniques 133

10.2 Comparison of the System Output and Data From the Handbooks 135

10.3 Use of the System and Further Development 142

10.4 Discussion 144

Chapter 11 Conclusions 147

Chapter 12 Suggestions for Further Work 149

References 150

Appendices 157

Appendix I The Computer Programmes 157

Appendix II List of the Author's Relevant Published Papers 232

Chapter 1 Introduction

1.1 The Significance of the Investigation

Selecting machining conditions or parameters is a daily task in machine shops throughout the world. F. W. Taylor, realised that the selection of machining parameters was a very important issue. He conducted a large number of machining tests, and developed the well known Taylor's equation which is still in current use [Colding 1992].

Grinding is an important finishing process for many engineering components. There are many parameters in grinding which influence each other. A problem that continues to confront the manufacturing industry is the establishment of efficient grinding conditions. This includes choosing a suitable grinding wheel, establishing the values of grinding parameters such as removal rate and depth of cut and establishing dressing parameters. The grinding conditions must meet the requirements of a specific machining task, characterised by the desired workpiece geometry, by the material, by the machine and by quality, time and cost constraints.

Many investigations have been carried out to establish process models for grinding including physical and empirical models which contribute significantly to understanding of the process[Tönshoff 1992]. However, since physical models cannot be accurately defined and empirical models have a restricted range of validity, process models are not always reliable in practice. To achieve the required quality requirements, operating parameters are often determined with the aid of grinding tests, which are both time-consuming and costly. The process quality and productivity depend to a large extent on the experience of the operator. As a result, many CNC operations are run inefficiently and far from optimum.

Adaptive control with optimisation(ACO) systems have been developed[Malkin

1981][Kelly 1989][Rowe 1991][Xiao 1993]which seek to optimise the grinding process as grinding proceeds. These systems seek to adjust operating parameters in a direction that continually optimises the process according to a predefined performance index. However, some parameters cannot easily be measured on-line, for example surface roughness is usually measured off-line. Other parameters may be adjusted by changes made from part to part. Optimisation may be a lengthy process depending mostly on the initial values of variables selected. Therefore, even with adaptive control the selection of proper grinding conditions is an important issue.

Artificial intelligence (AI) is the branch of science that studies how smart a machine can be and which involves the capability of a machine to perform functions normally associated with human intelligence, such as reasoning, learning and self-improvement [Shoureshi 1993]. Developments in AI have had an increasing impact on manufacturing systems. Almost all areas of manufacturing have been affected by AI.

Ideally, an intelligent machine tool can learn from experience and use the knowledge gained during the learning process to optimise the operation of the machine tool. It appears that precision grinding and other abrasive processes are particularly suited to the application of AI techniques because industrial practice relies heavily on skilled operators to achieve satisfactory results. It is in just such situations that there is the potential to systematise operator skills and knowledge [Rowe 1994a].

1.2 Aims and Objectives

The aim was to investigate the potential for intelligent selection of grinding conditions including the grinding wheel and the values of grinding and dressing parameters. The specific objectives of this investigation were:

(i) To study existing AI methods and evaluate their advantages and disadvantages for selection of grinding conditions.

(ii) To develop a hybrid AI method suitable for selection of grinding conditions.

(iii) To develop a hybrid AI prototype system for intelligent selection of grinding conditions.

1.3 Scope of the Investigation

A review of previous work showed that there was a need for investigation of a new approach for selection of grinding conditions.

Different intelligent approaches were studied and compared so that appropriate techniques could be found for the selection of grinding conditions. Accordingly, a strategy was proposed for selection of grinding conditions.

Methods for building a feedforward(FF) neural network with a back propagation learning algorithm were investigated and a general prototype system developed. This provided a tool for intelligent selection of grinding wheels.

A neural network-based intelligent approach was developed for the selection of appropriate grinding wheels to achieve a specified quality level. A system using the approach was built.

A case-based reasoning (CBR) approach was proposed for the selection of grinding conditions.

A rule-based reasoning approach was proposed to complement the case-based reasoning system for selection of grinding conditions. When case-based reasoning fails to deliver a recommendation, rule-based reasoning is used.

A multi-agent intelligent system was developed for selection of the grinding conditions based on the neural network, case based reasoning and rule based reasoning approaches mentioned. At the current stage, this system is for external plunge grinding with wheels dressed by single point diamond. Related technologies were also investigated. These technologies were object oriented programming and data base technologies.

The system was evaluated by comparison of results with handbooks.

Since each intelligent agent is relatively independent and autonomous, the chapters for describing them are presented separately.

Chapter 2 Review of the Approaches of Selecting Grinding Conditions

2.1 Introduction

Existing techniques employed to deal with the selection of grinding conditions can be classified into the following three categories:

- Data retrieval methods;
- Empirical model methods;
- AI methods

The emphasis in this review is on the application of AI methods.

2.2. Data Retrieval Methods

A handbook is often a logical and effective source of machinability data. Many handbooks are available, such as the widely used Machining Data Handbook [MDC 1980]. However few handbooks cover more than a restricted selection of grinding conditions. An example is illustrated in Table 2.1. According to different materials, the Machining Data Handbook gives the recommended grinding wheels and conditions.

The computerised data retrieval method uses a database of cutting conditions either as suggested in the handbooks or gathered in the industrial field. Cutting conditions for various combinations of material, cutting tool and operation are stored and used. Balakrishana[Balakrishana 1983] surveyed the area of computerised machinability data base systems. Some two dozen systems were identified and their basic features characterised. Unfortunately, the systems developed were applied mainly for turning, drilling and milling, not for grinding.

Although data retrieval methods are simple and practical, they have the following limitations[Balakrishana 1983]:

- The recommendations represent a "starting" set of cutting conditions and hence tend to be conservative in order to cope with worst case machining situations rather than optimal.
- The data apply only to a particular machining situation. The data may not be suitable for slightly different machining situations.

2.3 Empirical Model Methods

A model can be used to predict appropriate conditions for the grinding process. Many investigations have been carried out to establish process models including physical and empirical models [Peters 1984] [Tönshoff 1992]. Due to the fact that the physical interrelationships in grinding cannot be accurately defined, purely physical models are seldom used. Therefore, empirical models are more likely to be used for grinding.

Tönshoff [Tönshoff 1992] reviewed the state-of-the-art in the modelling and simulation of grinding processes. In order to evaluate the models developed by different authors, the models were transformed within the framework of a unified terminology and reduced to basic models. Three different aspects, kinematics, grinding wheel topography and the workpiece characteristics were taken into account. The basic models were expressed in the terms of variables, factors and exponents. Major differences

appear in the absolute values of the derived data. This is possibly due to different specifications of workpiece material and grinding wheel type. It was stated that the complexity of models varies considerably. The question of which model is suitable for a given application could not be answered in general. The decision must take into account the required accuracy. In practice, it might often be sufficient to employ simple and easy-to-handle models.

Based on empirical models and experimental investigation, many methods for selection of grinding conditions were proposed, such as grinding chart methods [Snoeys 1974] [Peters 1976] [Rowe 1987], off-line optimisation methods[Malkin 1980] [Peters 1980].

A limit chart method was described by Rowe[Rowe 1987]. Figure 2-1 shows a limit chart of the centreless grinding process for cast iron where the grinding variables are feedrate v_f and work speed v_w. The process boundaries were formed by available machine power, thermal damage and chatter. The grinding conditions should be within the region enclosed by the boundaries and the maximum removal rate occurred at the junction of the machine power and thermal damage boundaries.

An off-line process optimisation system for cylindrical plunge grinding operations was proposed by Malkin[Malkin 1980]. The objective was to optimise grinding and dressing parameters to maximise the metal removal rate. Removal rate was constrained by the maximum permissible workpiece surface roughness and the need to avoid workpiece thermal damage. The optimisation algorithm was based on the use of process models for workpiece thermal damage, surface texture, grinding power and grinding wheel dressing. The optimal conditions for dressing and grinding were achieved when thermal damage and surface texture constraints were simultaneously achieved. The system was implemented on a PC computer. The user was required to input grinding and dressing conditions, the maximum allowable surface roughness, and the measured grinding power and surface roughness. The system estimated optimal grinding and dressing conditions plus grinding efficiency.

Empirical models are of a limited value. Usually, an empirical model can only be used for the accurate description of one machining application. When employing empirical models for applications with changed boundary conditions, a significantly poorer representation of the grinding process is to be expected.

To compensate for changing boundaries on the limit chart and for the inaccuracy of preset values of parameters, adaptive control with optimisation(ACO) systems provide on-line adjustment of the operating conditions[Malkin 1981][Kelly 1989][Rowe 1991, 1994][Xiao 1992, 1993]. However, some parameter adjustments rely on process models. In addition, as outlined in Chapter 1, the performance of ACO relies on the initial values of operating parameters.

2.4 AI Methods

Advances in knowledge based systems, neural networks, fuzzy logic, and microprocessor technology provide tools for conceptualisation, and development of

intelligent manufacturing systems. For more than a decade there has been a major research activity related to intelligent manufacturing systems. Matsushima and Sata [Matsushima 1980] suggested a hierarchical structure of intelligent machine tool controllers to emulate human operators. The lower levels of the scheme involve off-line adaptive controllers and pattern recognisers. The higher level controls are global in nature, and process data are accumulated over a longer period of time. The results from the higher levels are manifested as changes in the lower level parameters. The conclusion of Matsushima and Sata is that off-line and on-line learning and self-organising techniques are crucial to the development of intelligent machine tools to operate the machines in optimal conditions. Ideally, an intelligent machine tool controller should be able to learn from experience, have self-organising knowledge bases, and be able to use the knowledge obtained to optimise the machining processes in real-time.

A number of investigations have been carried out to apply artificial intelligence in the field of abrasive processes in the following activities [Rowe 1994a]:

- storing and manipulating product information in databases
- storing and manipulating production information in databases
- selecting abrasive tools
- selecting abrasive machining conditions
- controlling abrasive machines
- optimising abrasive process performance
- monitoring grinding process performance
- compensating for grinding machine and process variations.

A summary of applications described in recent papers is given by Rowe, Li, Inasaki and Malkin [Rowe 1994a].

The emphasis in the following review is placed on grinding wheel selection and grinding

and dressing parameter selection.

2.4.1 Grinding wheel selection

Inoue [Inoue 1987], gives a detailed description of the use of rule and frame-based reasoning for grinding. A knowledge based system was capable of diagnosing vibrations in grinding through a system called GSKILL and assisted in the selection of an appropriate grinding wheel through the use of a system called GDMS. The decision support system, GDMS, made use of production rules and hierarchical frames. For example, a frame was employed to associate particular input data (machine tool-$T, workpiece-hardened, workpiece-SCM, wheel-WA, wheel-LMV, wheel-X100) with grinding performance factors (life constant-value, life factor-value, power factor-value, roughness factor-value and force factor-value). This frame AKO was associated with another frame, AKO-F, of implicit feedrate values. Also within GDMS a calculation module calculates grinding wheel life and grinding performance. The calculated data were then checked for constraint violations by a frame called MASTER. These constraint violations were fed to a production rules type knowledge base PKB and used to generate a frame of acceptable wheels where a change of grinding wheel is necessary. The knowledge based system which is programmed in Lisp is linked to a calculation module programmed in FORTRAN.

Midha and Zhu [Midha 1990], Zhu and Midha[Zhu 1992a] developed a rule-based reasoning system for optimum abrasive wheel selection using a proprietary expert system shell XI plus for a variety of grinding operations. The system provided two main applications: superabrasive and conventional wheel selection. Having obtained the necessary input from the user, such as workpiece material, hardness, size and surface condition, surface roughness required etc, the system recommended a suitable wheel based on the standard marking system. The wheel characteristics which were evaluated by the system are grain size, grain-bond-pore percentage, number of grains per unit volume, number of active grains per unit area, wheel bursting speed and wheel bending

strength. The following is an example of the use of rules for wheel selection:

if material group is "Ferrous metals and alloys"

and material type is "Austenitic stainless steels"

then grain type is "C"

and bond type is "V"

and structure number = 6

König [König 1991] described the "Grindex" rule based system to recommend grinding wheels. The system was developed within the TWAICE expert system shell. The knowledge base has three types of rules. The first set of rules analyses the grinding process. The second decides the problem solving path to be employed. The third set classifies and assigns knowledge from the database and procedural elements. The rule base uses production rules to determine a wheel specification. A production rule defines the relationships between an object, its attributes and its values (O-A-V) to other objects, attributes and values.

	Object	Attribute	Value
If	workpiece	material	= bearing steel
and	workpiece	thermal treatment	= hardened
then	wheel	structure	= 6 - 8

Ueda [Ueda 1988] also developed a rule based system for wheel selection written in Lisp. A rule based system was presented by Venk[Venk 1990] for wheel selection.

The disadvantages of many rule/frame based expert systems are that such systems represent a limited body of information which may fail to be updated. If expert systems are not essential for the performance of a production task, the system tends to fall into disuse. The most successful systems are integrated into a production system and are maintained and updated as the technology develops. This is important since if it is

quicker and more reliable for the experienced user to bypass the system, the system no longer plays a useful function.

2.4.2 Grinding parameter selection

Chen [1991] presented a surface grinding process advisory system. The system incorporated both analytical models and heuristic rules, and searches for an optimum solution by using fuzzy logic as an inference mechanism. The system consisted of four modules. The automatic rule generation module generates a set of fuzzy rules from each grinding process model and stores them in an analytical database. The database management module synthesises the generated rules. The generated rules are represented in terms of fuzzy membership functions and synthesis is performed numerically with fuzzy sets. The fuzzy inference module automatically assigns membership grades to the process variables as well as design variables. The optimisation module defuzzifies the design variables from the fuzzy design table and evaluates the objective function with the newly determined values of design variables. If desired conditions have not been reached, the current values of design variables are fed back to the automatic rule generation module and the iteration continues. How to deal with different workpiece materials and grinding wheels was not considered.

Midha, Zhu and Trmal [Midha 1991] and Zhu[Zhu 1992b] made use of knowledge engineering and process modelling for the optimum selection of grinding parameters. Based on the analysis of experimental data for basic grinding parameters, wheel wear and specific energy, process models were developed in such a way that they could be dynamically modified according to user input by a rule based system. The process models accommodate grinding parameters such as specific tangential grinding force F_t', specific energy e_c, maximum temperature T_{max}, wheel wear parameters. The first stage of the selection procedure was to use a rule based system "OPExpert" to give a set of nominal grinding values for a given grinding situation. These can be evaluated in the second stage for burn-free grinding and/or grinding with maximum grinding ratio with a

system "PRExpert". "PRExpert" is a dynamic model and rule based system for prediction of grinding results. A repeated use of the evaluation procedure yielded optimum grinding parameters for a desired criterion, e.g., low grinding force, good surface texture. The main attribute of the proposed approach is that grinding process models can be modified dynamically. The basis of the model may be demonstrated by the formulae used for the computation of specific tangential force F't. The specific tangential force is related to equivalent chip thickness h_{eq} by using a factor F and an exponent f in an equation of the form.

$$F'_t = F.(h_{eq})^f$$

The effect of a wide range of parameters were taken into account by calculating their modifying effect on F and f using factors stored in a database.

$$F = F_o + \sum_{i=1}^{n} F_i \text{ and } f = f_o + \sum_{i=1}^{n} f_i$$

where F_o and f_o are the values for $h_{eq} = 1$. The effect of material grindability is $F_i = \Sigma m_{ij}.a_{ji}$ or in matrix form, $F = M.A_1$. The factors m_{ij} are tabulated according to the material grindability. The selectors a_{ji} are decided by a rule based system according to the input data.

	v. difficult	difficult	middle	easy	v. easy
F_1	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
f_1	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}

The modifications are extracted and summed for each of the parameters of grindability - M, wheel hardness - H, equivalent wheel diameter - D, wheel dressing - W, and wheel grain size - G.

$$\left\{ \begin{array}{c} \sum_{i=1}^{n} F_i \\ \\ \sum_{i=1}^{n} f_i \end{array} \right\} = M*A.. + H*A.. + D*A.. + W*A.. + G*A..$$

Similar procedures were employed for surface roughness and the force ratio. Other parameters are then calculated using appropriate mathematical models. The advantage of the system is that different influencing factors for the material and wheel were taken into account.

Sakakura and Inasaki [Sakakura 1992] described the use of a feedforward neural network used together with a Brain-State-in-a-Box network to selection of dressing conditions illustrated in Figure 2.2.

The network was used to explore the potential of a system which demonstrates associative memory for decision making in grinding. The feedforward network (FFN) is trained using backpropagation. The training is based on results obtained from grinding experiments. The results used for training characterise the surface roughness values in terms of the probability density for each combination of dressing feed and dressing depth. The outputs of FFN may contain various combinations of dressing feed

and dressing depth satisfying a particular surface roughness requirement. The BSB network was used to recall the most suitable combinations. Other factors which affect the dressing conditions were not taken into account by the authors.

Sakakura and Inasaki [Sakakura 1993], designed a fuzzy rule based system for selection of grinding and dressing conditions. The fuzzy rules were based on an analysis of previous grinding results. A learning module evaluates individual data in a grinding data base and generates fuzzy rules. The rules are then stored in a fuzzy rule base. A maximum of 500 grinding examples were stored in the grinding data base. Old sets of practical grinding data are replaced by new values on a first in - first out rule. The learning module is based on genetic algorithms. Genetic algorithms were used to refine the rules in the rule base, create new rules by mutation, crossover and combination. Manipulation of membership functions was facilitated using binary strings. The fitness of a rule is calculated by a function which computes the strength of the match between the rule and the data in the grinding data base. Rules with high fitness are given a greater chance of survival than rules with low fitness. A further factor is introduced into the fitness test which reduces fitness of rules which are closely similar. The system is only suitable to the situation in which the workpiece material and the wheel are unchanged.

Kim and Inasaki 1993 [Kim 1993] also described a fuzzy rule based system to establish optimum grinding conditions, for maximum removal rates, subject to the constraints of grinding power, workpiece burn, chatter vibration, and surface roughness. Specialised knowledge of the grinding operation is acquired from the actual operation database. Coefficients in the experimental equations are obtained through the fuzzy regression model based on fuzzy set theory and are stored in the actual operation database.

Liao and Chen [Liao 1994] described a neural network approach for modelling and optimising a grinding process using creep feed grinding of alumina with a diamond wheel as an example. First, a generalised back propagation neural network with two-

hidden layers is used to establish the process model. The structure of the network was 5×5×4×3, namely, a input layer with 5 neurons, first hidden layer with 5 neurons, second hidden layer with 4 neurons and a output layer with 3 neurons. The five input variables used were bond type B, mesh size m, concentration c, work speed v_w, and depth of cut a. Surface finish R_a, normal grinding force per unit width F_n, and specific grinding power P' were the three output variables. A total of 16 experimental samples are used to train the network. Once the modelling procedure has been implemented, the back propagation algorithm with a Boltzmann factor is used to find the global optimal settings for the grinding process. In the modelling procedure, inputs are fixed and network parameters are adjusted to minimise the error function E. A similar procedure is used for optimisation. The only difference is that, in the optimisation phase, the parameters of the network were fixed and the inputs are adjusted so that the objective function is minimised (or maximised) subject to certain constraints. Sathyanarayanan [Sathyanarayanan 1992] also utilised a neural network approach to model the creep feed grinding of superalloys, but did the optimisation analytically using an off-line multi-objective programming technique. These two systems are limited in application to creed-feed operations.

Rowe [Rowe 1994b] presented a conceptual framework for an intelligent grinding machine as illustrated in Figure 2.3. All of the essential elements of Figure 2.3 were tested for plunge grinding operations. The Intelligent Grinding Machine has the potential to include ACO and the further AI features indicated below:

- to remember optimised conditions for future operations in a learning database
- to provide intelligent selection of process parameters from a learning database
- to integrate into a CIM environment
- to feed process information back to a higher level computer system
- to facilitate set-up.

In addition, some knowledge based approaches were employed to aid the selection of grinding conditions. Venk [Venk 1990] used a frame based system with a PCPlus

expert shell as a qualitative analysis tool (problem formulation) to aid optimisation of the centreless grinding process.

2.5 Discussion

A number of research applications of AI have been described in the literature. However, there is little evidence of the successful implementation of such techniques in standard

production machine tools and systems. However, it is obvious that the trend towards increased use of artificial intelligence in grinding systems and operations is clear, and unlikely to be reversed[Rowe 1994a].

Most of the techniques for selection of grinding conditions described in the literature, rely on process models. Rule based systems can be applied for selection of grinding wheels but it is difficult to cope with every situation for selection of grinding conditions. Neural networks can be used to develop models of the process but rely heavily on the quality and quantity of the training data. A common disadvantage of the systems described in the literature is that system learning is difficult. System learning usually requires development calling on domain experts and knowledge engineers.

Chapter 3 Study of the Grinding Conditions

Grinding is a complex manufacturing process with a large number of interacting variables. Before designing the selection system, it is necessary to define the variables and to find which variables should feature in the selection process. Consideration is therefore given to the relationship and influence between these variables. There are many types of grinding process. However, their action is essentially similar. The study is therefore limited to the external cylindrical plunge grinding operation.

3.1 The Basic Grinding Variables

Figure 3.1 illustrates an external cylindrical plunge grinding system. The subscript w refers to workpiece parameters, and the subscript s is used for wheel parameters.

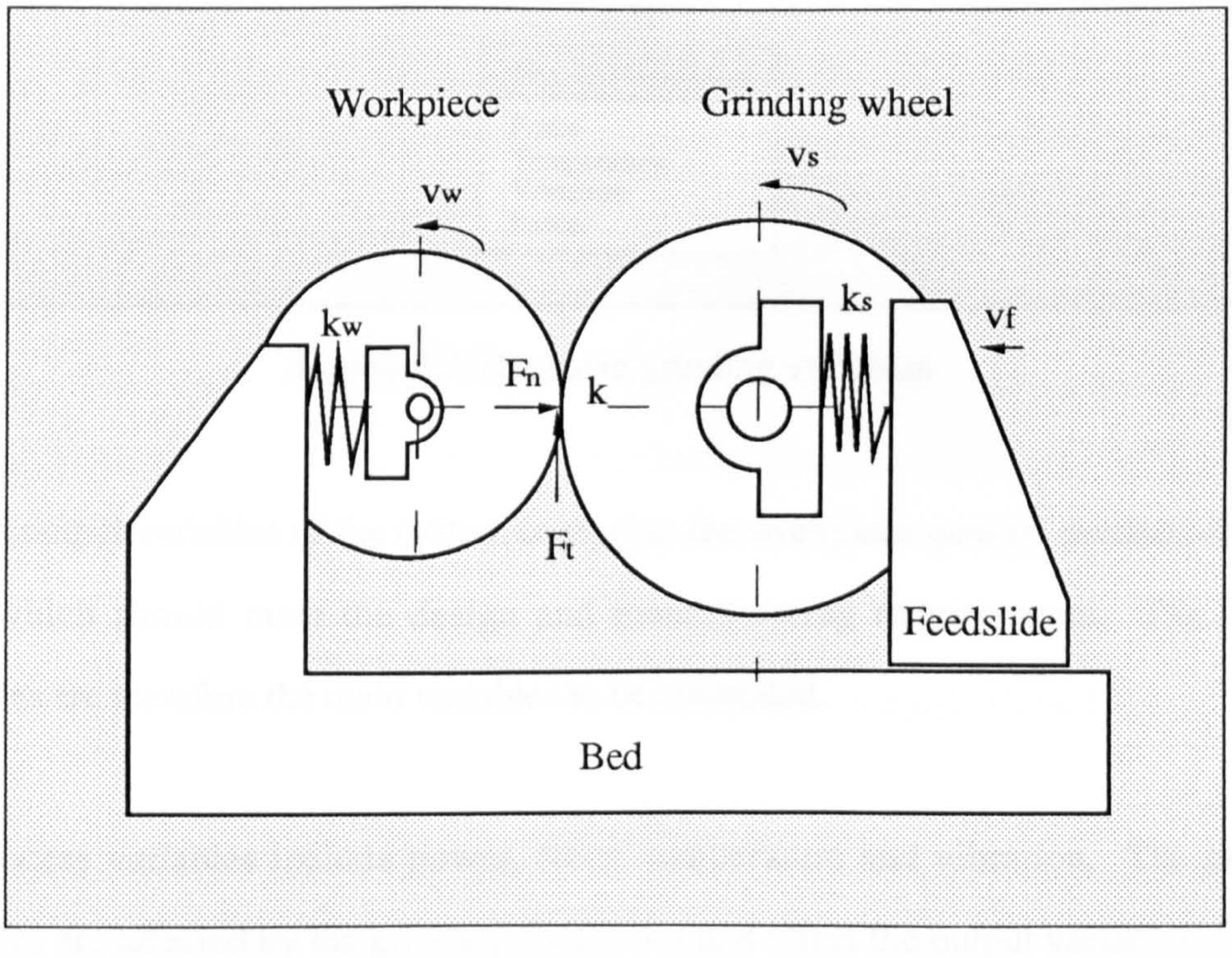

Figure 3.1 External cylindrical plunge grinding system [King 1986]

Material removal during grinding occurs as abrasive grains cut the workpiece surface. The penetration of the cutting points into the material being ground depends on the

topography of the wheel surface and the geometry and kinematic motions of the wheel and workpiece. The basic grinding variables may be divided into four categories as illustrated in Figure 3.2.

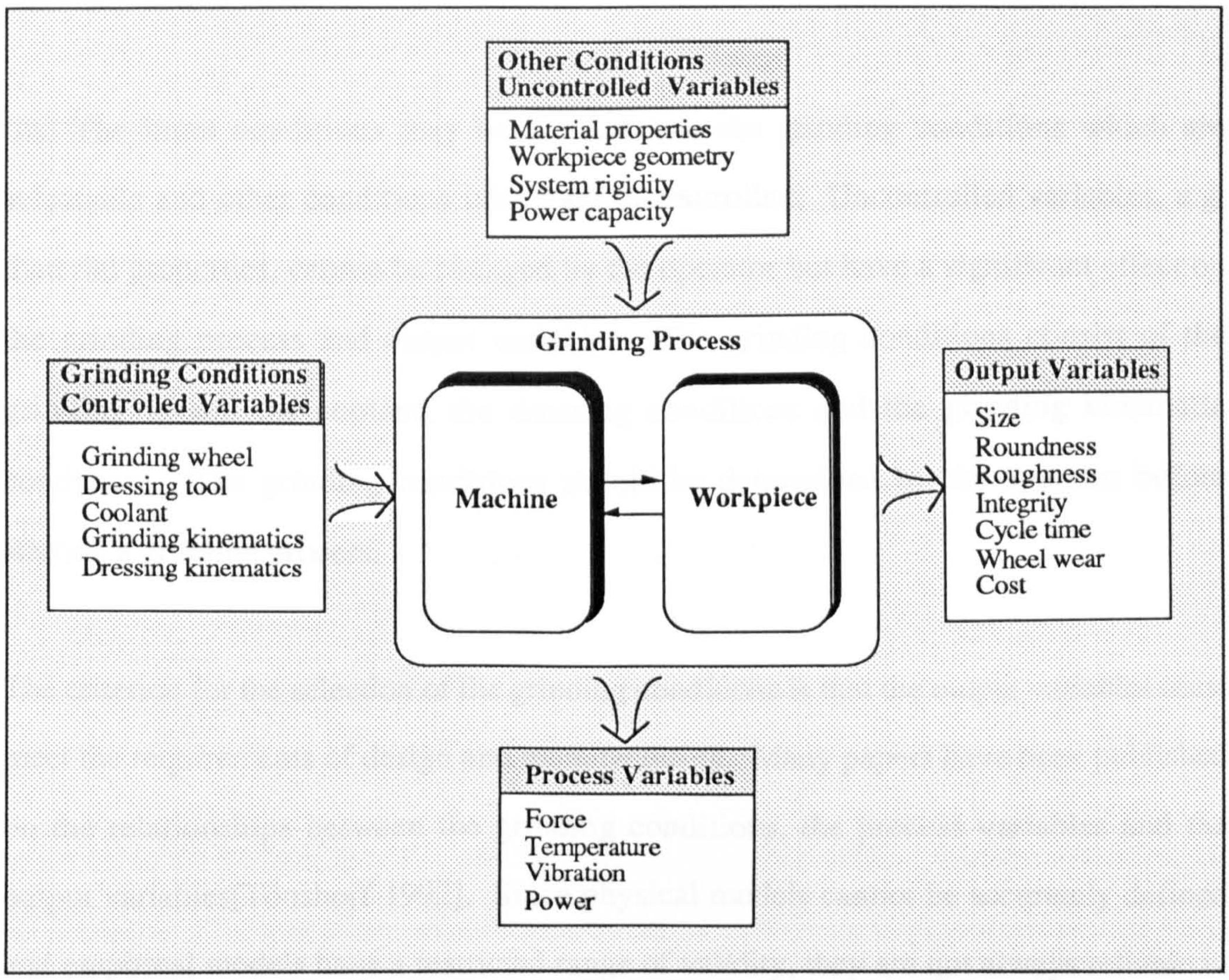

Figure 3.2 The basic grinding variables

(i) The output variables of the system comprise: the workpiece quality, productivity and cost, which should meet the design and manufacturing requirements. The output variables are therefore the main variables to be controlled.

(ii) Process variables include power, force, temperature and vibration. The process variables are affected by the grinding conditions and affect the output variables. As the grinding wheel engages the workpiece, forces are induced between the wheel and the workpiece. The force depends on the grinding conditions and affects the output variables and the other process variables. The higher the forces are, the faster the material removal rate. The force also affects the surface roughness, the deflection of the

system and the onset of thermal damage. For this reason, some adaptive control systems use controlled force techniques[Tönshoff, 1986]. However, process variables are intermediate variables which provide indirect evidence of the relationships between input and output variables.

(iii) The input conditions may be divided into the grinding conditions which are selectable and other conditions which are uncontrolled. Uncontrolled variables, e.g. material properties, cannot be changed by the operator but have a significant effect on the grinding process and output variables. The grinding conditions consist of the grinding wheel, the coolant, the dressing conditions and the grinding kinematic conditions. The grinding conditions should be determined by the operator before starting a grinding process.

The criterion for the selection of the grinding conditions is that the output variables must meet the requirements of design and manufacturing. Many papers have been published on the relationships between the grinding conditions, the process variables and the output variables[Tönshoff 1992]. Since physical models cannot be accurately defined and empirical models have a restricted range of validity, they are not always reliable in practice. But, these grinding models demonstrate qualitative relationships of the grinding process. Thus, in this chapter, a qualitative analysis of the main parameters is described. These basic parameters will be employed in later chapters.

3.2 The Requirements to Process Variables and Output Variables

The overall objective is to select appropriate grinding conditions which are as near to optimal as possible. The optimisation objective to be considered here is minimisation of the grinding time subject to quality constraints. The optimisation problem therefore can be summarised as to minimise cycle time T subject to the following constraints:

$e_c \leq e_c^*$ (burning constraint)

$R_a \leq R_a^*$ (roughness constraint)

$r \leq r^*$ (out-of-roundness constraint)

$s \leq s^*$ (size constraint)

$P \leq P^*$ (power constraint) 3.1

where e_c is the specific grinding energy, R_a is the workpiece surface roughness, r is the workpiece roundness, s is the workpiece size error and P is the grinding power, the asterisk signifies a constraint value. In the grinding process, some output variables, such as surface integrity, are not conveniently measured and controlled. A process variable which can be reliably measured and controlled is power[Kelly 1989]. The power is therefore controlled to prevent thermal damage by reference to a thermal model. In 1988 Rowe[Rowe 1988] presented a thermal model:

$$e_c^* = \frac{1}{C}(\kappa\rho c)_w^{1/2}\left(\frac{l_e}{v_w}\right)^{1/2}\frac{\theta_m^*}{R_w a} \qquad 3.2$$

where e_c^* is the critical specific grinding energy, θ_m^* is the critical maximum workpiece temperature, C is a heat flux distribution factor, κ is the thermal conductivity, ρ is the density, c is the specific heat capacity, l_e is the real contact length, v_w is the workpiece speed, R_w is the fraction of the total energy partitioned to the workpiece and a is the true depth of cut.

Here wheel wear and cost are not taken into account as independent variables.

3.3 The Description of the Grinding Problem

The selection of the grinding conditions is the solution to the grinding problem. Therefore, the operator should present the description of the grinding problem to the selection system. There are two major components of the problem description, goals to be achieved in solving the problem and constraints on these goals. The goals in the

grinding problem are the machining requirements. The constraints are the nature of the workpiece and the machine tool. In other words, the description of the grinding problem consists of the requirements to output variables and the uncontrolled variables which cannot be changed by the operator.

3.4 The Grinding Conditions

3.4.1 The grinding wheel

The grinding wheel characteristics have a direct effect on process efficiency, accuracy, surface roughness and surface integrity[Nakajima 1978]. It is therefore essential to select an appropriate grinding wheel. The best wheel for an application is a compromise between the ability to cut rapidly and the ability to hold form, maintain the surface roughness requirements and last a long time before dressing is required[MDC 1980]. The specification of the grinding wheel consists of six parts:

- Abrasive type
- Abrasive grit size
- Grade
- Structure
- Bond

However, since the wheel structure part is *use optional* and it is not shown on the products, the wheel structure is omitted in the selection of a wheel.

A comprehensive model is not yet available that can relate the wheel specification and the output variables. However, the main factors affecting grinding wheel performance are as follows[King 1986][MDC 1980][Universal 1992]:

(i) The type and hardness of material to be ground affect the selection of abrasive, grit size and grade. Alumina is the most efficient conventional abrasive for grinding high

tensile materials such as steel and cast iron. Silicon carbides abrasives are used to grind low tensile strength materials and non-metallic materials. CBN grinding wheels are recommended for alloys that are difficult to grind with conventional abrasives. The wheel grade must be adjusted to suit the hardness of the materials. The harder the material, the softer the grade of wheel required.

(ii) The surface roughness produced is affected by the abrasive grit size and bond type. High stock removal rates require coarse grit wheels. Small grit sizes are used to achieve fine finishes and close tolerances on finished workpiece geometry. Resinoid, rubber or shellac bonded wheels are used to achieve the finest finishes.

(iii) The selection of wheel grade and structure depend on the contact area between the grinding wheel and the workpiece. Large contact areas tend to produce low grinding pressures and require soft grade, open structure wheels. Conversely, small contact areas require harder grade and closer structure wheels. The size of the workpiece also affects the contact area. In external grinding, the larger the workpiece, the larger the contact area, requiring softer grade wheels.

(iv) The severity of the grinding operation affects the choice of grit size and grade. For example, a rough cast or forged workpiece requires a harder grade and coarser grit size.

3.4.2 The dressing variables

Dressing prepares the cutting surface of the wheel by removing dulled grains or by cutting through them to present new sharp grains. Dressing also removes wheel loading and particles of connecting bond material to open up the porosity of the wheel surface. Dressing conditions have an important influence on the following output and process variables[Verkerk 1979]:

- Grinding force
- Wheel wear

- Workpiece surface roughness
- Specific energy
- Surface integrity

There are several dressing methods. However the single-point diamond dressing tool is commonly employed in industry. The dressing variables are:

- Dressing lead f_d
- Dressing depth a_d
- Number of passes n_d
- Setting angle of the diamond
- Initial diamond shape
- Diamond wear

In these variables, diamond wear is uncontrollable. Angle and initial shape of the diamond have fixed values. The number of dressing passes n_d usually is not more than 4[Pattinson 1975]. Therefore, in the selection of the dressing parameters, the main variables to be considered are dressing lead f_d and dressing depth a_d. Results from many investigations [Verkerk 1979] show that coarse wheel dressing (high dressing lead and large dressing depth) produces an open structure which results in good cutting efficiency and lower grinding force but poor workpiece finish, whereas fine dressing produces a more closed structure to the wheel face which results in good workpiece surface finish but inferior cutting properties.

Malkin in 1989[Malkin 1989] gave a relationship between dressing parameters and initial workpiece surface roughness :

$$R_a = R_1 f_d^{1/2} a_d^{1/4} \left(\frac{Q_w'}{v_s} \right)^x \qquad 3.3$$

where R_1 and x are empirical constants. The exponent x lies typically in the range 0.15 -

0.6. Equation 3.1 applies for short grinding cycles with frequent wheel redressing. In such cases, the wheel may be dressed once per part, usually prior to the final finishing stage of the grinding cycle. With longer grinding cycles and less frequent dressing, wear of the grinding wheel alters its topography and the surface roughness changes with time[Malkin 1989].

As mentioned in section 3.2, abrasive grit size can significantly affect the surface roughness. For conventional abrasive wheels dressed prior to use, the grit size is usually found to have only a minor influence on the initial workpiece roughness after dressing. However, with continued grinding after the initial effects of dressing are removed, abrasive grit size more strongly affects the surface roughness[Malkin 1989]. The best approach should consider the effects of the dressing and the wheel grit size. Selection of the appropriate values of the dressing variables and wheel grit size can make the grinding behaviour more stable for the period of grinding before wheel re-dressing. A dressing strategy has been proposed to obtain a stable grinding wheel working surface[Rowe 1995].

The process of selecting the values of dressing parameters is made more difficult due to the uncertainty introduced by the shape of the dressing diamond which will change with the wear. The process is much simpler when a multipoint dressing tool is employed [Verkerk 1979]. However, the process outlined is considered to be the most practicable approach for the present state of knowledge.

3.4.3 The grinding variables

The controlled grinding variables include the feed and speeds. The kinematic variables affect the process and output variables. The main controlled grinding variables are:

- Wheel speed v_s
- Work speed v_w
- Feedrate v_f

• Spark-out time T_d

Many papers have been published on the relationships between the controlled grinding variables, the process variables and the output variables[Tönshoff 1992]. In practice, one of the most important and reliable basic parameters is the equivalent chip thickness h_{eq} which correlates fairly well with the main grinding parameters, as shown by many experiments[Peters 1976]. The equivalent chip thickness is defined as:

$$h_{eq} = \frac{v_w a}{v_s} = \frac{\dot{Q}_w}{v_s} = \frac{\pi d_w v_f}{v_s} \qquad 3.4$$

The relationship between h_{eq} and specific normal grinding force F_n', specific tangential grinding force F_t', specific energy e_c, roughness R_a and the G ratio may be fairly well approximated by power function[Peters 1976]:

$$F_t' = F_1 h_{eq}^f \qquad 3.5$$

$$F_n' = F_2 h_{eq}^f \qquad 3.6$$

$$e_c = F_1 h_{eq}^{f-1} \qquad 3.7$$

$$R_a = R_1 h_{eq}^x \qquad 3.8$$

$$G = G_1 h_{eq}^{-g} \qquad 3.9$$

where F_1, F_2, R_1 and G_1 are empirical constants, and the f, x and g are exponents. The factors and exponents depend on the nature of the workpiece material and the grinding wheel.

Since h_{eq} takes no account of wheel grit spacing or workspeed, Rowe [Rowe 1987]

proposed two basic parameters for describing the kinematic relationship between grinding parameters and applied them in an ACO system. The two parameters were mean grinding chip volume V_m and the mean chip aspect ratio A_r. The two parameters are defined as:

$$v_m = h_{eq} l_d b_c = \frac{\pi d_w l_d b_c v_f}{v_s} \qquad 3.10$$

$$A_r = \frac{l_c}{h_m} = \frac{0.5 d_e v_s}{l_d v_w} \qquad 3.11$$

where l_d is the dynamic grit spacing, b_c is the grinding chip width, d_w is the work diameter, l_c is the chip length, h_m is the maximum chip thickness and d_e is the equivalent diameter.

Another basic parameter based on the kinematics of the grinding process is the ratio between the equivalent chip thickness and the geometric contact length h_{eq}/lg [Brinksmeier 1993]. The parameter h_{eq}/lg includes the effects of workspeed but takes no account of the wheel grit spacing. It was argued that h_{eq}/lg correlates fairly well with the main grinding parameters and can be used to relate optimal grinding conditions from one operation to another.

In these basic models, spark-out time is not included. However, spark-out time will affect the roundness and the roughness of the workpiece. The spark-out period may be determined in the grinding process because the spark-out period required is strongly dependent on the deflection of the system. The deflection depends on the stiffness of the workpiece, the stiffness of the machine and the efficiency of the removal process. When an appropriate adaptive control system is available, spark-out period can be automatically adjusted[Rowe 1991].

3.5 The Grinding Conditions Selected by Selection System

From the above sections, it can be seen that not all grinding conditions can or need be selected. Some conditions are uncontrolled and some have fixed or optimal values. In addition, some values are better determined during the grinding process. Therefore, the independent grinding conditions to be selected are:

- Abrasive type
- Grain size
- Wheel grade
- Bond
- Dressing lead f_d
- Dressing depth a_d
- Wheel speed v_s
- Work speed v_w
- Feedrate v_f
- Coolant

3.6 Conclusions

There is a large number of variables in grinding each of which influence each other. Basic control parameters have been proposed which correlate fairly well with the main grinding control variables [Peters 1976] [Rowe 1987]. However, a comprehensive model which can widely and reliably relate all the variables of the grinding processes is not yet available.

Chapter 4 Strategy and Methodology for Selection of Grinding Conditions

The purpose of this chapter is to describe the basic concepts and features of artificial intelligence techniques and propose a strategy for selection of grinding conditions with suitable intelligent techniques.

4.1 Artificial Intelligence Techniques in Engineering

A commonly accepted definition of Artificial Intelligence is that "Artificial intelligence is the subfield of computer science concerned with the use of computers in tasks that are normally considered to require knowledge, perception, reasoning, learning, understanding and similar cognitive abilities" [Duda 1979]. AI methodologies applied in engineering mainly include the following approaches:

- Rule or frame based reasoning
- Case based reasoning
- Artificial neural networks
- Fuzzy logic
- Genetic algorithms
- Hybrid methods

The principal categories are rule/frame based reasoning, case based reasoning and artificial neural networks. Fuzzy logic is usually associated with rule based systems or neural networks. Genetic algorithms are usually employed as an optimisation or learning technique.

4.1.1 Rule based reasoning

The rule based reasoning system is a kind of knowledge based system (KBS) employing production rules as the main form of knowledge representation and

manipulation. A rule is a conditional statement that specifies an action that is supposed to take place under a certain set of conditions. The set of rules is usually in the form of IF<some condition is met> THEN<execute some action> [Adeli 1990]. It is useful to structure problem-solving systems in a way that facilitates a description of a search process. Many existing expert systems employ rule based reasoning.

Rule based reasoning has the following generally positive features [Dagli 1994]:

- Ease of exploring the knowledge base, i.e., the encoding of information in readable form.
- Ease of modification of the knowledge base, i.e., a rule may be added or removed.
- Flexibility of processing, i.e., the Inference mechanism may be chosen to suit the problem.

In grinding, it is difficult to build a formal relationship between the grinding wheel specification and the grinding behaviour. Almost all existing intelligent systems[Inoue 1987] [Venk 1990] [Zhu 1992a] for selection of grinding wheels employ rule based reasoning.

Rule based reasoning also has drawbacks. It is time consuming, both for the system developer and the contributing expert, to extract and encode a collection of rules into a coherent knowledge base. For example, Zhou's system[Zhu 1992b] for wheel selection contained over 2,000 rules. For poorly understood domains, for example, selection of the values of dressing and grinding parameters, it is difficult to code the imprecise knowledge involved in rules. In such cases, decision makers rely heavily on their experience rather than on explicitly stated rules. In existing intelligent systems, the selection of the values of dressing and grinding parameters mostly rely on process models. The rule based system mainly play the role of an assistant. The most serious limitation of rule based systems is an inability to learn from operating experience and the inability to take account of developing technology. In practice, an intelligent system may have only a small volume of initial knowledge and machine learning is therefore

important if relevant operating experience is to be accumulated.

4.1.2 Artificial neural network (ANN)

Artificial neural networks(ANNs) constitute a new approach to computation. As formulated by Kohonen[Kohonen 1988] " artificial neural networks are massively parallel interconnected networks of simple, usually adaptive elements and their hierarchical organisations which are intended to interact with the objects of the real world in the same way as biological nervous systems do". ANNs are useful for classification, autoassociation, time-series prediction and function approximation.

Neural networks have the following main advantages comparing with rule based reasoning:

• knowledge is obtained by learning from examples. Therefore an ANN can be used in problems with poorly understood domains, especially for modelling multivariable, non-linear systems. This feature is particularly relevant to the grinding process.

• Knowledge is stored in a distributed fashion. The knowledge is not stored by address or in a particular neuron of the network. Instead, each item of knowledge is stored over all neurons, and each neuron contributes to representing many pieces of knowledge. Generally, distributed schemes require less memory for storing knowledge and are naturally fault tolerant.

• Neural networks can provide solutions to problem which have not previously been experienced as long as the solutions lie within the same domain as the training data.

These characteristics make neural networks potentially useful for modelling grinding processes[Sathyanarayanan 1992][Liao 1994]. The neural network can model the grinding process as a black box and there is no need to fully understand the grinding process.

The main disadvantage of ANNs for grinding is the difficulty of finding sufficient

reliable training data to cover the whole domain of interest. Training data are obtained mainly through grinding experiments although production data could also be employed[Sathyanarayanan 1992][Liao 1994]. For a workpiece - wheel combination, a group of training data can be used to train an ANN. However, when the workpiece-wheel combination changes, the neural network needs to be retrained based on relevant information. The training of an ANN is time consuming and is not guaranteed to provide a reliable results. Neural networks are therefore suitable for a particular operation where training data are available. It is difficult therefore to develop an ANN which can model the grinding process comprehensively.

To date, many kinds of neural network architecture have been developed. Popular neural networks include the multi-layer feedforward network, adaptive resonance theory (ART) models and Hopfield models [Monostori 1992]. However, for modelling, the network most commonly used is the multi-layer feedforward network incorporating an error feed back propagation learning algorithm. The multi-layer feedforward network has proved to be capable of approximating any non-linear function with arbitrary accuracy, Hornik, Stinchcombe and White [Hornik 1989].

4.1.3 Case based reasoning

Case-Based Reasoning(CBR) is an approach which seeks to identify a close match between a new operation to be performed and the characteristics of a previously successful case stored in a case base. The approach solving a problem is to remember a similar problem solved in the past and adapt the solution to solve the new problem. An early description of the concept of case based reasoning was given in *Dynamic Memory* [Schank 1982]. The approach is close to the human decision-making process[Yoon 1993]. In this process, the key point is usage of experience instead of rules. This is a major difference between skilled operators and novices. A novice solving an operating problem usually uses rules but this is not always the best way to solve a problem, particularly when the problem lies outside the domain of the available rules.

CBR has several advantages [Ketler 1993].

• CBR can be used in problems with poorly understood domains . It does not require understanding of why a previous solution was successful. Case based systems can often avoid some of the difficult representational and behavioural modelling needed when using a rule based approach.

• CBR automates the process of incorporating new knowledge into an existing knowledge base. A CBR system automatically utilises this additional knowledge for solution of future problems.

• The knowledge acquisition bottleneck is much easier with CBR than with other learning methods. The reason for this is that the classification model used for information retrieval provides the basis for storage of new information. Cases unlike rules require a minimum of debugging of the interactions between them. Thus, initial knowledge acquisition can be routine.

CBR technology has received much recent attention [Barletta 1991] [O'connor 1992] [Mott 1993] [Tsatsoulis 1993][Watson 1995] but is still a relatively young area within AI and there are still many problems that need to be researched. In grinding, it is difficult if not impossible to obtain enough cases to cover the whole problem space in the initial stage when the system is set up. CBR may fail to give a solution where the number of cases is insufficient.

4.1.4 Comparison of different approaches

Each of the above described methodologies has their advantages and disadvantages. A comparison of the main features of the different methodologies is given in Table 4.1. From Table 4.1, it can be seen that case based reasoning has the best overall performance.

Table 4.1 The main features of intelligent technologies

Subject	*RBR*	*ANN*	*CBR*
knowledge acquisition	difficult	easy	easy
Learning	poor	good	very good
Development time	low	fast	fast
Maintenance	difficult in large system	easy	easy
Explanation	fair	none	good
Major source of knowledge	expert	database, expert	casebase, expert

4.1.5 The hybrid intelligent system

Real world grinding problems are varied and complex. A grinding application may neither fit the assumptions of a single technology nor be effectively solved by the strengths and capabilities of a single technology. One approach to deal with complex real world problems is to integrate the use of several other technologies in order to combine their different strengths and overcome a single technology's weakness to generate hybrid solutions[Dagli 1994]. Hybrid systems can be developed in a variety of ways. For example, one technology may embed another or several technologies to complete a task in an integrated approach. The 'blackboard' architecture [Engelmore 1988] is a hybrid intelligent framework having the greatest potential for complex problem-solving.

A blackboard model is a highly structured, opportunistic problem-solving model that prescribes the organisation of knowledge and data and the problem-solving behaviour within the overall organisation[Engelmore 1988]. The basic principle of the model can be described using the following example.

The system operates in a similar manner to a group of experts collaborating to solve a

complex problem. The experts have at their disposal their collective expertise and a large blackboard. Each expert is a specialist whose knowledge may be relevant at some point in the problem-solving process. The experts agree to maintain a record of their current best partial solution(s) to the problem on the blackboard for all to see. Anyone able to contribute to the current partial solution writes their contribution on the blackboard. The information on the blackboard may also be modified or deleted by experts during problem solving. Each expert watches the blackboard, looking for opportunities to contribute to the solution which arises in the course of problem solving as the combination of items on the blackboard fits their particular specialist expertise [Mirzai 1990].

The blackboard is a conceptual, and not a computational, framework[Dagli 1994]. It allows many different extensions and variations to this prescriptive model. The application itself and the various knowledge agents available to build the system will determine the final implemented form of the blackboard. The concept embraces the notion of problem decomposition with different agents attacking areas of the problem to which agents are most suited. This allows for very flexible knowledge application and the utilisation of multiple inferencing techniques and approaches[Occello 1994] [Chevrier 1994] [Mani 1994][Botti 1995].

4.2 The Strategy for the Selection of Grinding Conditions

Based on the above description, it is expected that a system can be achieved which can select grinding conditions using existing grinding knowledge. The store of knowledge should improve and increase continually through operation. The system should be integrated into a production system and be maintained and updated as the technology develops. This is important since if it is quicker and more reliable for the experienced user to bypass the system, the system no longer plays a useful function[Rowe 1996].

The case based reasoning approach was adopted for the selection of grinding conditions

because the process is affected by a large number of factors, the effects of which are poorly defined. Using grinding experience as the main source of system knowledge, rather than explicit rules, the development of the knowledge base is easier and faster. The most important factor in using case based reasoning is that technology can be relatively easily updated by incorporating new cases into the case base. The operator can add the new cases into the case base without the assistance of the knowledge engineer. In addition, CBR avoids the drawback with neural networks that training is time consuming and not guaranteed. However, it is difficult to find enough grinding cases to cover a sufficiently large problem coverage in the initial stage of development. Case based reasoning may therefore fail to provide a solution if the number of cases is insufficient. Therefore, other techniques need to be associated with case based reasoning.

Generalisation of rule based systems for a wide range of grinding operations may prove to be intractable. So far, since deep knowledge of the grinding process is lacking, it is impossible to build a comprehensive rule base to meet the requirements of most situations. However, in the initial stage of case based reasoning when cases are insufficient to cover the whole of the problem space, rule based systems can be used to suggest a starting point for decision making in grinding.

A feedforward network with backpropagation learning was adopted for the grinding wheel selection system. A neural network approach can be developed as an independent system for selection of the grinding wheel but also used to complement the case based reasoning. Catalogue data were used to provide training data.

In some situations, the operator may have to make decisions. For example, if the values of operating parameters recommended by the system are considered to be unsuitable for any reason, the operator should provide alternative values. For example, the operator may decide to use the existing wheel on a machine to save time.

A multi-agent approach is proposed for the hybrid system. Agents are capable of acting independently, cooperatively and collaboratively to achieve a collective goal. The agents interact through a blackboard model. The system is illustrated in Figure 4.1.

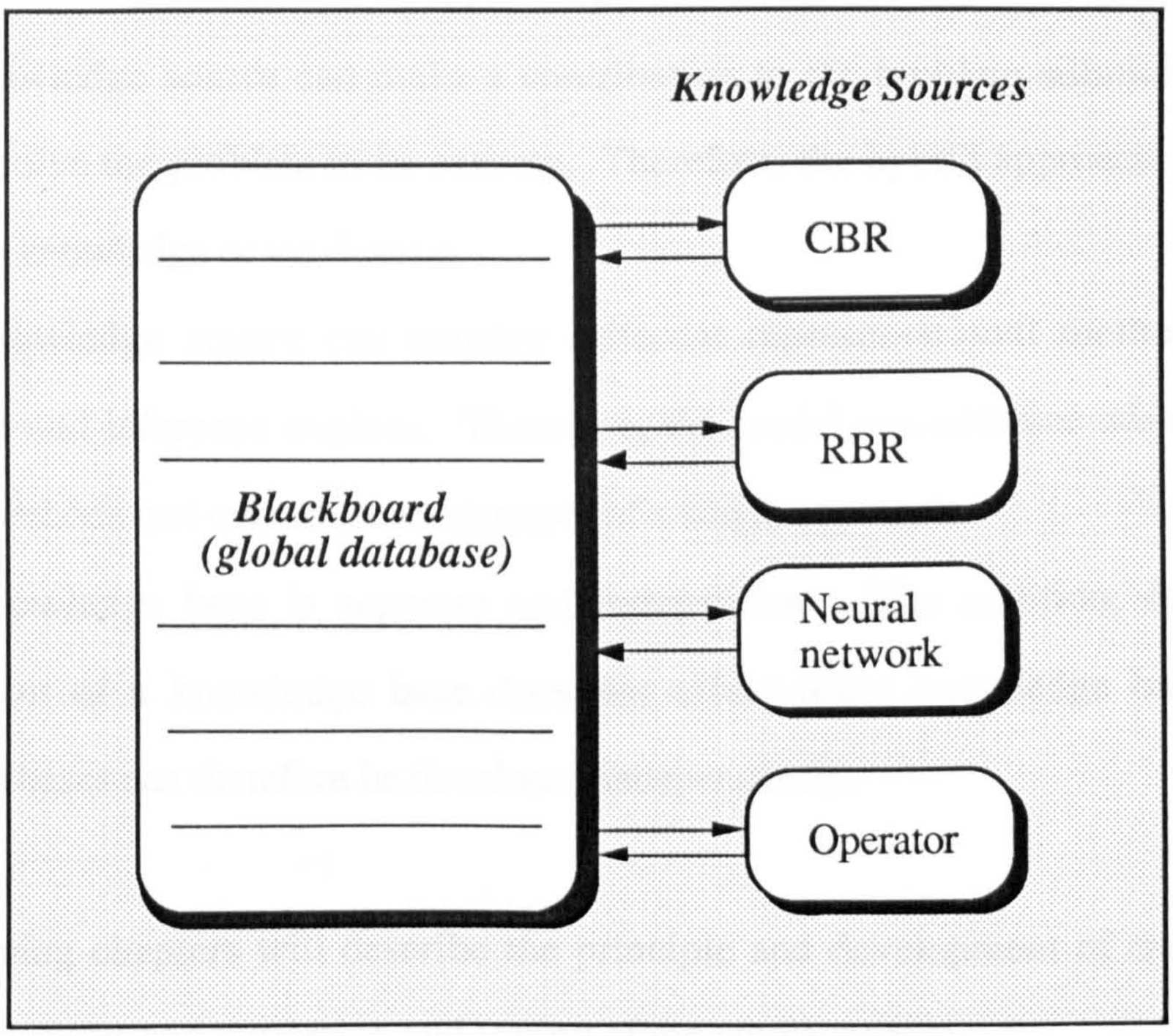

Figure 4.1 A blackboard model for grinding operation

The proposed system has four knowledge agents:

(i) Case based reasoning is employed as the main problem-solving part which can select combinations of the grinding wheel and values of control parameters.

(ii) Rule based reasoning is employed to select values of the control parameters, where relevant data are unavailable in the case base.

(iii) A neural network is employed to select a grinding wheel if required.

(iv) The operator can make a final decision about the wheel or the values of control parameters.

Each knowledge agent can make a contribution to problem solution. The priority of the agents in decision-making is :

1. Operator

2. Case based reasoning

3. Rule based reasoning and the neural network

The main advantages of the hybrid approach proposed are:

• Each knowledge source can make a contribution to the problem solution, without having to solve the problem in its entirety. Therefore, the hybrid approach allows for incomplete knowledge of the domain.

• Each knowledge source can employ different representational methods of the knowledge and inference engines. Therefore, the model can combine advantages of different methods and overcome weaknesses of a single approach.

• Each knowledge base is separate and independent. The addition, deletion or modification of a knowledge base does not affect other knowledge bases. The knowledge bases can therefore be developed independently.

The following chapters will describe the principle and development of the different knowledge agents and the design and testing of the whole system.

4.3 The Software Approach and Tools for Development of the System

Three independent systems were developed.

• A prototype neural network system

• A wheel selection system using a neural network

• A multi-agent system for selection of grinding conditions

The systems were developed in the DOS character(text)-based, rather than in a graphical environment, which means the systems run faster. At the time of development, Dos was considered to be more suitable for an industrial environment. The system was developed using an object-oriented programming approach employing the Borland C++ programming language.

The object-oriented programming paradigm is a software design and development technology. The technology incorporates several sophisticated and efficient mechanisms that provide an organisational framework for the development of a large and complex software project. Much of the value of object-oriented programming results from the feature of inheritance. The idea of inheritance is that a programmer starts with a library of already-developed object types, or classes, and uses the object classes for new applications by adding data elements or operations to form new classes.

Inheritance is a particularly useful feature for designing user interfaces. Developing a user-friendly interface may take much time and effort. The user interfaces of the systems were developed using the Borland Turbo Vision package which is included in Borland versions C++ 3.0 or 3.1. Turbo Vision is an object-oriented, event-driven environment. The Turbo Vision Package includes many classes which can be inherited for developing user interfaces.

Chapter 5 The Prototype Neural Network System

The development of the neural network application required a development tool. There are a number of commercial neural network development tools available. However, these tools have some limitations for practical problems.

(1) It may be inconvenient to transfer the trained neural network into the application system. For example, the NeuralDesk [NeuralDesk 1992] package only gives a textual report of the network construction and values of the weights.

(2) It may not be possible to integrate a commercial development package into an industrial application system because of the size, cost and copyright problems of the package. If an executable application is employed rather than the complete package, re-training will be difficult in an industrial application.

A prototype neural network system was therefore developed and embedded in an application system. This chapter describes the principle and the structure of the prototype.

5.1 Multilayer Feedforward Network with the Error Back Propagation Learning Algorithm

5.1.1 An artificial neuron

A single artificial neuron is an information-processing unit that is fundamental to the operation of a neural network. Figure 5.1 shows the model for a neuron used in the prototype. The inputs are attenuated by the neuron weights, w_i, and summed to produce the value *net*. The value *net* is operated on by an activation function to produce an output. The activation function used was the sigmoidal function.

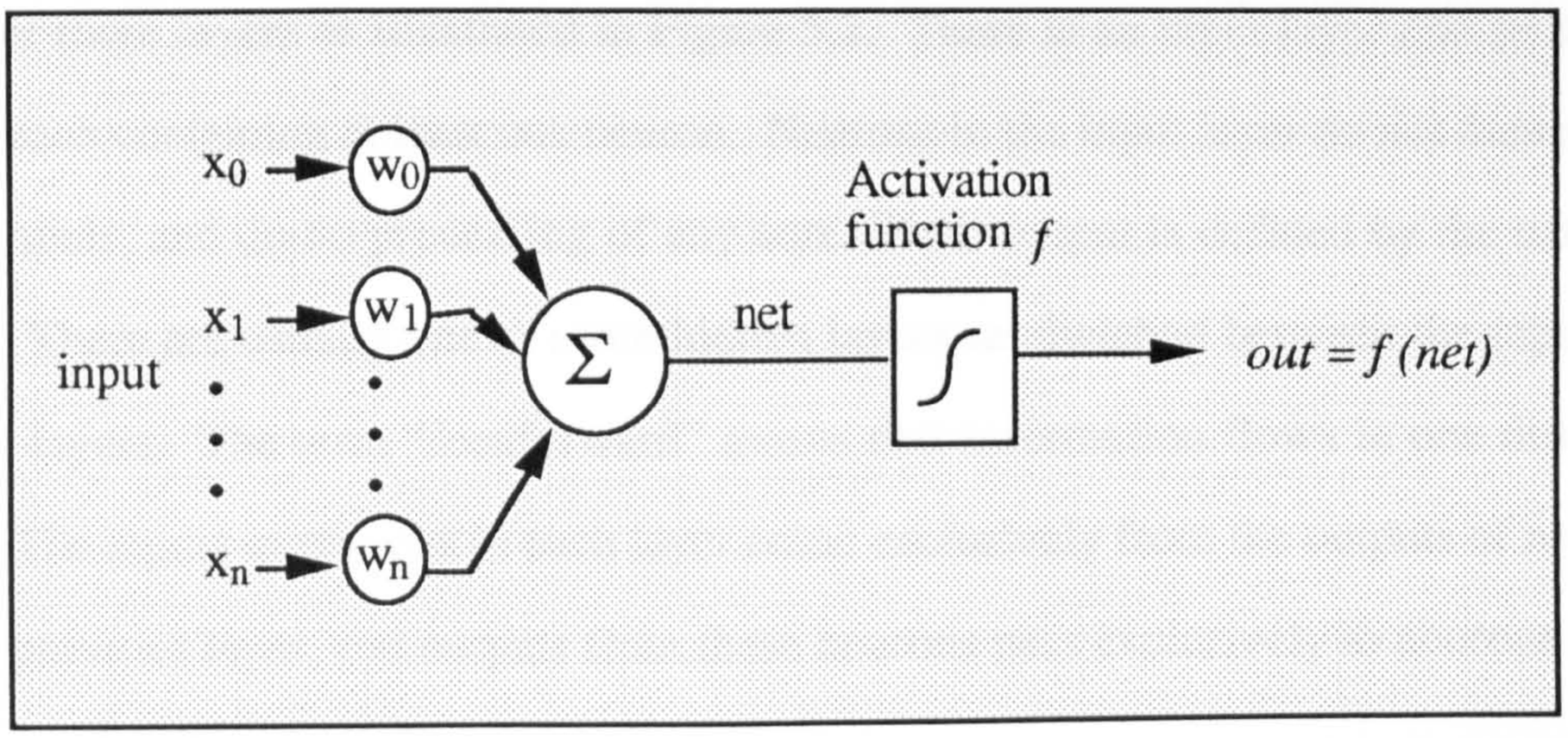

Fig. 5.1 An artificial neuron

The operation of the neuron is described by the equations

$$net = \sum_{i=0}^{n} w_i \, x_i \qquad 5.1$$

$$out = f(net) = \frac{1}{1 + e^{-net}} \qquad 5.2$$

5.1.2 The multilayer feedforward network

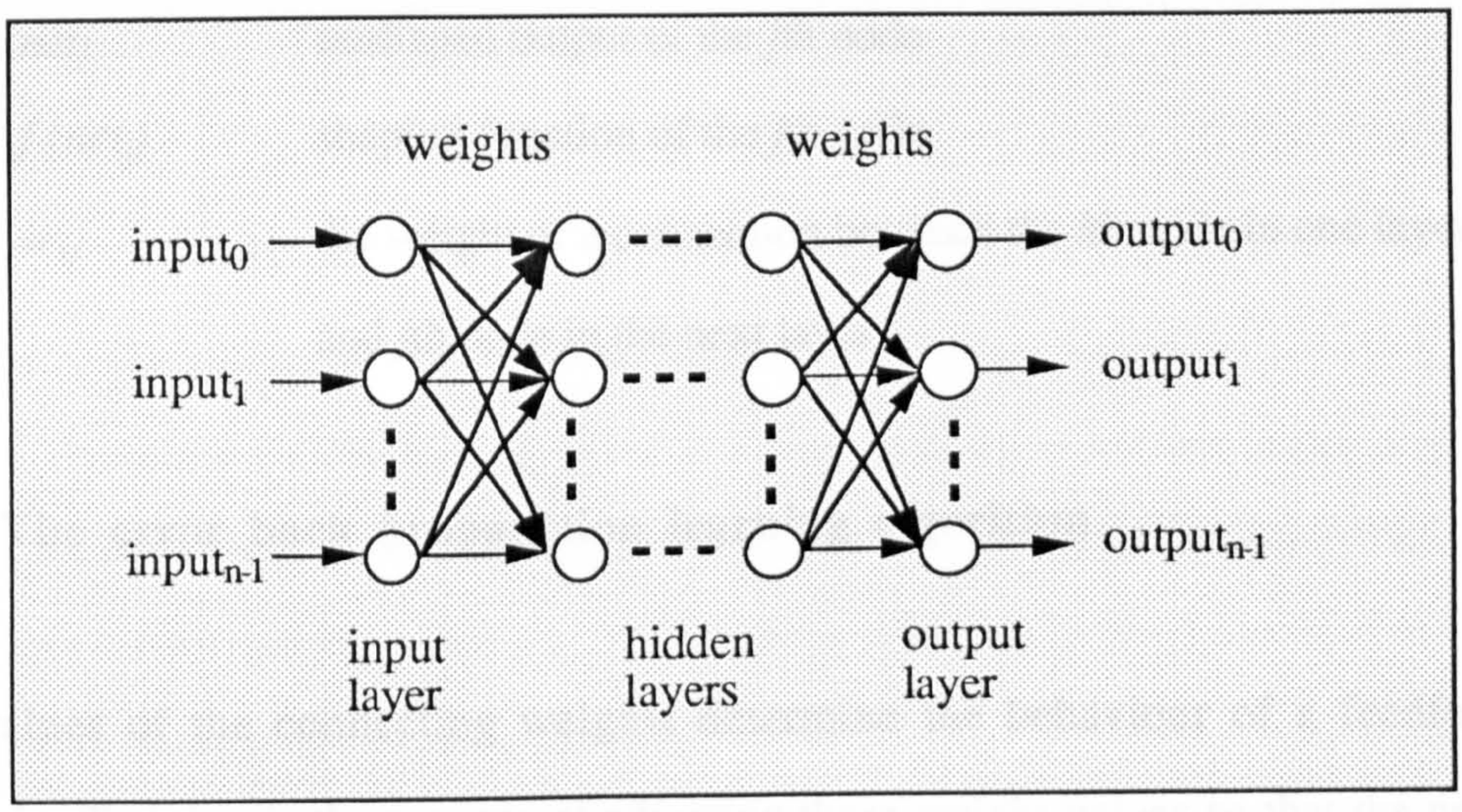

Fig. 5.2 The multi-layer feedforward neural network

The multilayer feedforward network consisted of a set of neurons logically arranged into

three or more layers as illustrated in Figure 5.2. There is an input layer and an output layer, each containing at least one neuron. Neurons in the input layer do not themselves have inputs, and do no processing of any sort. Their output is defined by the network input. There are usually one or more hidden layers sandwiched between the input and output layers. The term "feedforward" means that information flows in one direction only. The inputs to neurons in each layer come exclusively from the outputs of neurons in the previous layer, and outputs from these neurons pass exclusively to neurons in the following layer.

The output of every neuron in the network is computed using equations 5.1 and 5.2. These equations can be expressed as follows.

$$net_j = \sum_{i=0}^{n} w_{ij}\, o_i \qquad 5.3$$

$$o_j = f(net_j) = \frac{1}{1+ e^{-net_j}} \qquad 5.4$$

where:

o_j	output of the jth node
net_j	unshaped output of the jth node
$f(net_j)$	shaping function of the jth node
w_{ij}	the weight of a connection between the ith node in one layer and jth node in the next layer

5.1.3 The error back propagation learning algorithm

The values of the connecting weights determine the behaviour of a feedforward network. Training is the process of adjusting these weight values so that the network behaviour matches some desired behaviour. By far the most common training or learning algorithm for multilayer feedforward network is error back propagation(BP) [Rumelhart 1986]. The mechanism of the back propagation procedure is a

generalisation of gradient descent techniques. The network is presented with a series of pattern pairs. Each pair consists of an input pattern and a target output pattern. Each pattern is a vector of real numbers. The target output pattern is the desired response to the input pattern and is used to determine the error values in the network when the weights are adjusted.

Suppose there are n outputs in the network. For pattern p, the desired outputs are represented as vector $D=(d_{p,0}, ..., d_{p,n-1})$, while the corresponding actual outputs are represented as vector $O=(o_{p,0}, ..., o_{p,n-1})$. The error function E_p is defined as

$$E_p = \frac{1}{2}\sum_{i=0}^{n-1} (d_{p,i} - o_{p,i})^2 \qquad 5.5$$

E_p is a square error function. The function is easily computed and perhaps most importantly, its partial derivative with respect to individual weights can be computed explicitly [Masters 1993]. When E_p approximates to 0, mapping between inputs and outputs for pattern p is realised. The gradient descent technique is applied to change the weights in its original and simplest form by an amount proportional to the partial derivative of the error function E_p in respect to the given weight.

$$\Delta w_{ji} = - \eta \frac{\partial E_p}{\partial w_{ji}} \qquad 5.6$$

where j denotes a neuron in a layer and i a neuron in the preceding layer, and w_{ji} the weight between these two neurons. The constant η is called the learning rate, and is usually set within the range $0<\eta<1$. The learning rate is used to adjust the size of each step leading towards an optimum solution.

One major problem with the conventional BP algorithm is its slow convergence speed. Many improvements have been suggested to accelerate the convergence speed. Equation 5.4 gives the method most commonly used[Monostori 1992] which was employed in

the proposed prototype system.

$$\Delta w_{ji}(t+1) = -\eta \sum_{p} \frac{\partial E_p}{\partial w_{ji}} + \alpha \Delta w_{ji}(t) \qquad 5.7$$

where t is a timestep, α is a momentum factor, usually set within the range $0 \leq \alpha \leq 1$. The effects of the momentum term are to magnify the learning rate for flat regions of weight space where the gradients are more or less constant, and to prevent oscillations.

The computation of the partial derivatives in Equation 5.6 and 5.7 is as follows.

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \qquad 5.8$$

where, according to Equation 5.3

$$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pi} \qquad 5.9$$

Defining,

$$\delta_{pj} = \frac{\partial E_p}{\partial net_{pj}} \qquad 5.10$$

where, δ_{pj} is an error term for pattern p on node j ,.

$$\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj}\, o_{pi} \qquad 5.11$$

For the output neurons

$$\delta_{pj} = \frac{\partial E_p}{\partial net_{pj}} = (d_{pj} - o_{pj}) \frac{\partial o_{pj}}{\partial net_{pj}} = (d_{pj} - o_{pj}) \frac{e^{-net_j}}{(1+ e^{-net_j})^2}$$

$$= o_{pj}(1 - o_{pj})(d_{pj} - o_{pj}) \qquad 5.12$$

For the hidden neurons

$$\delta_{pj} = \frac{\partial E_p}{\partial net_{pj}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}}$$

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_m \frac{\partial E_p}{\partial net_{pm}} \frac{\partial net_{pm}}{\partial o_{pj}}$$

$$= \sum_m \frac{\partial E_p}{\partial net_{pm}} \frac{\partial}{\partial o_{pj}} \sum_l w_{lm}\, o_{pl}$$

$$= \sum_m \frac{\partial E_p}{\partial net_{pm}} w_{jm} = \sum_m \delta_{pm} w_{jm}$$

$$\frac{\partial o_{pj}}{\partial net_{pj}} = \frac{e^{-net_j}}{(1 + e^{-net_j})^2} = o_{pj}(1 - o_{pj})$$

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_m \delta_{pm}\, w_{jm} \qquad 5.13$$

where m is a node in the layer following node j.

Equation 5.7 can be expressed as :

$$w_{ij}(t+1) = w_{ij}(t) + \eta \sum_p \delta_{pj} o_{pi} + \alpha (w_{ij}(t) - w_{ij}(t-1)) \qquad 5.14$$

Doubling all individual errors will quadruple the mean square error, causing difficulty in comparing results[Masters 1993]. Therefore, during training, the network performance is monitored using the root mean square (RMS) error. The RMS error is computed from

$$E_a = \left\{ \sum_{p=1}^{k} \left\{ \left[\sum_{i=0}^{n-1} (d_{pi} - o_{pi})^2 \right] / n \right\}^{\frac{1}{2}} \right\} / k \qquad 5.15$$

where n is the number of the outputs and k is the number of the patterns in the training set.

5.2. The Structure of the Prototype

The structure of the system is illustrated in Figure 5.3. The operational structure of the system consists of three parts: the training module, the test module and the application module. For ease of use, the system has a user-friendly interface which consists of four modules: the file editor, the system set-up window, the training display window and the test window.

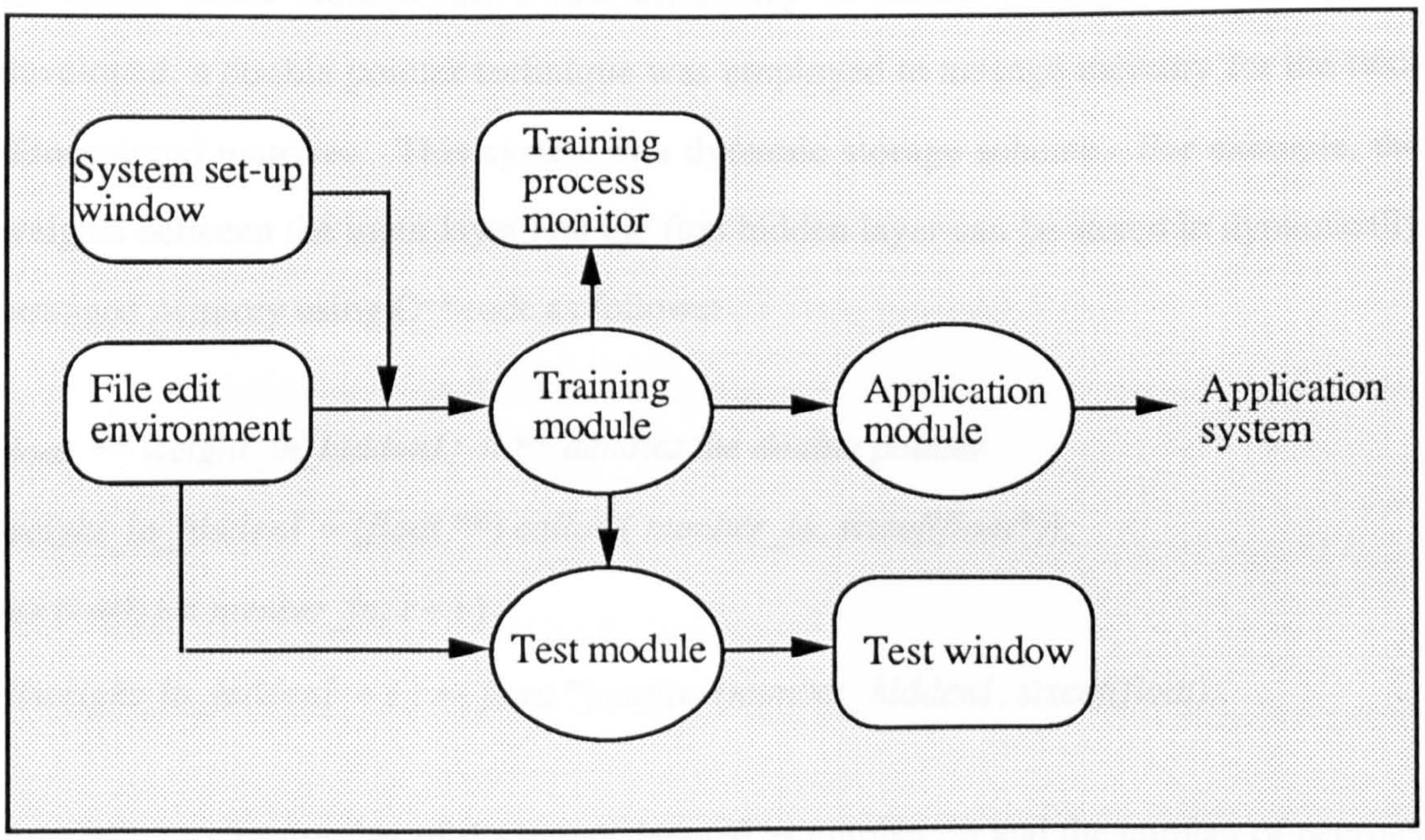

Figure 5.3 The structure of the prototype

First of all, it must be determined how many hidden layers the prototype should have. Theoretically, a BP network with one hidden layer is sufficient to solve most modelling problems, provided sufficient hidden neurons are available[Hornik 1989]. However, in practice, the number of neurons is not unlimited, and a BP network with two hidden layers has been reported to have more general classification boundaries and faster convergence speed than a single hidden layer network[Chester 1990] [Obradovic 1990].

In some cases three hidden layers are required and may provide even faster convergence[NeuDesk]. Therefore, up to three hidden layers were provided in the prototype system. The upper limit of the number of neurons in each hidden layer depends only on the capacity of the computer memory available.

A second problem is that neural networks need a huge amount of memory for training. To speed up the training process, training patterns are read once from training data files and stored in memory. Moreover, large numbers of weights need to be stored in memory. All data are stored in the form of two-dimensional matrices. For a large network, a static storage scheme occupies a lot of memory leading to the problems of insufficient memory. In addition, since the prototype cannot know the size required in advance, static storage has a low efficiency of memory usage. In the system developed, a double pointer technique was employed to arrange memory for the two-dimensional matrices. This system is a dynamic storage scheme. For example, the weights between the input layer and the first hidden layer can be stored in dynamically arranged memory using C++ code as follows:

```
float ** weight_in_hidden1; // ** denotes the double pointer
weight_in_hidden1 = (float **) calloc ( number_in, sizeof(float*));
for (i =0; i < number_in; i ++)
*(weight_in_hidden1 + i ) =( float *) calloc (number_hidden1, sizeof(float);
```

According to the number of the inputs denoted as number_in and the number of neurons in the first hidden layer denoted as number_hidden1, the memory required can be arranged exactly. If the use of the weights finishes, the memory is freed for other use.

5.3 The Executive Modules of the System

5.3.1 Training module

The training method is implemented in two stages. During the forward pass, the outputs

of all neurons are calculated. Then, using a backward pass, starting at the output neurons, the derivatives required for the weight modification are computed. The training procedure is as follows:

1. The weights in the network are initialised to some small non-zero value. A back-propagation network is sensitive to the initial values of weights[Zurada 1992]. Properly selected initial weights can shorten learning time and result in stable weights. In the prototype, the initial weights are randomly selected between -0.2 and 0.2.

2. The output of each neuron is computed for the presented input in order from the input layer to the output layer. This is a forward pass. The computation uses equations 5.3 and 5.4.

3. A comparison is made of the output o_{pj} of each output neuron j against the presented ideal output value d_{pj} and the output error δ_{pj} is computed for each output neuron j using equation 5.12.

4. The errors δ_{pj} are computed using equation 5.13 for each hidden neuron j.

5. The sum of the error squared E_p is computed using equation 5.5.

6. The procedure is repeated from step 2 for the remaining input-out pairs in the training set.

7. The RMS error is computed using equation 5.15. The procedure is stopped if E_a is less than given δ, ; otherwise, training is continued.

8. The weights between the hidden layer and the output layer are adjusted using equations 5.12 and 5.14.

9. The weights between the hidden layers and the weights between the hidden layer and the input layer are adjusted using equations 5.13 and 5.14.

10. The procedure is repeated from step 2.

The programming instructions for training were included in an independent module as illustrated in Figure 5.4. The module communicates with the user through the user interface and communicates with other modules in the system through the configuration file and weight file.

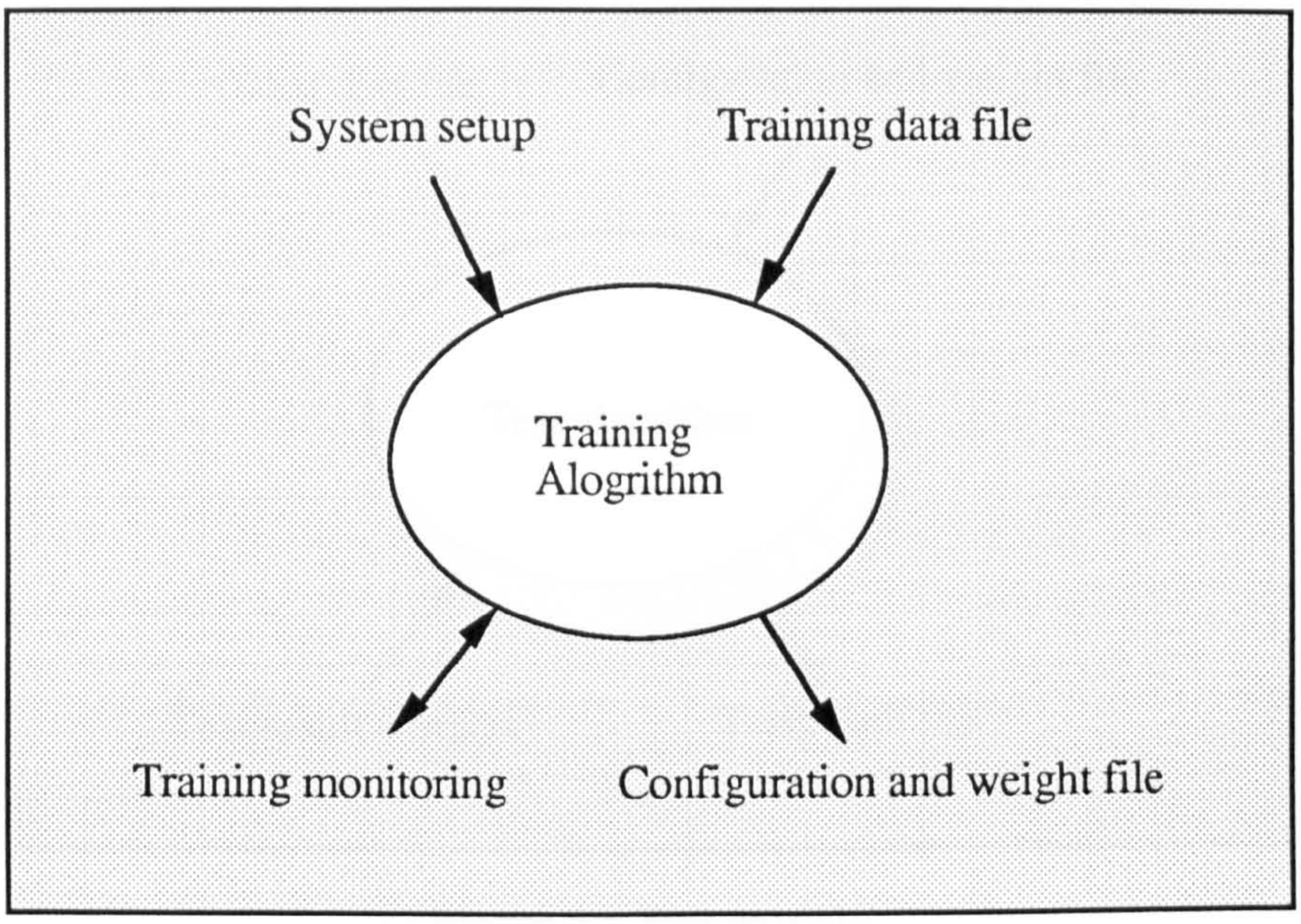

Figure 5.4 Training module

5.3.2 Test module

After a neural network is trained, it must be tested. If the test results are not satisfactory the system must be retrained. The test should use a data set which has not been previously employed by the network. The test procedure is a forward pass as follows:

1. The system configuration and the test data are loaded.

2. The output of each neuron for the test input is computed in order from the input layer to the output layer. The computation uses equations 5.3, 5.4.

3. The computed results are displayed and saved.

4. A comparison is made by the user between each system output and the ideal output in order to judge if the training is successful.

The module is independent as illustrated in Figure 5.5. The module communicates with the other modules via data files.

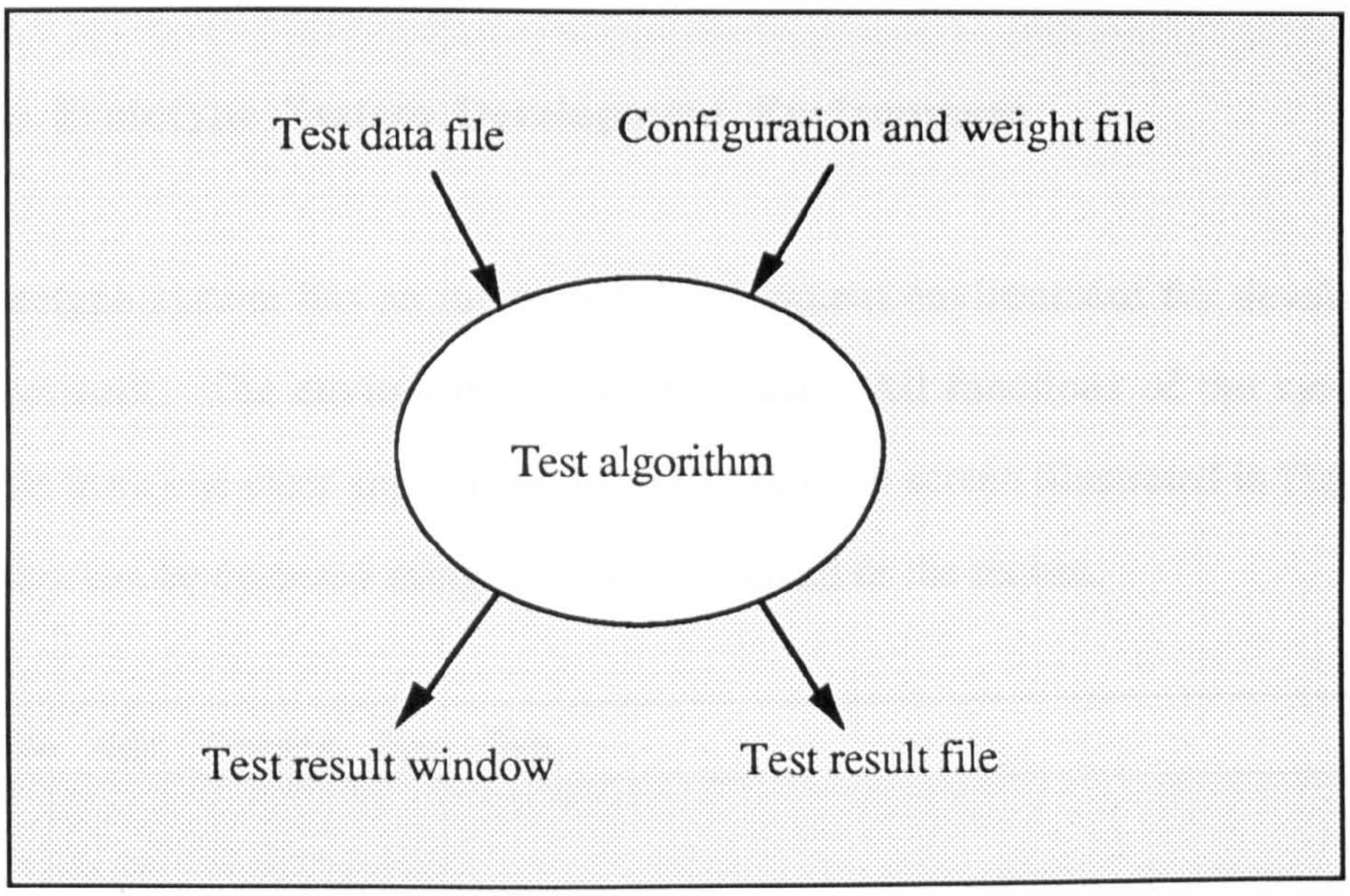

Figure 5.5 Test module

5.3.3 Application module

The application module illustrated in Figure 5.6 is a forward pass algorithm similar to the test module. However, the application module is an independent package which can be embedded into an application system.

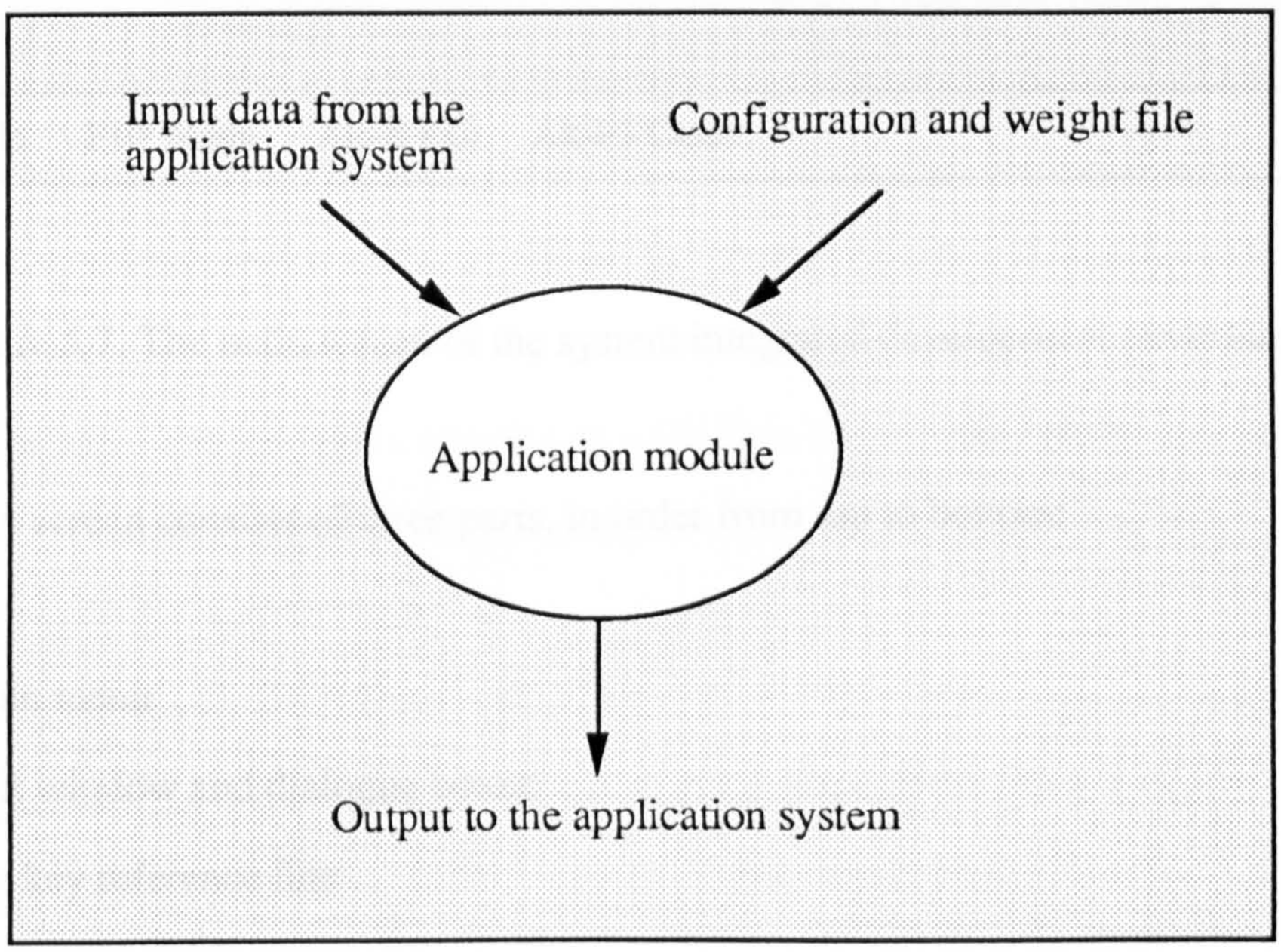

Figure 5.6 Application module

5.4 The Prototype System Development Environment

The prototype system has an integrated development environment for developing a neural network. The environment is easy to use. All functions of the system are accessible from one clear and simple display - the main screen illustrated in Figure 5.7. The system can be operated using either the keyboard or the mouse.

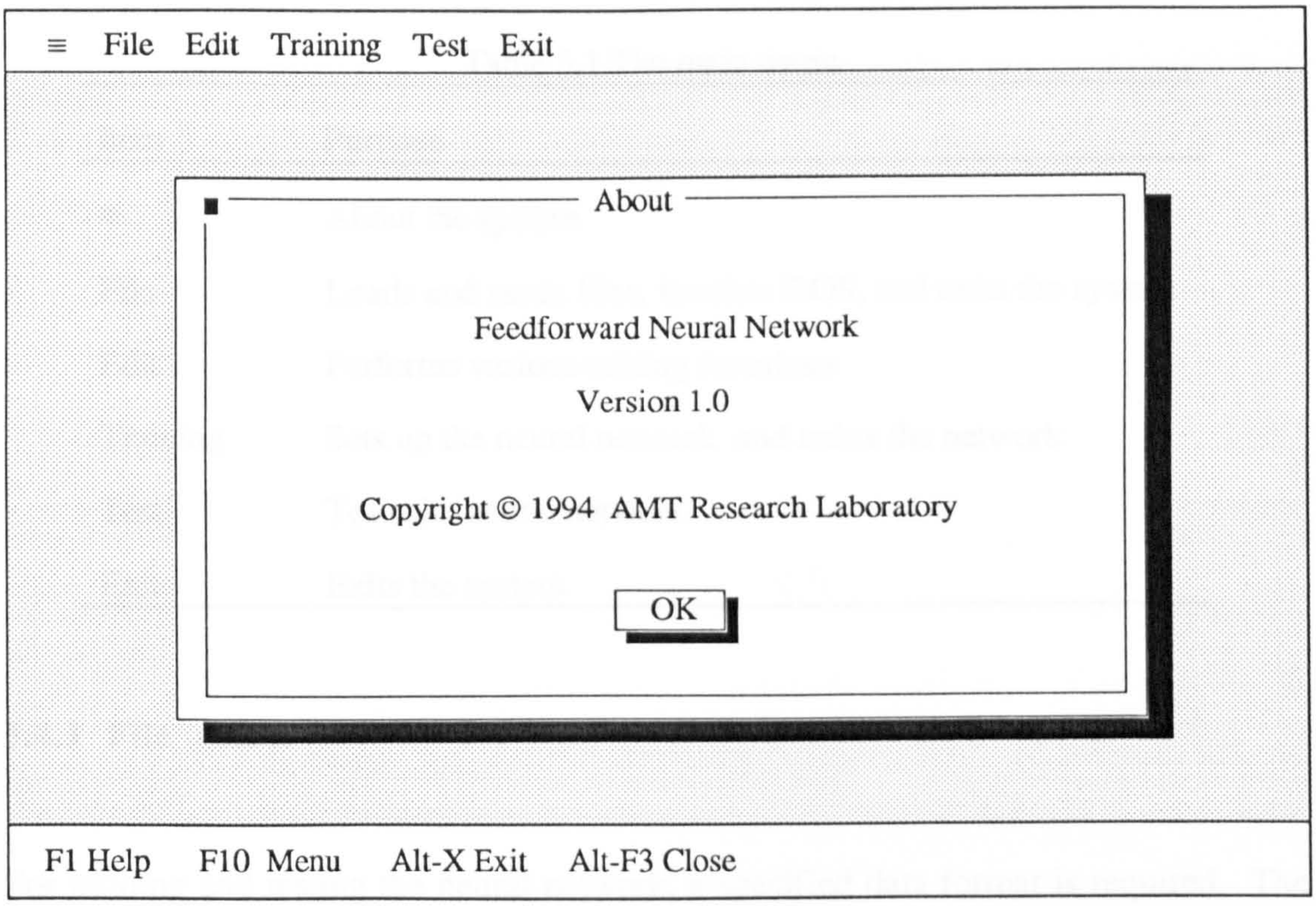

Figure 5.7. The main screen of the system integrated development environment

The main screen consists of three parts, in order from top to bottom:

- The main menu
- The edit window and dialogue boxes
- The hot key reference line

The main menu is used to tell the system to do something. When a main menu item is

selected, a pull down menu is displayed that contains a list of choices. Table 5.1 shows what each menu selection does.

The edit window is used to edit the file and the dialogue box is used to input items that are not easily entered using a menu or display output information.

The hot key reference line displays several hot keys.

Table 5.1 The main menu

Item	Purpose
≡	About the system
File	Loads and saves files, invokes DOS, and exits the system
Edit	Performs various editing functions
Training	Sets up the neural network and trains the network
Test	Tests the neural network
Exit	Exits the system

5.4.1 File editor

For training and testing the neural network, a specified data format is required. The prototype system provides a data editor for the user. The editor is similar to many text editors such as the Borland C++ editor. The user can also employ other editors such as the DOS editor. The file editor consists of a file item and an edit item in the main menu. The editor includes the following functions which are displayed in the pull-down menus.:

•. File item

File open, New file, Save file, Save as, Change directory, DOS shell, Exit

• Edit item

Undo, Cut, Copy, Paste, Show clipboard, Clear.

The data format in the prototype is that each input data item must be separated by a comma while each group pattern is separated by a 'Return line', which is easy to edit. A format example is illustrated in Figure 5.8. When data are saved as a data file, the format presents the information without any additional data about the number of inputs, the number of patterns and the identification for different inputs and patterns . The output data has the same format. The data file can therefore be read easily by different systems. In addition, neural networks require inputs in the range from 0 to 1. Since the net uses the sigmoidal activation function, it restricts the output value to the range between 0 and 1 and it is difficult to produce output values close to 0 or closed to 1. It is suggested that the output operation should be set in the region between 0.1 and 0.9[NeuralDesk 1992].

	Input1	*Input2*	*Input3*	• • •
Pattern1	0.1,	0.5,	1.0,	
Pattern2	0.2	0.9,	0.3,	
Pattern3	0.5,	0.5,	0.0,	
•				
•				
•				

Figure 5.8 Data format

5.4.2 System setup window

The Training item in the main menu includes 'system setup' and 'training network'. Before training a network, the structure of the neural network must be determined. The structure of the system includes the number of hidden layers, the number of the neurons in each hidden layer, the value of the learning rate, the value of the momentum factor and the value of the error threshold. The system setup window is illustrated in Figure 5.9.

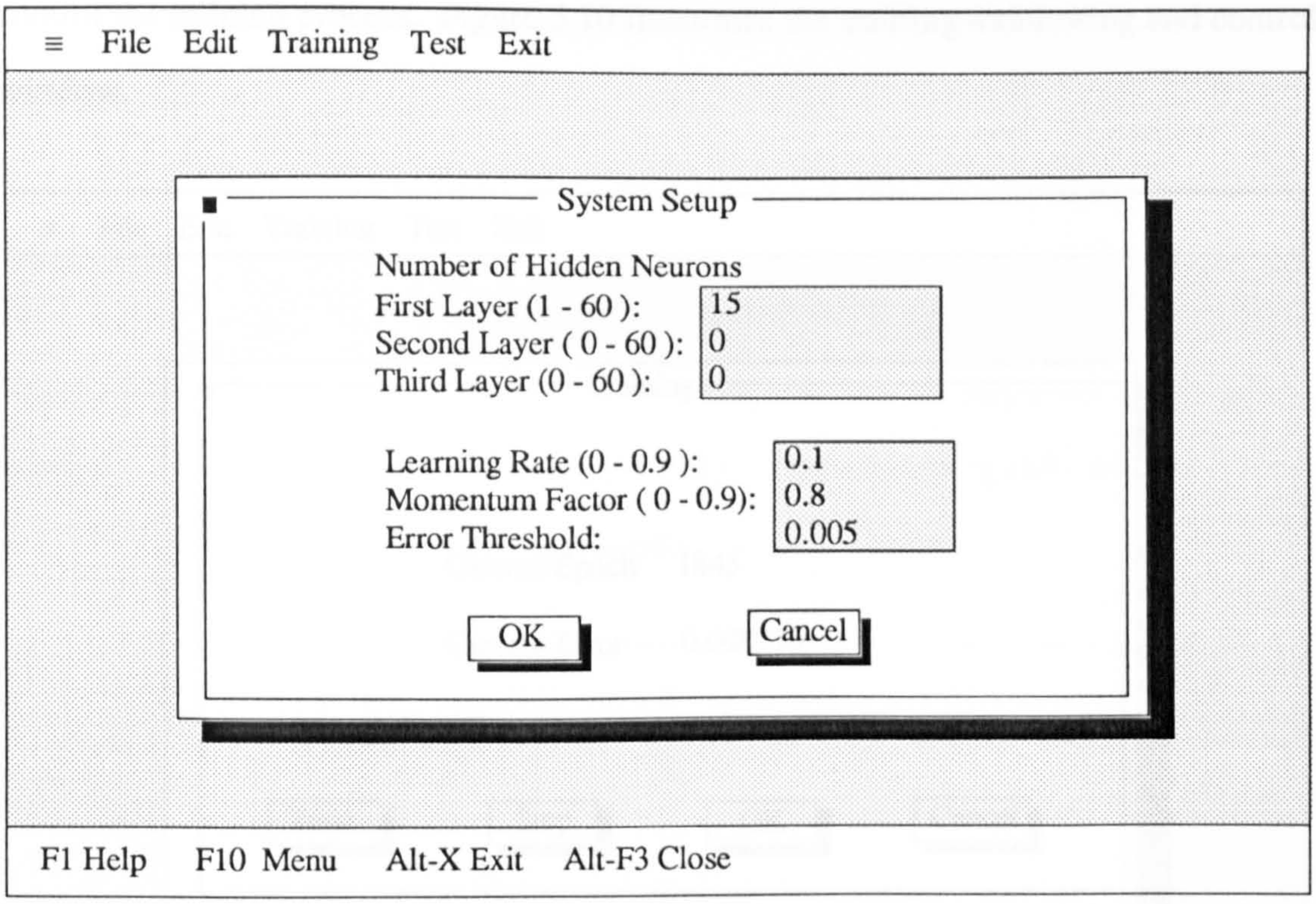

Figure 5.9 System setup menu

The number of input and output neurons of the network is automatically determined by the system according to the training data set. The system reads the input and output data from the training data files, then the system determines the number of inputs and the number of outputs as well the number of patterns according to the number of commas and the number of Returns in the training data set. For example, in the input data file, the number of commas is *n* and the number of Returns is *k*, which means the inputs should be equal to *n*/*k* and the training patterns should be equal to *k*. Similarly, in the out data file, the number of commas is *m* and the number of Returns is *j*, which means the outputs should be equal to *m*/*j* and the training patterns should be equal to *j*. If *n*/*k* or *m*/*j* is not an integer, or the number of input training patterns *k* does not equal the number of output training patterns *j* , the data file has an error and must be checked.

5.4.3 Training process monitoring and control

In the training process, it should be possible to monitor the training information and

control the training process. Figure 5.10 illustrates the training monitoring and control window.

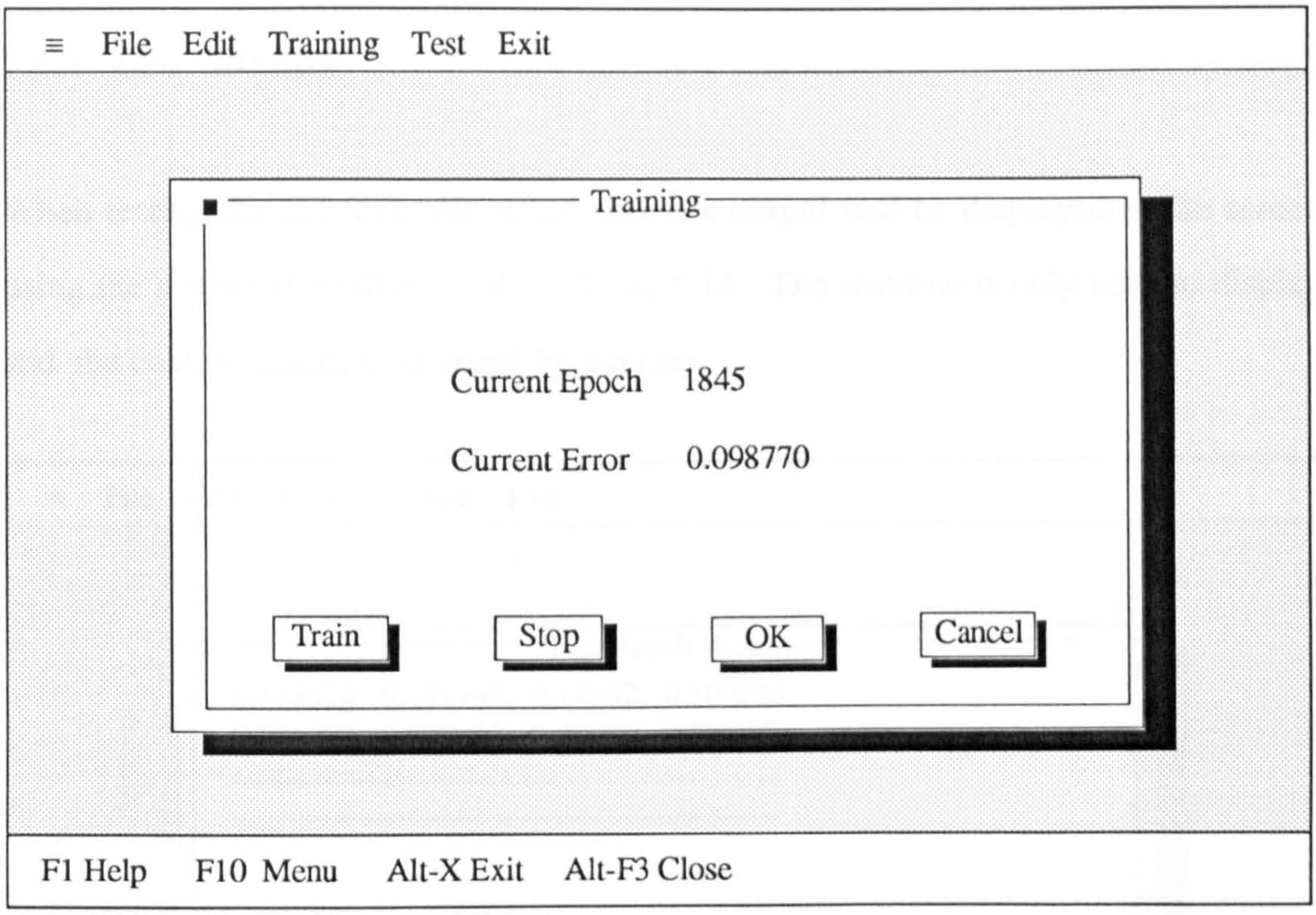

Figure 5.10. Training monitoring and control

The training display has two information outputs: 'current epoch' and 'current error'. The current epoch value refers to the number of times the training process has been completed and the weights adjusted. Current error refers to the average error of the neural network based on equation 5.15 for the current conditions.

The training process can be controlled by the user through the window command buttons. The functions of these buttons in the window are as follows:

• The Train button is used to start the system training

• The Stop button is used to suspend the training process and keep the current system situation in the last epoch. The training process can continue from a stop point if the Train button is pressed.

• The OK button is used to switch off the training window whether the training process is finished or not. The system configuration and the training result or the current system situation can be saved as a file.

• The Cancel button is used to stop the training process and erase the network structure and the window.

5.4.4 Test window

When testing the network, the tested network output will be displayed in the screen using the test window illustrated in Figure 5.11. The window is only used to display and the content cannot be changed by the user.

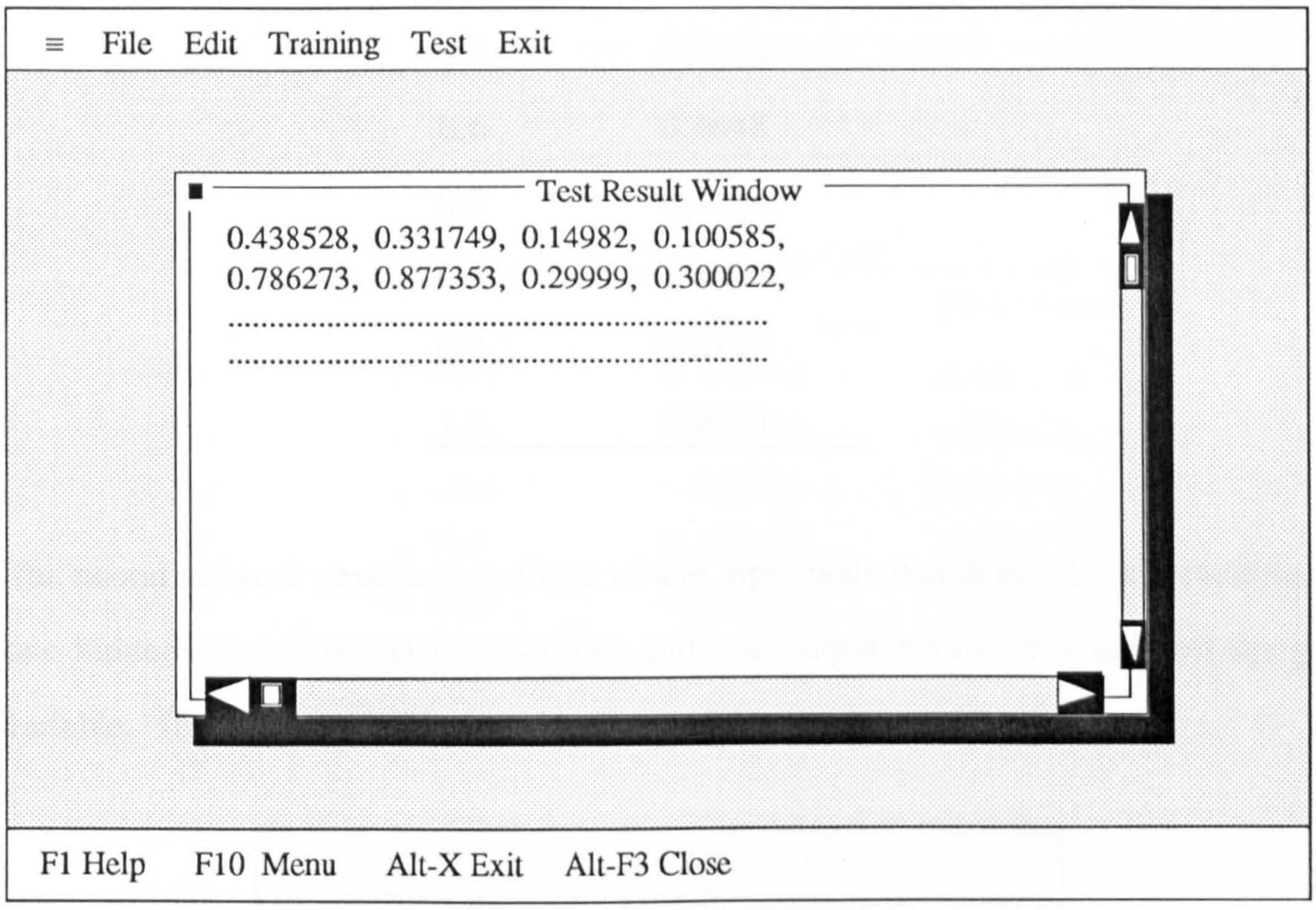

Figure 5.11. Test result window

5.5 The Prototype System Test and Evaluation

The prototype system was tested by modelling and solving a non-linear function. For simplicity, the non-linear function was a two-dimensional one:

$$y = 0.1 + (6x^3 - 9x^2 + 4x)/1.25,\ 0 \le x \le 1 \qquad 5.13$$

10 equally spaced points were defined as training data as illustrated in Table 5.2.

Table 5.2 Training data

x	y
0.0	0.1000
0.1	0.3528
0.2	0.4904
0.3	0.5416
0.4	0.5352
0.5	0.5000
0.6	0.4648
0.7	0.4584
0.8	0.5096
0.9	0.6472
1.0	0.9000

The neural network structure consisted of one input node that denoted the x variable, one hidden layer that had five neurons and one output neuron that denoted the y variable. This structure is illustrated in Figure 5.12.

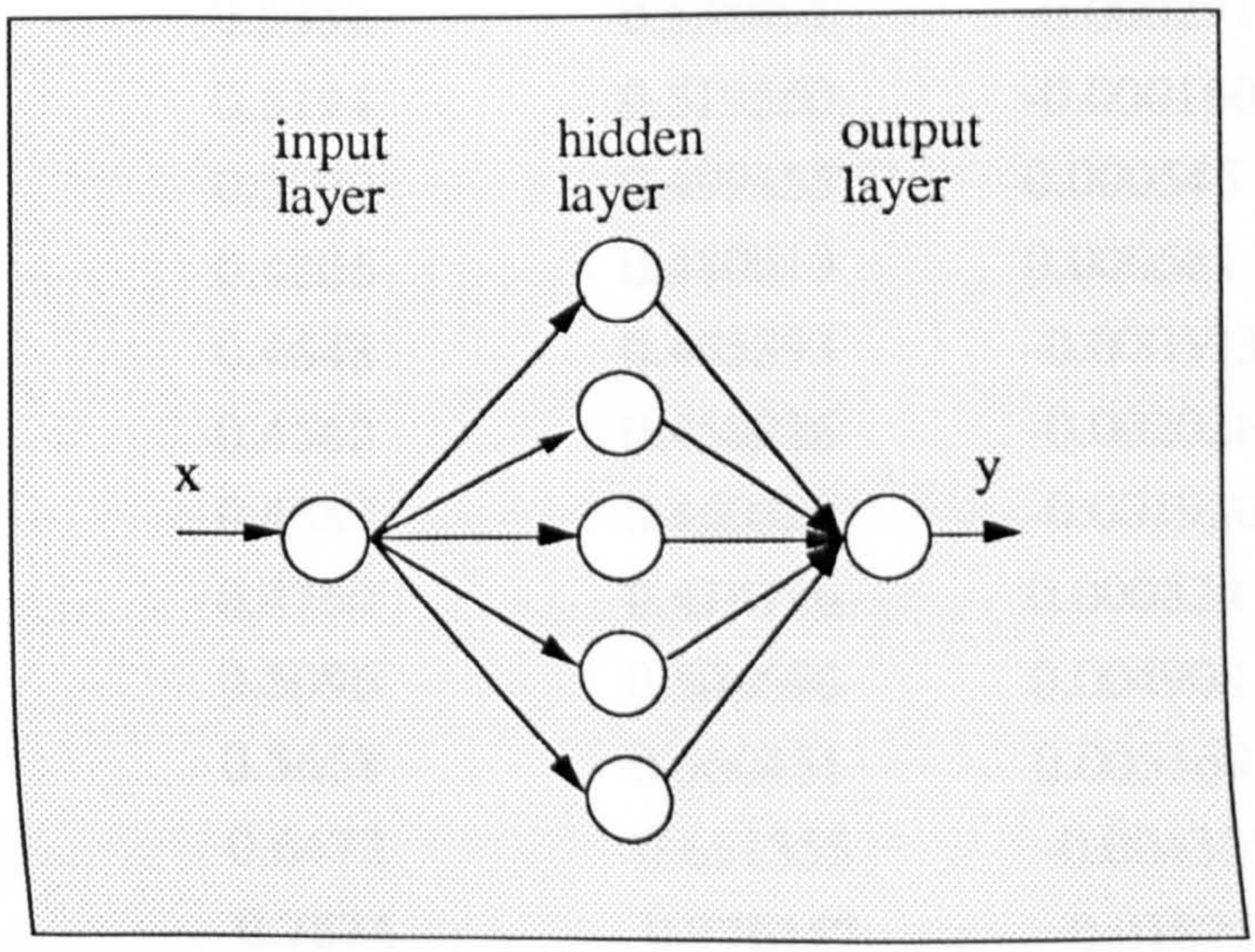

Figure 5.12 The test neural network

The training parameters were:

learning rate:	0.3
momentum	0.9
RMS error threshold	0.002

After training, a set of data were used to test the network. To observe the training results, the test set included the training input data. Table 5.3 illustrates the ideal values derived from the equation, the network output values and the errors. The RMS error can also be used to evaluate the test results. However, the variable k in equation 5.15 is now the number of patterns in the test set instead of in the training set.

Table 5.3. Network function test

x	y ideal value	y output value	error value
0.00	0.1000	0.100996	-0.000996
0.05	0.2426	0.227162	0.015438
0.10	0.3528	0.351074	0.001726
0.15	0.4342	0.438453	-0.004253
0.20	0.4904	0.492768	-0.002368
0.25	0.5250	0.524468	0.000532
0.30	0.5416	0.540210	0.001390
0.35	0.5438	0.543227	0.000573
0.40	0.5352	0.535491	-0.000291
0.45	0.5194	0.519560	-0.000160
0.50	0.5000	0.499455	0.000545
0.55	0.4806	0.480019	0.000581
0.60	0.4648	0.465393	-0.000593
0.65	0.4562	0.458226	-0.002026
0.70	0.4584	0.460320	-0.001920
0.75	0.4750	0.474176	0.000824
0.80	0.5096	0.504696	0.004904
0.85	0.5658	0.560431	0.005369
0.90	0.6472	0.651555	-0.004355
0.95	0.7574	0.776157	-0.018757
1.00	0.9000	0.897088	0.002912
RMS ERROR			0.005816

Figure 5.13 shows that the network models the function within a 1% root mean square error. However, it is obvious that the errors are bigger where the curves are steeper. If the number of training points in these areas are increased the accuracy can be further improved.

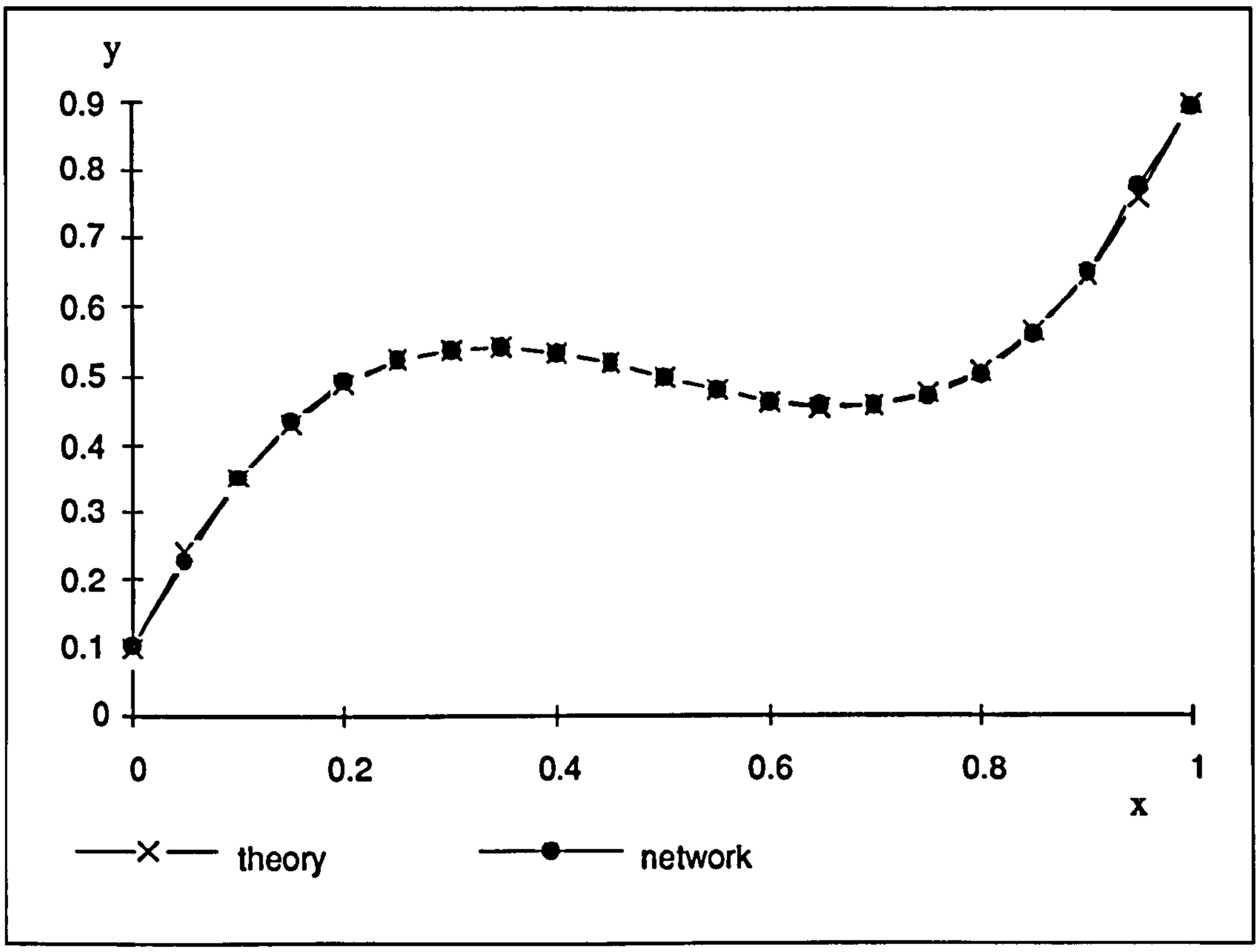

Figure 5.13 Locus of the function and the network approximation

For further evaluation of the prototype, the system was compared with a commercial package, NeuralDesk II[NeuralDesk 1992]. NeuralDesk II is a neural network software package developed by Neural Computer Sciences. The package consists of a feedforward neural network with a backpropagation learning algorithm. Neural Desk is based on the Microsoft Windows environment. The prototype and NeuralDesk were both used to model and solve function 5.13. For exact comparison of the two systems, the same training parameters were employed. Since the average error was calculated using different methods in the two systems, the maximum error method was used, where training is terminated when the error for every output in every pattern in the set is

below the error value set. The training parameters were as follows:

learning rate:	0.1
momentum	0.9
Max. error value	0.004

The results for the two systems are illustrated in Table 5.4

Table 5.4 Test comparison for the prototype and the NeuralDesk

x	y Theoretical value	y Prototype	y NeuDesk
0.00	0.1000	0.100515	0.100544
0.05	0.2426	0.228578	0.227493
0.10	0.3528	0.351883	0.351320
0.15	0.4342	0.437600	0.438516
0.20	0.4904	0.491632	0.492811
0.25	0.5250	0.524148	0.524517
0.30	0.5416	0.540715	0.540153
0.35	0.5438	0.543950	0.542990
0.40	0.5352	0.535893	0.535168
0.45	0.5194	0.519514	0.519345
0.50	0.5000	0.499104	0.499411
0.55	0.4806	0.479521	0.480011
0.60	0.4648	0.464921	0.465347
0.65	0.4562	0.458069	0.458341
0.70	0.4584	0.460768	0.460924
0.75	0.4750	0.475175	0.475264
0.80	0.5096	0.505599	0.505600
0.85	0.5658	0.560251	0.559980
0.90	0.6472	0.650150	0.649419
0.95	0.7574	0.775442	0.774873
1.00	0.9000	0.898159	0.899250
RMS ERROR		0.005382	0.005481

Some performance indexes of the systems are as follows:

	Prototype	NeuralDesk
Training epoch:	102296	114215
Training Time	5 minutes	16 minutes
RMS error	0.005382	0.005481

From the above it can be seen that the two systems have similar performance. However, the prototype ran much faster than NeuralDesk because the prototype is text-based while NeuralDesk is graphic-based. In addition, the prototype can produce a directly embedded package whilst the NeuralDesk only produces a textual report file.

5.6 Local Minima Problem of the BP Algorithm

The backpropogation algorithm is not guaranteed to converge to the correct solution. This is because that the backpropogation algorithm may settle in a local minimum. However, in practice the problem of local minima can usually be overcome by multiple starts with different random weights[Lippmann 1987]. The high-dimensional weight space provides a large number of degrees of freedom. In addition, the momentum term in equations 5.7 and 5.14 helps to prevent the algorithm converging at a local minimum.

5.7 Summary

A prototype neural network development tool has been successfully developed and tested. The system has several advantages:

- Using a text-based environment, the system runs much faster than a Windows-based system. This is very important because training is time consuming.
- The prototype employs a user-friendly interface.
- The application module of the prototype can be easily integrated into an application system.

Chapter 6 Grinding Wheel Selection Using a Neural Network

Wheel selection using a neural network was not only developed as an independent system but also as an intelligent agent integrated into a multi-agent system for selection of the grinding conditions. The system developed emphasised the external grinding operation and the selection of aluminium oxide and silicon carbide wheels.

6.1 The Structure of the Wheel Selection System

The neural network system treats the wheel selection process as a black box as illustrated in Figure 6.1.

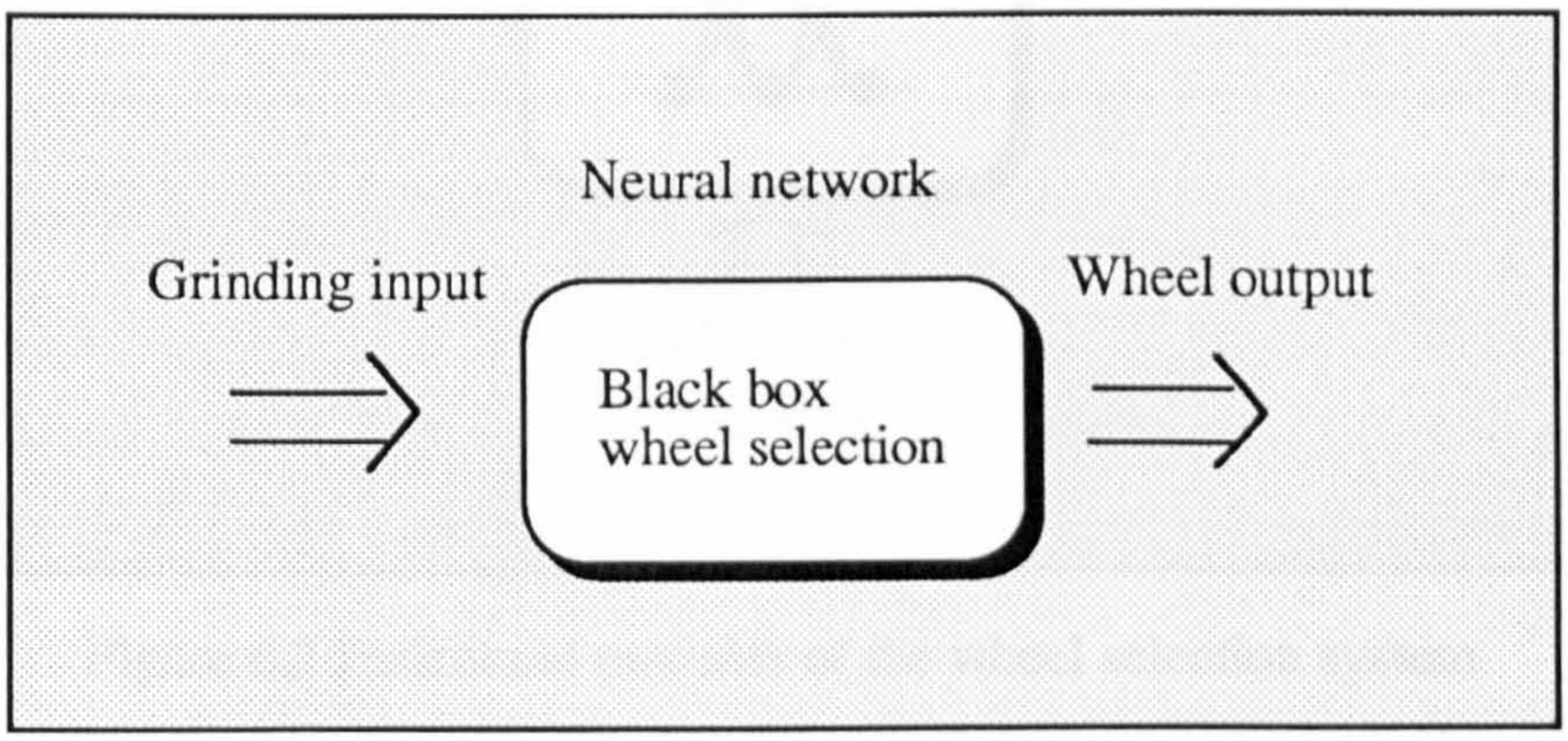

Figure 6.1 neural network modelling

The system consists of input, output, encode and neural network functional modules. The system is illustrated in Figure 6.2.

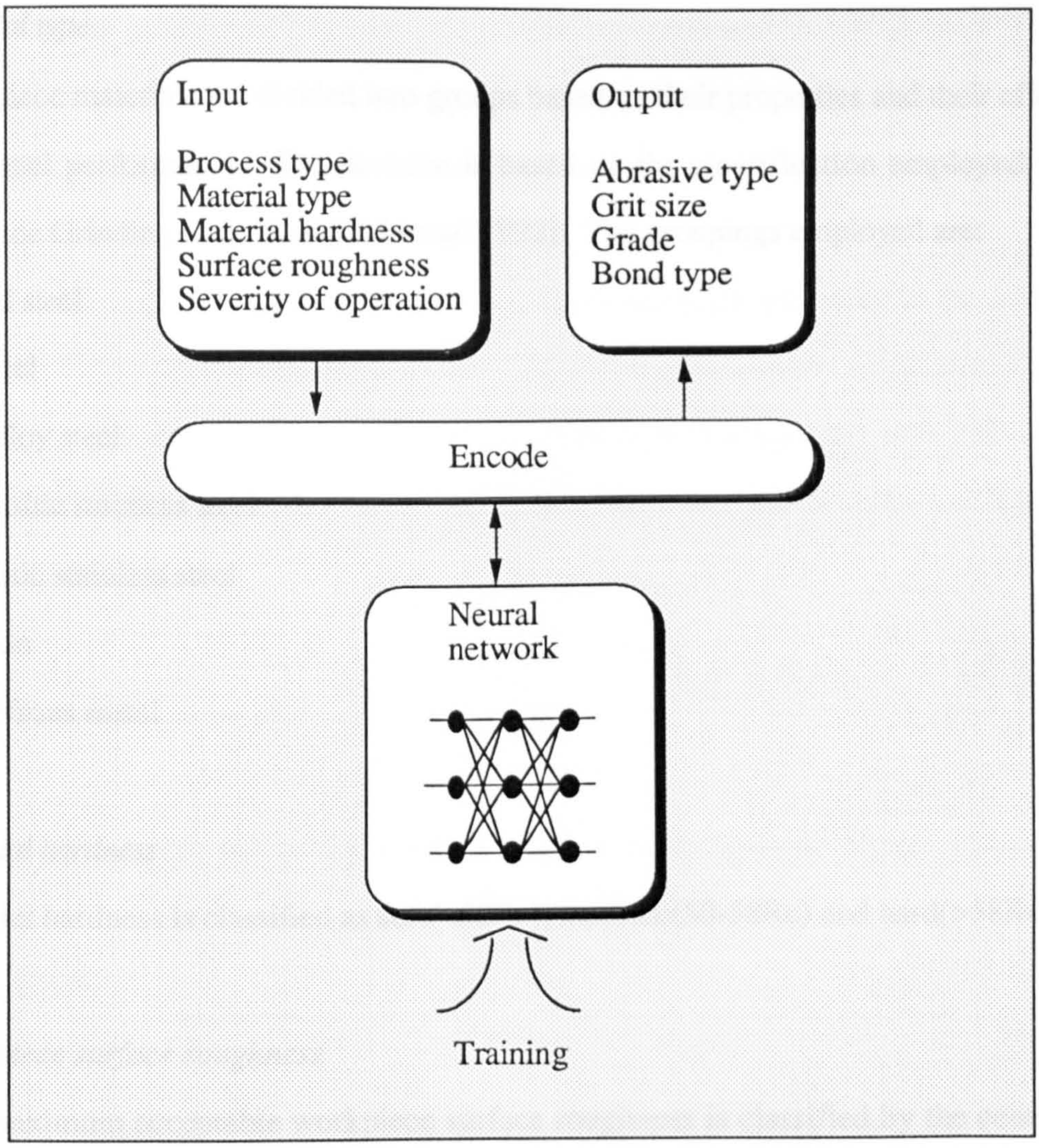

Figure 6.2 Functional modules of the wheel selection system

6.1.1 The system input

In Chapter 3, the factors affecting grinding wheel performance were described. The wheel selection system was based on those factors. The input variables are summarised as follows:

Process type:

Process types included external, surface, internal and centreless grinding. The grinding methods include traverse grinding and plunge grinding.

Material type

Workpiece materials are divided into groups based on their properties and their effect on the wheel performance. The division is based on the classification employed in the reference Grinding Data Book[Universal 1992]. The groupings employed are:

general steel

tool steel

high alloy steel

martensitic stainless steel

austenitic stainless steel

cast iron

non-ferrous metal

Material hardness

Material hardness is classified as soft(<50Rc), medium(50-58Rc) and hard(>58Rc).

Workpiece surface roughness

The maximum acceptable workpiece surface roughness is classified by the centre line average value in μm.

Severity of operation

Severity of operation is classified under the following headings[Universal 1992]:

rough cast or forged

interrupted cut

wide wheel, light pressure

narrow wheel, high pressure

large diameter workpiece

small diameter workpiece

The severity of operation is constituted by four independent inputs:

rough cast or forged

interrupted cut

wide wheel, light pressure **or** narrow wheel, high pressure

large diameter workpiece **or** small diameter workpiece

The items in different groups can be chosen simultaneously according to the operating situation. However, the items in the same group cannot be chosen simultaneously. For example, 'interrupted cut' and 'small diameter workpiece' can be chosen simultaneously. If the operating situation does not include a 'severity' condition, no items need be chosen.

6.1.2 The system output

The specification of the grinding wheel is given as the system output . The output includes four variables in the specification:

abrasive type

abrasive grit size

grade

bond

6.1.3 Encoding

When training the network, the input and output training pairs must be encoded into a form that the neural network can recognise so that the input information can be recognised and the output results interpreted. The encoding methods are important for the effectiveness of the neural network. Suitable methods can ease the training process and allow higher accuracy.

Material encoding

Materials were encoded using the one-of-*n* method which uses the binary code. The method uses as many input neurons as there are values that the variables can take. There

are 7 material groups and therefore 7 neurons are allocated to the material inputs. Exactly one of the neurons will be turned on according to the value of the variables. All of the other neurons will be turned off. The advantage of the encoding method is not to imply any order relationship among these variables. However, this method requires more neurons. These codes are illustrated in table 6.1

Table 6.1 Material encoding

	encoding						
material	neu1	neu2	neu3	neu4	neu5	neu6	neu7
general steel	1	0	0	0	0	0	0
tool steel	0	1	0	0	0	0	0
high alloy steel	0	0	1	0	0	0	0
martensitic stainless steel	0	0	0	1	0	0	0
austenitic stainless steel	0	0	0	0	1	0	0
cast iron	0	0	0	0	0	1	0
non-ferrous metal	0	0	0	0	0	0	1

Surface roughness encoding

Surface roughness is an ordinal variable. Surface roughness is encoded using the one-of-one method, namely, one input neuron expresses all possible values as illustrated in Table 6.2.

Table 6.2 Roughness encoding

roughness (R_a)	>0.9	0.7-0.9	0.4-0.7	0.2-0.4	<0.2
encoding	0	0.2	0.4	0.6	0.8

Hardness encoding

Material hardness is an ordinal variable, and was therefore encoded using the one-of-one method illustrated in Table 6.3.

Table 6.3 hardness encoding

hardness	<50Rc	50-58Rc	>58Rc
encoding	0	0.5	1

The encoding of the severity of operation

There are various variables in the severity of operation. All variables could be independently encoded to train a neural network, but it would make the system complicated and a large training set would be required to cover the high order of multi-dimensional space. To make the system more effective, an encoding method using some rules derived from grinding knowledge was employed.

The severity of operation as well as the grinding method, traverse or plunge grinding, mainly affect the selection of the abrasive grit size and the grade. The encoding method employed merges these variables into two codes. One code corresponds to the effect on abrasive grit size while the other code corresponds to the effect on grade. Table 6.4 shows the effects of the severity of operation and the grinding method[Universal 1992].

Table 6.4 The effects of the severity of operation

	grit size	grade
rough cast or forged	1 size coarser	1 grade harder
interrupted cut	1 size finer	2 grades harder
wide wheel, light pressure	1 size coarser	1 grade softer
narrow wheel, high pressure	1 size finer	1 grade harder
large diameter workpiece	no change	1 grade softer
small diameter workpiece	no change	1 grade harder
plunge grinding	1 size finer	no change

Accordingly, the codes are shown in Table 6.5. 0.1 denotes no change. 0.0 and 0.2 or 0.3 denotes change in two different ways as explained below.

Table 6.5 Severity of operation encoding

	code1	code2
rough cast or forged	0.0	0.2
interrupted cut	0.2	0.3
wide wheel, light pressure	0.0	0.0
narrow wheel, high pressure	0.2	0.2
large diameter workpiece	0.1	0.0
small diameter workpiece	0.1	0.2
plunge grinding	0.2	0.1

Where some items can be chosen simultaneously, the codes are merged. The principle employed to merge these items is that if the effects of the items are all positive or are all negative the maximum effect is used as the input and if the effects of some items are positive but the rest are negative the average effect is used as the input. The rule is separately applied to code1 and code2. The rule is:

IF min code(items) > 0

THEN code = max code (items)

IF min code(items) = 0

AND max code(items) >0

THEN code = max code (items) - 0.1

IF max code = 0

THEN code = 0

For example, if 'interrupted cut' and 'large diameter workpiece' are chosen, the input code is:

code1 = max code(0.2, 0.1) = 0.2

code2 = max code(0.3, 0.0) - 0.1 = 0.2

The encoding method uses existing grinding knowledge to achieve a simple neural network and training process. Since the effect of these factors is relatively small, the wheel selection is not highly sensitive to the accuracy of the knowledge.

Output encoding

The wheel specification includes abrasive type and bond type, grit size and grade. However, to prevent the system becoming too complicated, the output employs one-to-one encoding. The encoding is illustrated in Table 6.6.

Table 6.6 Output encoding

Abrasive type	11A	48A	51A	WA	C						
Encoding	0.1	0.3	0.5	0.7	0.9						
Grit size	24	30	36	46	60	80	100	120	150	180	220
Encoding	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Grade	Q	P	O	N	M	L	K	J	I	H	G
Encoding	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Bond type	Vitrified	Resinoid	Rubber	Shellac							
Encoding	0.1	0.2	0.3	0.4							

6.2 The Neural Network Topology Architecture

A three layer network with one hidden layer was employed for wheel selection. The number of input and output neurons of the neural network is determined by the input and output as well as the encoding form. Therefore, the system has 11 input neurons, 7 for workpiece material, 1 for material hardness, 1 for surface roughness and 2 for grinding method and severity of operation. The system has 4 output neurons, one each for abrasive type, grit size, grade and bond. However, there is no rule for the selection of the optimal number of neurons in the hidden layer. A straightforward way to deal with the problem is to start from a small number of neurons, then , gradually increase the neurons in the hidden layer until an acceptable accuracy and speed of convergence is

obtained[Rangwala 1989]. According to this, twelve neurons were required in the hidden layer. The three layer neural network for wheel selection is illustrated in Figure 6.3.

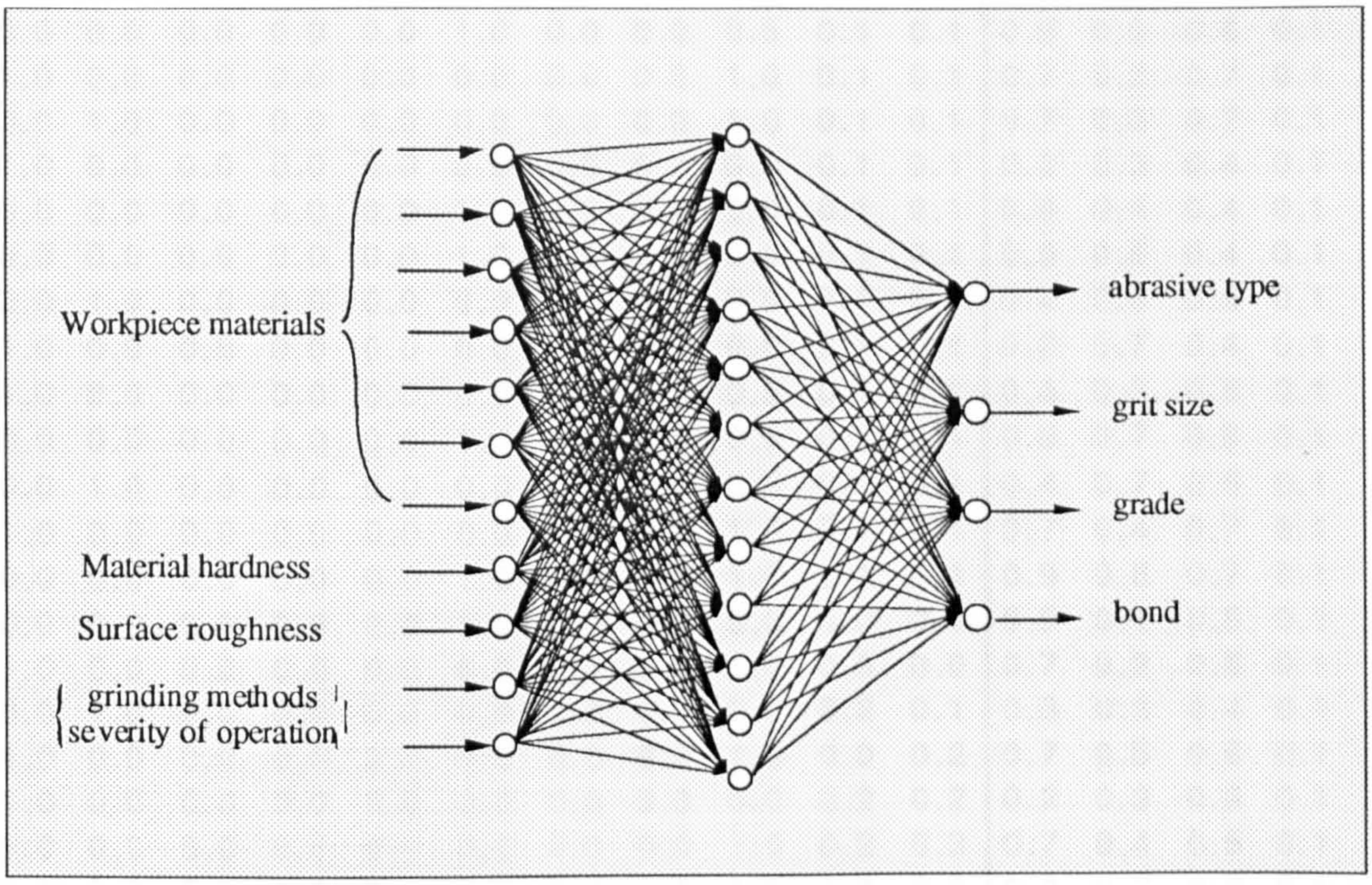

Figure 6.3. Neural network for wheel selection

6.3 The Training of the Neural Network

Data sets in sufficient volume and coverage suitable for training were not available from recent grinding experience. Training data were therefore taken from published handbooks, for example, the Grinding Data Book published by Unicorn Abrasives UK Ltd[Universal 1992] and the Machining Data Handbook published by Metcut Research Associates Inc.[MDC 1980].

Training data were collected to cover the required problem space. Forty seven examples were taken as training data spread over the required problem space. Table 6.7 illustrates the encoding of the examples in the training set.

Table 6.7 Training examples and the encoding

Input											*Output*			
ste	*tool*	*sup*	*ms*	*as*	*cas*	*n-f*	*rou*	*har*	*s1*	*s2*	*abr*	*grit*	*gra*	*bon*
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.4	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.3	0.2	0.4	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.1	0.1	0.9	0.2	0.4	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.5	0.1	0.1	0.9	0.2	0.6	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.1	0.1	0.7	0.3	0.7	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.1	0.1	0.7	0.3	0.7	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.2	0.3	0.4	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.4	0.0	0.1	0.1	0.9	0.4	0.4	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.6	0.0	0.1	0.1	0.9	0.5	0.4	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.6	0.5	0.1	0.1	0.4	0.5	0.6	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.1	0.1	0.2	0.7	0.4	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.5	0.1	0.1	0.4	0.3	0.6	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.8	0.5	0.1	0.1	0.9	0.7	0.6	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.8	0.5	0.1	0.1	0.4	0.7	0.6	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	1.0	0.1	0.1	0.7	0.4	0.7	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.8	1.0	0.1	0.1	0.9	0.8	0.7	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.9	0.1	0.5	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.7	0.2	0.8	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.1	0.3	0.3	0.4	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	1.0	0.0	0.2	0.7	0.5	0.6	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.3	0.3	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.2	0.3	0.7	0.4	0.5	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.8	1.0	0.0	0.0	0.9	0.7	0.8	0.1
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	0.2	0.3	0.7	0.9	0.5	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.8	0.0	0.0	0.0	0.9	0.6	0.5	0.1
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.2	0.3	0.9	0.3	0.2	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	0.2	0.3	0.7	0.9	0.5	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.3	0.2	0.7	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.4	0.2	0.7	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.4	0.3	0.7	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.2	0.0	0.1	0.1	0.9	0.3	0.7	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.1	0.1	0.9	0.2	0.7	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.4	0.0	0.2	0.1	0.3	0.5	0.7	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.6	0.0	0.1	0.1	0.9	0.5	0.7	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.6	0.0	0.1	0.1	0.9	0.5	0.7	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.8	0.0	0.1	0.1	0.3	0.7	0.7	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.8	0.0	0.1	0.1	0.4	0.7	0.7	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.8	0.0	0.1	0.1	0.9	0.7	0.7	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.8	0.0	0.1	0.1	0.9	0.7	0.7	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.1	0.0	0.9	0.2	0.8	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	0.3	0.2	0.7	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.4	0.0	0.2	0.3	0.4	0.5	0.5	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.8	0.0	0.2	0.1	0.9	0.8	0.7	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.2	0.2	0.9	0.3	0.6	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.8	0.0	0.2	0.3	0.3	0.8	0.5	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.4	0.6	0.8	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.8	0.0	0.1	0.3	0.9	0.7	0.5	0.1

The meanings of the abbreviations in Table 6.7 are as follows:

ste	general steel
tool	tool steel
sup	high alloy steel
ms	martensitic stainless steel
as	austenitic stainless steel
cas	cast iron
n-f	non-ferrous metal
rou	roughness
har	hardness
s1	severity of operation code1
s2	severity of operation code2
abr	abrasive type
grit	abrasive grit size
gra	grade
bon	bond

The meanings of these codes are given in Table 6.1-6.6. For example No.1:

Workpiece material:	General steel
Material hardness:	< 50 Rc
Surface roughness:	> 0.9 Ra (μm)
Severity of operation:	None
Grinding method:	Traverse grinding
Wheel specification	11A36MV or 48A 36MV

The settings for the training parameters were:

learning rate:	0.1
momentum	0.6
RMS error value	0.005

The learning rate and the momentum were chosen based on training trials to achieve a fast training process. The training process performance results were:

training epoch: 61465

training time 83 minutes

6.4 Network Test

To test the trained wheel selection system, a test procedure was required. The test used an example set that had not been previously employed by the network. These examples were collected from the same sources as the training examples. The input of test examples and the encoding are illustrated in the Table 6.8. The meanings of these codes are shown in Table 6.1-6.5.

Table 6.8 Test input codes

ste	*tool*	*sup*	*ms*	*as*	*cas*	*n-f*	*rou*	*har*	*s1*	*s2*
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	1.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	1.0	0.1	0.3
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.4	0.5	0.2	0.3
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.2	0.3
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.6	0.0	0.0	0.1
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.6	1.0	0.1	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.6	0.0	0.1	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.4	0.0	0.2	0.1
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.4	0.0	0.1	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.6	0.0	0.0	0.1
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.8	0.0	0.2	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.1	0.1
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.2	0.0	0.2	0.3
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.6	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.4	0.0	0.2	0.3
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.1	0.1
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.6	0.0	0.1	0.0

The required output codes for the inputs and the network output codes are illustrated in Table 6.9.

Table 6.9 The output codes

Required output				*Network ouput*			
as	grit	grad	bon	as	grit	grad	bon
0.7	0.3	0.8	0.1	0.709619	0.279449	0.790056	0.100000
0.2	0.6	0.5	0.1	0.192915	0.554553	0.529875	0.100000
0.7	0.5	0.5	0.1	0.711707	0.461198	0.493476	0.100002
0.9	0.5	0.4	0.1	0.923340	0.454635	0.434441	0.099991
0.9	0.3	0.2	0.1	0.890099	0.298992	0.205683	0.099999
0.9	0.4	0.4	0.1	0.884158	0.415262	0.369250	0.100000
0.7	0.6	0.7	0.1	0.646959	0.631757	0.686459	0.099994
0.3	0.5	0.7	0.1	0.297592	0.516528	0.688266	0.099999
0.3	0.5	0.7	0.1	0.304983	0.499852	0.699258	0.099995
0.3	0.4	0.8	0.1	0.285608	0.396980	0.759940	0.099997
0.4	0.4	0.7	0.1	0.410171	0.378865	0.708914	0.100000
0.4	0.5	0.8	0.1	0.368516	0.540422	0.776152	0.099993
0.4	0.2	0.6	0.1	0.437132	0.184437	0.640509	0.100004
0.9	0.2	0.7	0.1	0.876583	0.223106	0.682052	0.099999
0.9	0.4	0.5	0.1	0.916948	0.357931	0.531671	0.100001
0.9	0.4	0.8	0.1	0.889208	0.386951	0.783292	0.099999
0.9	0.5	0.5	0.1	0.889075	0.502965	0.451223	0.099995
0.9	0.2	0.7	0.1	0.886889	0.198336	0.690944	0.100003
0.9	0.5	0.8	0.1	0.906652	0.489587	0.811433	0.099994

The wheel specifications corresponding to Table 6.9 are shown in Table 6.10.

Table 6.10 Wheel specifications

Required output	*Network ouput*
WA46IV	WA46IV
11A,48A100LV	11A100LV
WA80LV	WA80LV
C80MV	C80MV
C46OV	C46OV
C60MV	C60MV
WA100JV	WA100JV
48A80JV	48A80JV
48A80JV	48A80JV
48A60IV	48A60IV
48A,51A60JV	51A60JV
48A,51A150IV	48A150IV
48A,51A36KV	51A36KV
C36JV	C36JV
C60LV	C60LV
C60IV	C60IV
C80LV	C80LV
C36JV	C36JV
C80IV	C80IV

All the test results fitted the requirement. Therefore, the training may be considered to be successful.

6.5 Alternative Wheel Selections

The neural network gives a single specification for the wheel. The wheel is probably the

most suitable recommendation based on the training data set and the grinding requirements. However, if the wheel recommended is not available the user needs a suggestion for an alternative. The system provides choices based on the following principles.

• The wheel grade can change a grade softer or a grade harder than the grade recommended. For example, if the grade recommended was K, the alternative may be J or L. The range of grades is from E to Z.

• The grit size can be changed to a size finer or a size coarser than the grit size recommended. For example, if the grit size recommended is 80, the alternative may be 100 or 60. Frequently used grit sizes are 24, 30, 36, 46, 60, 80, 100, 120, 150, 180, 220

• The abrasive type can be changed to a basically similar type. For example, the type 48A can be replaced by the 51A or 51A can be replaced by 48A. The 11A can be replaced by the 48A. However, Type WA for grinding hardened steel and Type C for grinding low tensile strength materials such as cast iron and non-ferrous materials are not recommended replacements for ensuring the grinding performance.

6.6 The CBN Wheel

The above description does not include the selection of CBN wheels because currently there is insufficient CBN wheel information available to train the neural network. To all situations, the Machining Data Handbook[MDC 1980] provides only a choice, **B100T100B**. For a CBN wheel, the single recommendation was adopted for the system as a basic specification for a guide to the user. According to the principles of selecting the grit size and the grade of conventional wheel, it might be assumed that the same principles are suitable for the CBN wheel. However, there is insuffient evidence to prove the assumption.

6.7 Implementation of the Wheel Selection System

6.7.1 The system configuration

Two versions of the wheel selection system were developed, one for incorporation into the system for grinding conditions and the other for independent wheel selection. The system described here is the independent system for wheel selection. The system consisted of two parts, 'Training' and 'Enquiring'. The training process was completed using the neural network prototype system described in Chapter 5. The wheel selection system directly uses the training result through a link to the application network which is a module in the neural network tool. This relationship between training and application is illustrated in Figure 6.4.

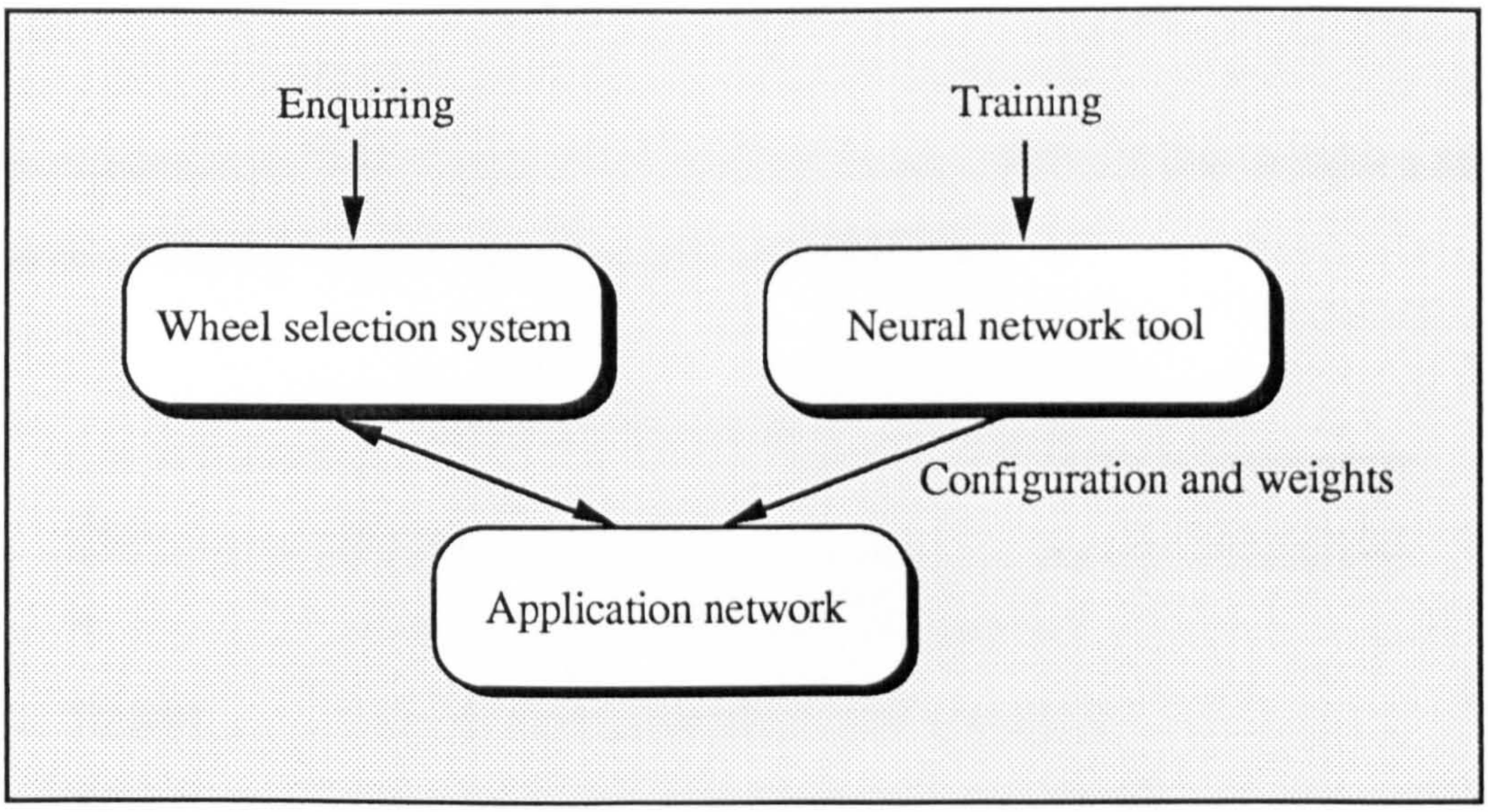

Figure 6.4 Training and application

6.7.2 The system user interface

The main screen

The wheel selection system has a main screen which consists of the main menu, the dialogue area and the hot key reference line. The main screen is illustrated in Figure 6.5

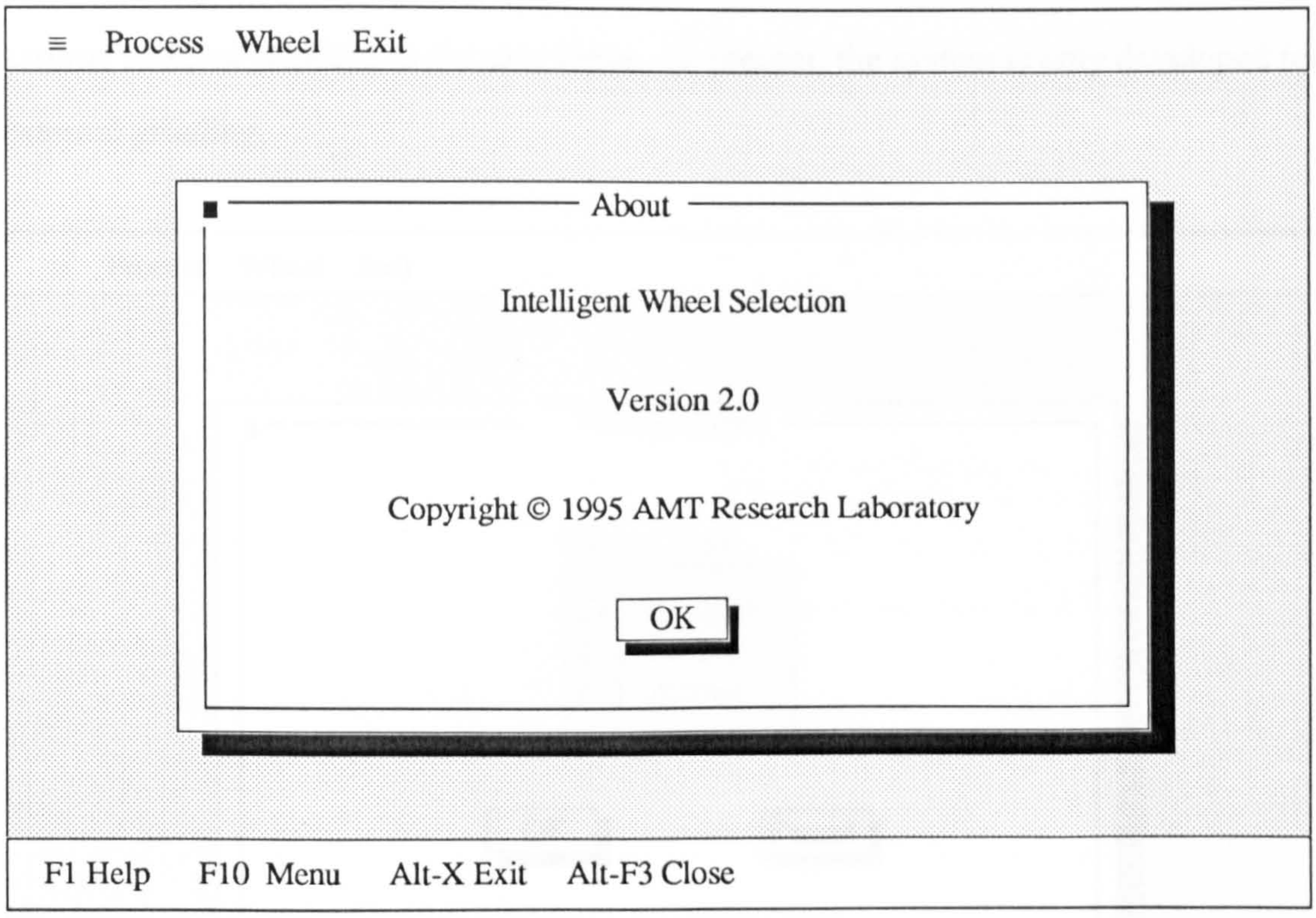

Figure 6.5 The main screen of the wheel selection system

When a main menu item is selected, a pull down menu is displayed that contains a list of choices. Table 6.11 shows what each menu selection does.

Table 6.11 The content of main menu

Item	Purpose
≡	About the system illustrated in Figure 6.5
Process	Choose which grinding process is required
Wheel	Input workpiece information and output result
Exit	Exits the system

Process input

Different grinding processes require different grinding wheels so that the user is required to input the grinding process information first. Because different processes require different grinding information, the process item was separated from the grinding information. The process selection dialogue is illustrated in Figure 6.6. The system

default value is external grinding. If the user does not use the menu, the system will assume external grinding as the user input. At present, the system is only developed for external grinding.

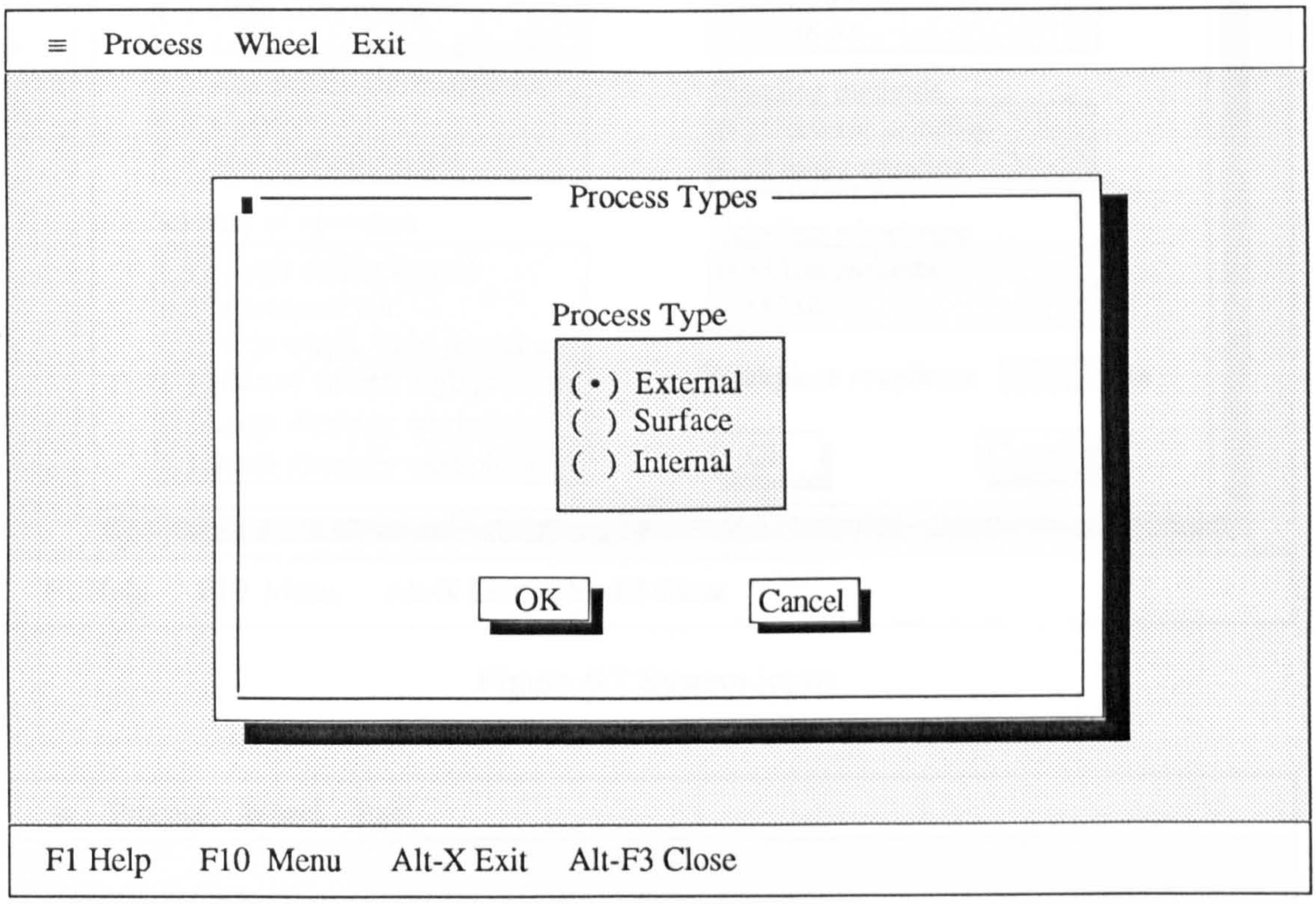

Figure 6.6 Process input

Workpiece information input and system output

After the process has been selected, the workpiece information is entered. The input dialogue for external grinding is illustrated in Figure 6.7. When the system input is complete and the OK button has been pressed, the system will retain the information and give the result for the wheel selection using the neural network. Based on the result, the system will list four choices. Figure 6.8 illustrates the output screen. If the user wants more choices, use of the 'More" button, will provide alternatives based on the principles described in section 6.7.

≡ Process Wheel Exit

System Input

Workpiece materials

(•) General steels
( ) Tool steels
( ) High alloy steels
( ) Martensitic stainless steels
( ) Austenitic stainless steels
( ) Cast iron
( ) Non-ferrous metals

Workpiece Hardness

( ) <50 Rc
(•) 50-58 Rc
( ) > 58 Rc

Grinding methods

(•) Traverse grinding
( ) Plunge grinding

Severity of operation

[ ] Rough cast or forged
[×] Interrupted cut
[ ] Wide wheel, light pressure
[ ] Narrow wheel, high pressure
[×] Large diameter workpiece
[ ] Small diameter workpiece

Grinding wheel type

(•) Conventional
( ) CBN

Workpiece roughness 0.40 Ra

OK Cancel

F1 Help F10 Menu Alt-X Exit Alt-F3 Close

Figure 6.7 System input

≡ Process Wheel Exit

Output

The recommendation is

48A60LV

Alternatives are

48A60MV
48A60KV
48A80LV
48A46LV

OK More

F1 Help F10 Menu Alt-X Exit Alt-F3 Close

Figure 6.8 System recommendation

6.8 Application Example

The following example illustrates how the system works.

The user inputs the following information:

Process type: Cylindrical grinding

Workpiece material: General steel

Material hardness: 50 - 57 Rc

Surface roughness: 0.4 Ra (μm)

Severity of operation: Large diameter workpiece and interrupted cut

Grinding method: Traverse grinding

Wheel type: Conventional

The result displayed on the screen is

The recommendation is

48A60LV

Alternatives are

48A60MV

48A60KV

48A80LV

48A46LV

6.9 Conclusions

This study of the potentional application of neural network techniques for grinding wheel selection demonstrates the following features:

(i) A neural network system is easily and quickly developed when training data are available.

(ii) The system is flexible, allowing for further learning from new data to enlarge and improve the knowledge base

(iii) A large amount of knowledge requires only a small amount of memory.

(iv) A major limitation of the system is the lack of sufficient examples from recent practice to train the system., especially there is a lack of CBN wheel examples.

Chapter 7 The Selection of Grinding Conditions Using Case Based Reasoning

7.1 The Case Based Reasoning Approach

Figure 7.1 illustrates the mental process adopted by an operator in seeking to determine suitable grinding conditions for a new problem. In this process, the key point is the use of experience instead of rules. Figure 7.2 illustrates a case-based reasoning approach designed to mimic the decision process of a skilled operator. The CBR system developed aims to produce suitable grinding conditions based on past optimal cases stored in a case base. Instead of developing a knowledge base that contains explicit rules, the CBR system is based on a case base of prior cases or examples.

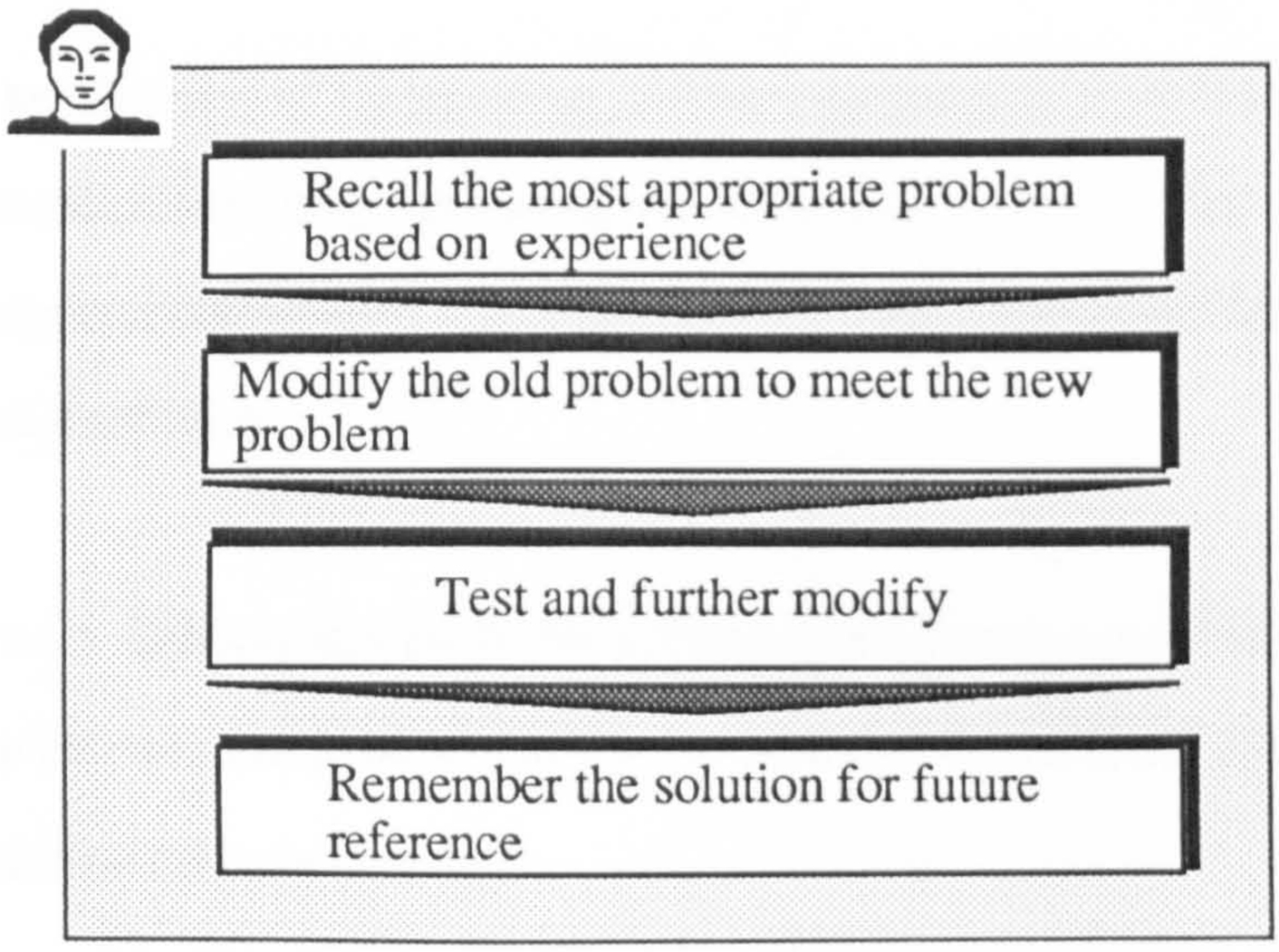

Figure 7.1. Typical procedure for dealing with a new problem

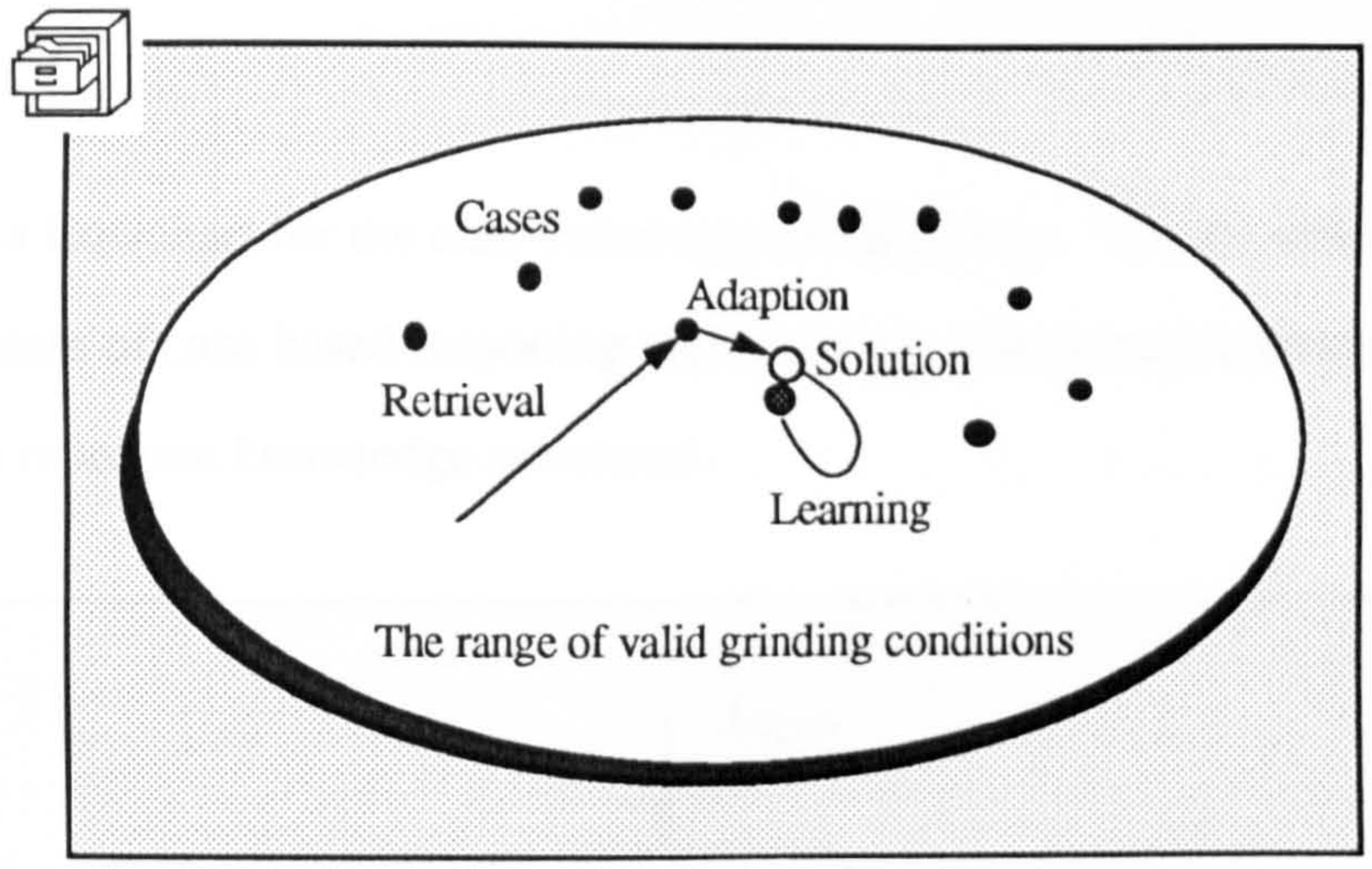

Figure 7.2. Optimisation of grinding conditions

The basic approach proposed for the development of the case-based reasoning system for grinding was as follows:

(i) Each of the key features of the grinding problem are assigned an index. Indexation is employed to establish similarity between cases, since the indexes define and clarify the relationships between elements of a problem that are considered to be important for the determination of a solution.

(ii) Past cases with indexes similar to the problem specification are retrieved from the case base. The indexes assigned to the key variables of the problem specification are matched with similar cases in memory.

(iii) The old case from memory is adapted to match the new problem. If an exact case exists in memory, it is used directly. If there are almost similar cases, the nearest case is adapted to conform to the new demands.

(iv) The new solution is tested. A new solution must be tested to decide if it indeed solves the problem. If the test is successful, the new case is stored in the case base for future retrieval. If the solution proposed is unsuccessful, the case must be further

modified.

Figure 7.3 is a flowchart for the case based reasoning system. The flowchart illustrates the basic process of case based reasoning and learning. The boxes represent processes, and the ovals represent knowledge structures.

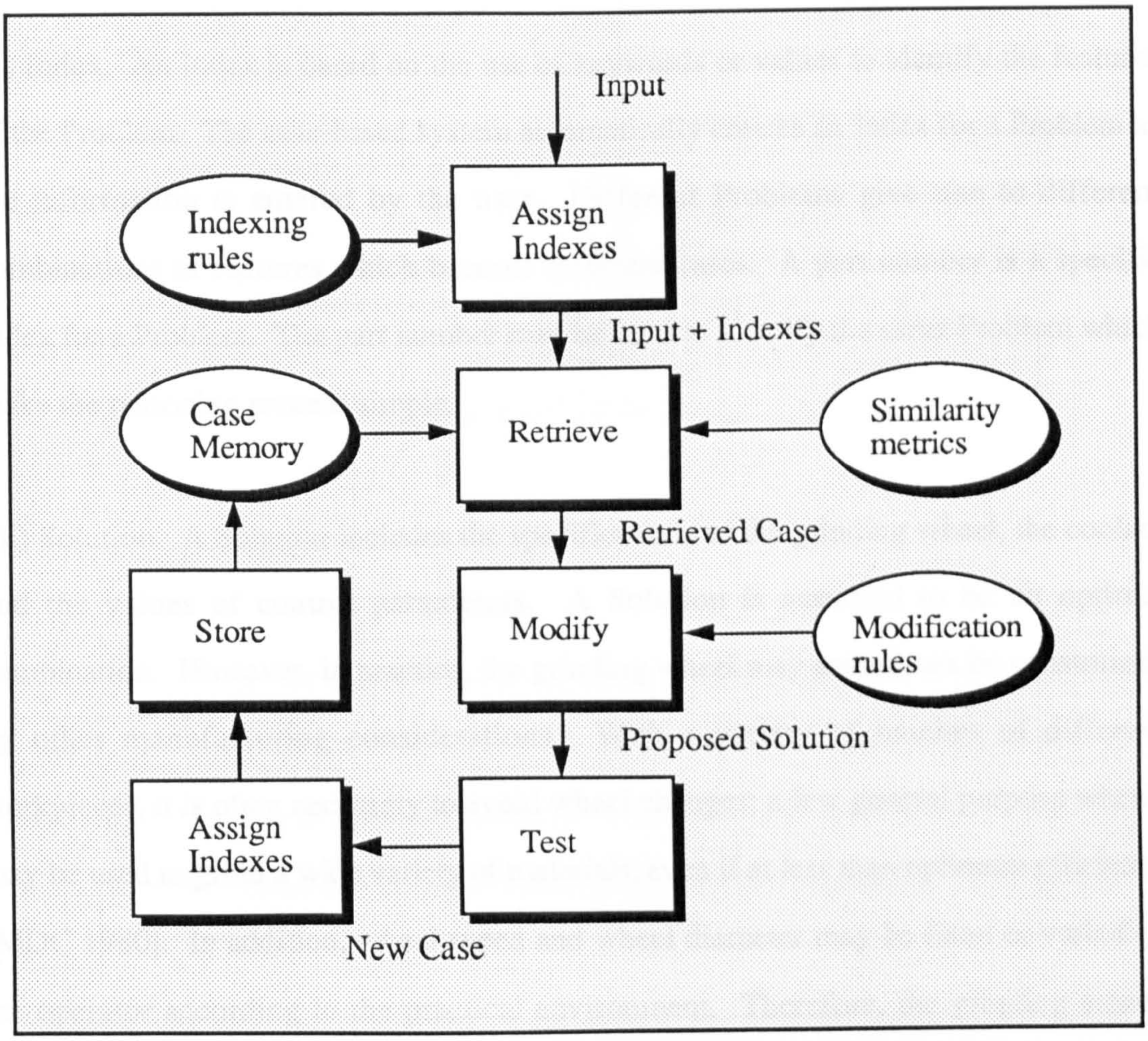

Figure 7.3 The flowchart of case base reasoning

7.2 Case Representation

A case base was designed to contain knowledge about previously successful grinding operations. The cases contain information about the grinding operation, such as the process, the wheel, the coolant, workpiece data and values of control parameters. Each case is a practical machining record relating technological parameters and machining results. Ideally, these cases should be optimal, or at least, close to optimal. A case

consists of three main parts, Problem, Index, and Solution, where the use of the capital letter denotes the specific definitions given below.

(i) Problem. A Problem consists of the workpiece information and the machining requirements which cannot be changed by the operator as described in Chapter 3.

(ii) Index. An Index is based on the use of keywords or values to identify the features of the Problem. The case-based system automatically creates an Index for a Problem as the information is entered by the user. Different Problems give rise to different combinations of features which become different cases. A part number is a specific Index for a Problem. The part number may be used to identify the same Problem which make the reasoning process simple.

(iii) Solution. A Solution includes the specifications of the grinding wheel, the coolant and the values of control parameters. A Solution is assumed to be an optimal combination. However, in practice, the grinding wheel may sometimes be constrained by other manufacturing considerations. With a variety of batches of different workpieces, it is often necessary to avoid wheel changes; a few general purpose wheels may be used to grind a wide variety of materials, even if at less than optimum efficiency [MDC 1980]. In addition, wheel speed and wheel diameter may be fixed or varied by the operator according to the practical environment. Therefore, the grinding wheel, wheel speed and wheel diameter may be taken either as part of the Solution or as part of the Problem according to the requirements of the production situation. However, the more variables the Problem includes, the more restrictions the Solution has, that is, there will be fewer suitable cases in the case base. The only dressing tool currently considered is a single point diamond. If another type of dressing tool is employed the dressing conditions will need to be adjusted appropriately.

The prototype case description is illustrated in Figure 7.4.

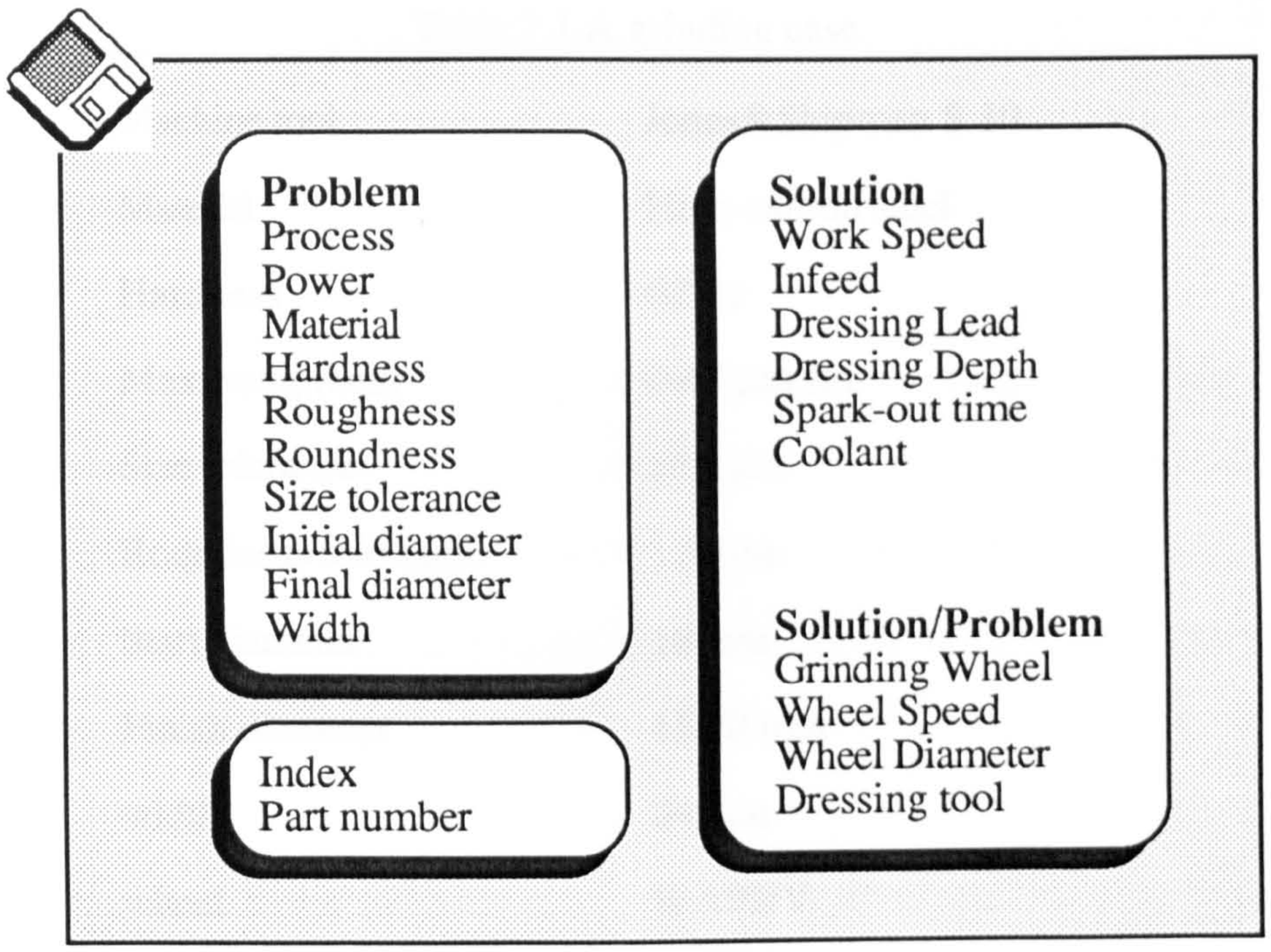

Figure 7.4 Case prototype

7.3 Case Collection

Cases in the case library should provide as much coverage as possible of the nature of the Problem. However, as with all human operators, a case based reasoning system needs to be given time to learn and develop. Therefore, case collection has two stages; initial collection of cases and case collection by learning from operational experience.

Initial collection of cases can be conceived as coming from several different places: an existing database, journals, textbooks, human experts and grinding experiments. These cases should be optimal or reasonable combinations of grinding conditions. Table 7.1 illustrates a grinding case taken from an experiment[Chen 1995].

Table 7.1 A grinding case

Machine tool	Jones&Shipman S 10
Material	High-carbon steel
Hardness	62 Rc
Max. roughness	0.45 μm Ra
Size tolerance	±12 μm
Roundness tolerance	1.5 μm
Start diameter	16 mm
Finish diameter	15.70 mm
width	24 mm
wheel	WA80JV
Wheel speed	33 m/s
Wheel diameter	365 mm
Dresser	Single point diamond
Coolant	Water based
Workspeed	0.25 m/s
Feedrate	0.02 mm/s
Dressing depth	0.015 mm
Dressing lead	0.15 mm/rev
Spark-out time	4 s
Specific energy	60 J/mm^3
Grinding ratio	42

Most grinding cases described in publications do not provide complete information. This is possibly because investigators were concerned with a specific aspect of the process rather than the whole grinding problem. It is therefore difficult to find enough grinding cases to cover an adequate Problem space in the initial case collection. The case based reasoning system therefore must learn during the process of its usage. This is a key issue which decides whether case based reasoning will be successful or not. A major advantage of CBR is the automation of the process of incorporating new

knowledge into an existing knowledge base.

7.4 Indexation

There are three basic approaches to indexation: nearest neighbour, inductive and knowledge-based[Barletta 1991]. The proposed system employs a knowledge-based approach which applies existing knowledge to determine which features are important for case retrieval. This is the preferred approach to indexation if such knowledge is available[Barletta 1991].

If the grinding conditions for a Problem can be deduced from an existing case to an acceptable accuracy, the two cases are considered to be similar. If the definition of similarity is too strict, the number of suitable cases may be very small and if the definition is too loose, the accuracy of the Solution will be poor. The features of a Problem are therefore divided into the categories 'Very Important', 'Important' and 'Unimportant' according to the grinding domain knowledge. Very Important indexes are used to index the basic similarity of cases. For example, initially in the selection process it is considered to be Very Important that materials to be ground should have similar grindability and grinding wheels should have the same abrasive type, bond type and grit size. The Very Important indexes are used to determine a set of applicable cases. If there is a mismatch in this category the case is rejected. The Important indexes are used to choose the nearest case from those applicable cases determined by the Very Important indexes. If there is a mismatch in this category the case may be modified to fit the Problem. The features which do not affect the selection of cases are assumed to be Unimportant. Unimportant features are ignored. The process of deciding the categories of features depends on grinding knowledge and is discussed below.

The Indexes are described by a set of symbols. The symbolic set is called the Index Vocabulary. Usually, the Index Vocabulary is a subset of the vocabulary used for full symbolic representations of cases[Koloden 1993]. The selection of the symbolic

vocabulary should be clear and simple. In the Solution or Problem part, if an item is indexed as 0, it denotes that the item is to be determined as part of the Solution and is not constrained as part of the Problem.

Workpiece material Indexing

The workpiece material which has various properties is the most important factor to affect grindablity. Materials are therefore grouped based on experience according to the similarity of properties affecting grindability. Associated properties include the density, the melting temperature, the specific heat and the thermal conductivity. It is assumed that materials in the same group have similar properties so that they can use the same starting conditions. Table 7.2 gives approximate properties of materials taken from references[Ashby 1992][Kalpakjian 1991].

Table 7.2 Properties of the material groups

Material	Density (kg/m^3)	Tensile strength (MPa)	Melting temperature (K)	Specific heat ($Jkg^{-1}K^{-1}$)	Thermal conductivity ($Wm^{-1}K^{-1}$)
Mild steel	7.9	430	1765	482	60
High carbon steels	7.8	650-2000	1570	460	40
Low alloy steels	7.8	420-2000	1750	460	40
High alloy steels	7.8	460-1700	1680	500	12-30
Cast irons	7.4	10-800	1403	511	53
Superalloys	7.9	1300	1550	1550	11
Al alloy	2.63-2.82	80-150	476-654	880-920	121-239
Cu alloy	7.47-8.94	300-500	885-1260	377-435	29-234
Ni alloy	7.75-8.85	550-1500	1110-1454	381-544	12-63
Ti alloys	4.43-4.70	1000-1300	1549-1649	502-544	8-12

Accordingly, an example of the material groups and their Indexes is illustrated in Table 7.3.

Table 7.3 Material Indexes

Material	Index
Low-c steel & alloy	LS
High-c steel & alloy	HS
Tool steel	TS
High speed steel	SS
Superalloy	SA
Austenitic stainless	AS
Martensitic stainless	MS
Cast iron	CI
Al alloy	AA
Mg alloy	MA
Cu alloy	CA
Ni alloy	NA
Ti alloy	TA

Since it is not known how to modify a case for different materials, the material groups belong to the Very Important category. However, the grouping does not detail each kind of material. Differences are dealt with as follows. Material hardness is used to distinguish the grindability of a material within a material group. For more accurate matching in each material group, specific material names can be used as sub-indexes. The material sub-indexes belong to the Important category. For example, Index HS-1080 means a material classified as AISI 1080 which belongs in the High-C steel & alloy material group.

Material hardness Index

Small variations in material microstructure can make a large difference in grinding behaviour. Hardness is a primary indicator of machinability and therefore hardness is used to take account of microstructure [ASM 1989]. Material hardness also affects the grinding force and the selection of the wheel. The hardness is classified as soft(< 50Rc), medium(50-57Rc) and hard(>57Rc). The indexes are S, M and H. Since it is not know how to modify a case for the effect of material hardness, these indexes belong to the Very Important category.

Grinding wheel Index

Usually, a grinding wheel specification has five elements as outlined in Chapter 3. However, the most important factors affecting grindability are abrasive type, grit size and the bond type. Therefore, the grinding wheels are indexed according to abrasive type, grit size and bond type as illustrated in table 7.4. The wheel indexes belong to the Very Important category. If the grinding wheel is to be selected as part of the Solution, the wheel Indexes are set to 0.

Table 7.4 Indexes for grinding wheel

Abrasive type	Aluminium oxide	Silicon carbide	CBN								
Index	A	C	B								
Bond type	Resin		Vitrified		Metal			Rubber			Shellac
Index	B		V		M			R			E
Grit size	10	12	14	16	20	30	36	46	60	80	100 ...
Index	A	B	C	D	E	F	G	H	I	J	K ...

Workpiece roughness Index

Workpiece roughness is divided into 5 groups each associated with a particular wheel grit size. Values of Ra between 0.10 and 1.6 μm are represented by numbers from 1 to 5 and associated with the values as a sub-Index. Examples are illustrated in Table 7.5.

The main Indexes belong to the Very Important category because the necessary modification for different wheels is unknown. The sub-Indexes belong to the Important category because some grinding parameters can be modified to meet the variety. In the same roughness group, wheel grit size is unchanged. If the wheel has been specified, the wheel grit size implies a particular roughness group. In this situation, the main roughness group indexes will be set to 0.

Table 7.5 Example of indexes for roughness

Roughness (μm R_a)	Index
1.60	1-1.60
1.40	1-1.40
1.10	1-1.10
0.90	2-0.90
0.80	2-0.80
0.70	2-0.70
0.50	3-0.50
0.40	3-0.40
0.35	4-0.35
0.25	4-0.25
0.20	4-0.20
0.17	5-0.17
0.14	5-0.14
0.12	5-0.12
0.10	5-0.10

Other Indexes

The values of workpiece diameter are directly employed as Indexes. For example, the workpiece diameter Index 25 denotes that the workpiece initial diameter is 25mm. In a similar manner, the values of wheel diameter and wheel speed are directly employed as

Indexes. When wheel diameter and wheel speed are to be selected as part of the process of achieving the Solution, the Problem Index is initially set to 0. From equations 3.4, 3.10 and 3.11, workpiece diameter, wheel diameter and wheel speed affect the basic parameters. However, the values of these parameters can be modified to meet different requirements. Therefore, the workpiece diameter Index, wheel diameter Index and wheel speed Index belong to the Important class.

Other features which do not affect the selection of initial grinding conditions are classified as Unimportant. These features do not need an Index. Unimportant features for selection purpose include the roundness, size tolerance, width *etc*. These features may possibly be controlled in-process in some cases. For example, the grinding width affects the power and force. The grinding width can therefore be used to increase or decrease feedrate proportionally to ensure that the power does not exceed the power capacity of the machine. This is a feature of adaptive control. The Unimportant features may not be input into the system or be input into the system as a reference for adaptive control.

7.5 Information Retrieval

7.5.1 Use of a part number

When repeatedly machining specific parts of a type which have been ground before, it may be preferred to retain the previous grinding conditions. To retrieve the conditions and avoiding repeated input and reasoning, the simplest way is to use the unique identification provided by the part number. The user inputs the part number, and the system looks for the number in the casebase. If successful, the exact case is located, otherwise the system fails to produce an answer.

7.5.2 Using the Index

Selection of an appropriate case is achieved when the Problem Index is matched with a

case in the case base. When a user inputs the Problem, the system identifies the features of the Problem and creates the Problem Index. The system then matches the Index of a Problem with the Index of a case which consists of two steps:

• The system matches the Very Important Indexes. The Very Important Indexes of the Problem and of the case in the case library must be same. If the match is successful for one or more cases, a set of applicable cases has been located. The system then proceeds to the next step, otherwise, the system fails to complete the reasoning process and quits the process.

• A nearest case is retrieved from the set of applicable cases through matching where possible the Important Indexes. The nearest case is decided based on the similarity metric, described below.

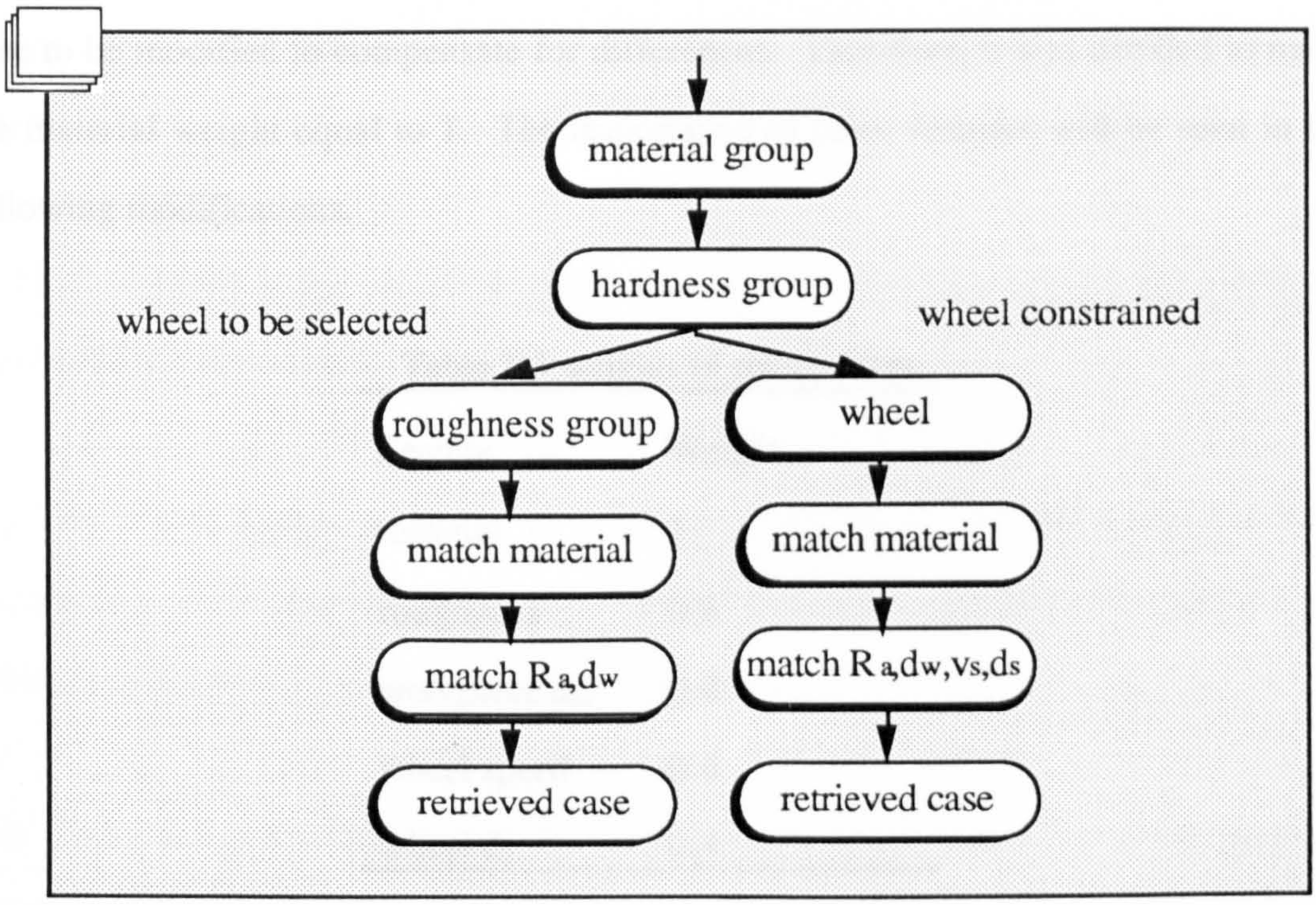

Figure 7.5 Retrieving a case

Depending on whether a grinding wheel is specified or whether the system is to select a suitable wheel, the retrieval process will match different indexes as illustrated in Figure 7.5.

A similarity metric is proposed to judge the similarity between a new case as defined by the Problem and an old case located by the Very Important Indexes. The nearest case is the case with the highest similarity value. The similarity metric is defined as :

$$similarity = \frac{\sum_{i=1}^{n} weight_i \times sim_i}{\sum_{i=1}^{n} weight_i} \qquad 7.1$$

where, $weight_i$ denotes the importance of each feature and the complexity of the modification. The more important the feature is or the more difficult the modification, the higher is the value of the weight. The values of the weights for the important features are listed in Table 7.3. Different materials in the same material group will have different properties. Unfortunately, there is insufficient available information to allow a case to be modified to compensate for differences. Therefore, it was decided to make the material weight equal to 1. The importance of other features will be seen in the following modifications.

Table 7.3 weights of the features

Feature	Weight
material	1
roughness	0.6
workpiece d_w	0.4
wheel speed	0.4
wheel d_s	0.4

Sim_i denotes the similarity of the *ith* feature of two cases. For the material feature, sim(material) is defined as: if the materials are the same, the sim(material)=1, otherwise, sim(material)=0. For other features, sim_i is defined as:

$$sim_i = 1 - \left(\frac{Problem_i - Case_i}{variable\ range}\right)^2 \qquad 7.2$$

where

$Problem_i$ = *the value of ith feature in Problem*

$Case_i$ = *the value of ith feature in existing case*

variable range = *variable range of the value of ith feature*

7.6 Modification of a Case

In most situations, the case retrieved will not exactly fit the problem definition. The case must therefore be modified to conform to the new requirements. There are ten independent variables to be determined as outlined in Chapter 3. Since the Problem case is almost similar to a stored Solution, it is expected that the stored case needs only a minor modification to apply to the Problem. The strategy for modification of a case is based on the following assumptions:

(i) Since similar cases belong to the same material and roughness groups, the wheel and fluid need not be changed.

(ii) The wheel speed, v_s, either has a fixed value or an optimal value has been established, which is not affected by the variations of other variables. Therefore, wheel speed does not need modification.

(iii) There are many factors which affect surface roughness. The most sensitive parameter is dressing lead f_d. Dressing lead is therefore modified by the system to satisfy the required surface roughness and other parameters are unchanged.

Based on the above assumptions, the feedrate, v_f, the workspeed, v_w, and the dressing lead, f_d, there are three parameters which require modification in case-based reasoning.

For changes in work diameter or wheel speed and wheel diameter, the following is an explanation of the process of modification.

In Section 3.4.3, four kinds of basic parameters were discussed, which correlate fairly well with the main grinding parameters. The parameters are equivalent chip thickness h_{eq}, mean grinding chip volume V_m, the mean chip aspect ratio A_r and the ratio between the equivalent chip thickness and the geometric contact length h_{eq}/lg. The values, v_f and v_w are modified based on these parameters.

To maintain optimal conditions, the equivalent chip thickness h_{eq}, mean grinding chip volume V_m, the mean chip aspect ratio A_r and the ratio between the equivalent chip thickness and the geometric contact length h_{eq}/lg should all be unchanged. In a similar grinding process, the wheel is unchanged and the values of parameters need only a minor modification. Therefore, the dynamic grit spacing l_d and the grinding chip width b_c may be assumed to be unchanged. For changes in workpiece diameter from d_{wo} to d_{wn} or wheel speed from v_{so} to v_{sn} the feedrate is changed from v_{fo} to v_{fn}.

From equation 3.10

$$V_m = h_{eq} l_d b_c = \frac{\pi d_{wo} l_d b_c v_{fo}}{v_{so}} = \frac{\pi d_{wn} l_d b_c v_{fn}}{v_{sn}} \qquad 7.3$$

and therefore

$$v_{fn} = v_{fo} \frac{d_{wo}\, v_{sn}}{d_{wn}\, v_{so}} \qquad 7.4$$

For changes in equivalent diameter from d_{eo} to d_{en} or wheel speed from v_{so} to v_{sn}, the work speed is changed from v_{wo} to v_{wn}.

From equation 3.11

$$A_r = \frac{l_c}{h_m} = \frac{0.5 d_{eo} v_{so}}{l_d v_{wo}} = \frac{0.5 d_{en} v_{sn}}{l_d v_{wn}} \qquad 7.5$$

and therefore

$$v_{wn} = v_{wo} \frac{d_{en}\, v_{sn}}{d_{eo}\, v_{so}} \qquad 7.6$$

where $d_e = \frac{d_s d_w}{d_s + d_w}$

By a similar procedure, from equation 3.4 and 3.12 the same results as equations 7.4 and 7.6 can be obtained, which provides support for the modification strategy.

For a different surface roughness, dressing depth a_d and equivalent chip thickness h_{eq} are unchanged, a change of surface roughness from R_{ao} to R_{an} will require a change in dressing lead from f_{do} to f_{dn}:

From equation 3.3,

$$f_{dn} = \frac{f_{do} R_{an}^2}{R_{ao}^2} \qquad 7.7$$

The process of selecting a value of dressing feed is not an exact process due to the uncertainty introduced by the shape of the dressing diamond which will change with the wear mentioned in Section 3.4.3. However, with a little modification the method is feasible.

The modification of the dressing parameters is difficult so that the weight of the roughness is taken to be 0.6. From equations 7.4 and 7.6, the modifications caused by workpiece diameter d_w, wheel diameter d_s and wheel speed v_s are relatively more reliable

than the modifications caused by roughness. Taking both reliability and sensitivity into account, the weights are assigned a value of 0.4.

7.7 Evaluation of the Solution

The Solution should be evaluated and if necessary modified further. New Solutions are stored in the case base and form new cases. This process provides the learning capability of the case-based reasoning system. Two approaches to implementation are proposed, depending on whether the system is used with or without adaptive control.

(i) Learning from an Adaptive Control Optimisation system

Figure 7.6 illustrates a framework combining case-based reasoning and adaptive control optimisation.

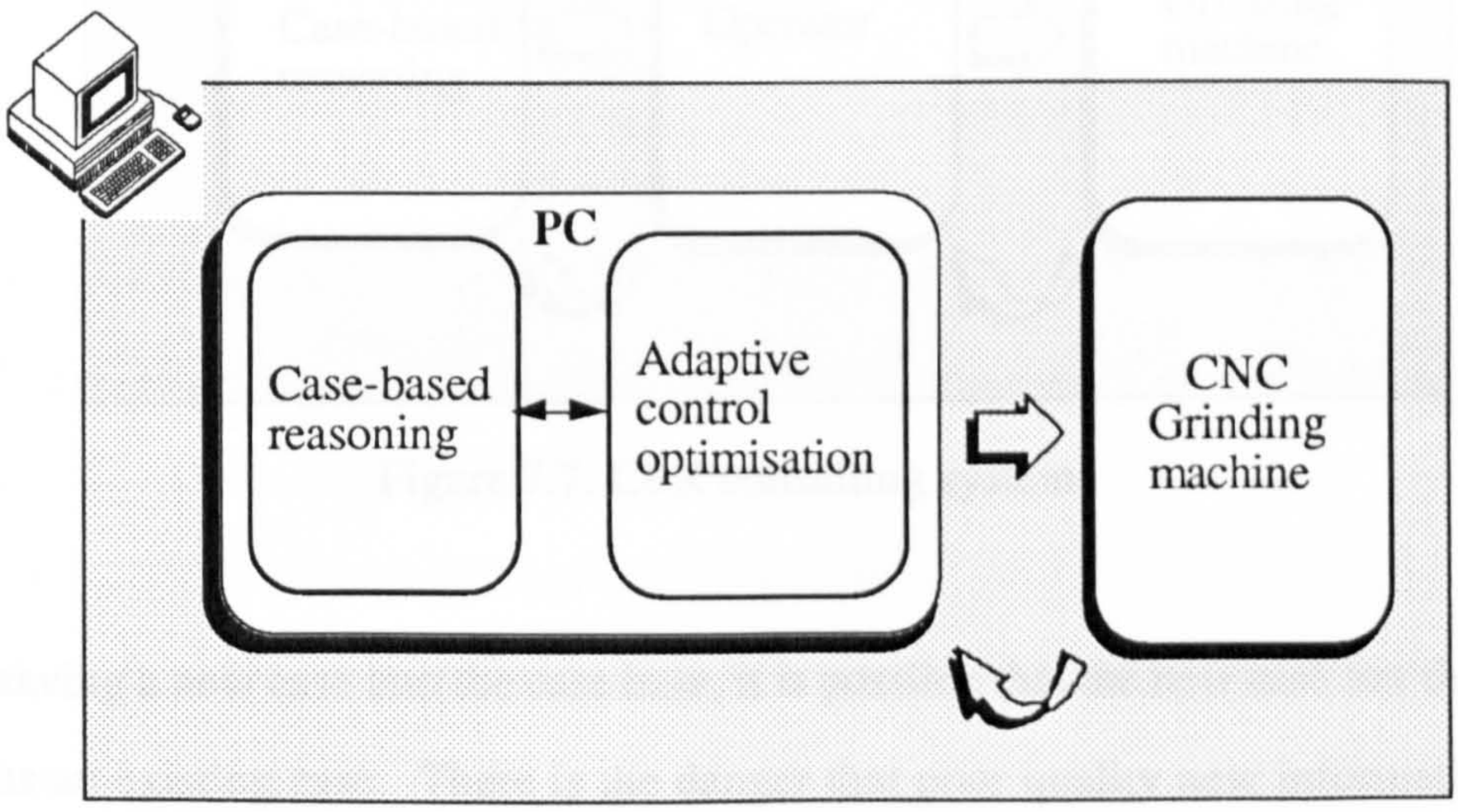

Figure 7.6. Combination of CBR and ACO

Case-based reasoning is an off-line reasoning approach while adaptive control is an on-line and real-time control system. Adaptive control adapts the values of parameters to take account of the changes in the operating environment in order to maintain optimal grinding conditions. Moreover, if the values produced by case-based reasoning are close to optimal, adaptive control adapts the values to optimal for the particular grinding wheel condition and learns the values. An adaptive control system can therefore provide

feedback of learned values to a case-based reasoning system which stores the new case for future reference. The approach is based on the system proposed by [Rowe 1991][Rowe 1995]. The system adapts and optimises the parameters subject to the limitations of burn, power and roughness.

(ii) Learning from the Operator

The case-based reasoning system can also be used to recommend grinding conditions to the operator as illustrated in Figure 7.7. When satisfactory values have been achieved in practice, the operator stores the Solution in the case-based reasoning system.

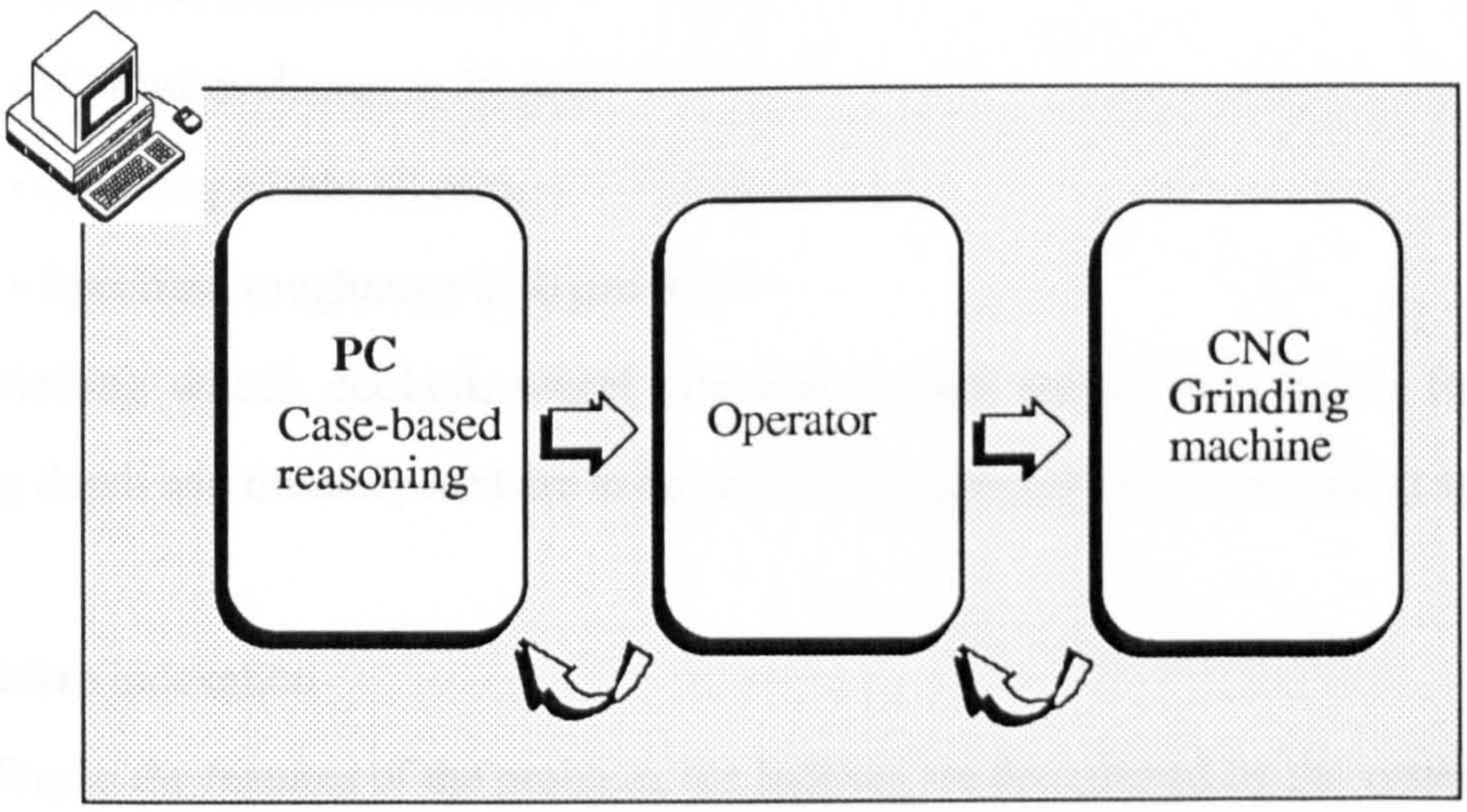

Figure 7.7. CBR consulting system

When saving a new case into the case base, it is possible that the new case has the same Index as an existing case. There is the danger that poor quality new information can over-write good quality old information. Therefore, when the new case is saved into the case base, the system compares the cases to determine which is better. The basic for comparison is the specific removal rate:

$$Q_w^{'} = \pi d_w v_f$$

Since d_w is the same in the two cases, v_f can be used for comparison. The case having the higher v_f is retained in the case base.

7.8 An Example

The following example illustrates how the system works.

(i) Problem description

The user inputs the following information:

- External cylindrical plunge grinding with single-point dressing
- Workpiece material: High carbon steel & alloy group, AISI 1080
- Material hardness: 62 Rc.
- Workpiece diameter: 25 mm
- Grinding width: 30 mm.
- Specified roughness: 0.40 μm Ra.

The Grinding wheel, coolant, wheel diameter, wheel speed, workspeed, feedrate, dressing depth and dressing feed are to be selected by the case-based reasoning system.

(ii) Problem indexation

According to the features of the problem, the Indexes are determined by the system as

HS, H, 0, 3, - 1080, 0.40, 25, 0, 0.

where, HS identifies high-carbon steel, H.....high hardness, 0 ..... wheel to be selected, 3.....roughness in the third group. The former features belong to the Very Important category.

1080.....material, 0.40 (Ra) ..... roughness, 25 (mm) ..... work diameter, 0.....wheel diameter to be selected, 0.....wheel speed to be selected. These latter features belong to the Important category.

(iii) Case retrieval

The system searches a set of cases with the same Very Important identifiers, then, uses the similarity algorithm to retrieve a nearest case.

The case index is:

- HS, H, AVK, 3, - 1060, 0.45, 16, 365, 33,

The Problem description of the nearest case retrieved is therefore:

- Workpiece material: high-carbon steel group, AISI 1060 and high hardness
- Dressing tool: single point diamond
- d_w = 16 mm, R_a = 0.45 mm, b = 24 mm

The corresponding solution of the case retrieved is:

- wheel: WA60JV,
- coolant: water based
- wheel diameter: d_s = 365 mm,
- wheel speed: v_s = 33 m/s,
- work speed: v_w = 0.25 m/s,
- feedrate: v_f = 0.02 mm/s,
- dressing lead: f_d = 0.15 mm/s,
- dressing depth: a_d = 0.015 mm

(iv) The Solution

The Solution is then achieved by modifying the nearest case retrieved from the case base. Using the kinematic relationships, v_f is modified to 0.013 mm/s, v_w is modified to 0.38 m/s and f_d is modified to 0.12 mm/s. The values of the parameters for the solution are illustrated in Table 7.4, and can be compared with the values for the nearest case.

v) Learning and storing the new case

The initial values of the grinding parameters are presented to the operator either for use by the ACO system or for direct use by the operator. The learned values together with the problem description constitute a new case which is stored in the case base. These values may be modified by the operator after testing, where the operator is allowed this discretion.

Table 7.4 Case study

	Problem	Similar case	Solution
material	high-c steel,AISI1080	high-c steel,AISI1060	high-c steel,AISI1080
wheel		WA80JV	WA80JV
coolant		water based	water based
hardness	62 Rc	60 Rc	62 Rc
d_w	25 mm	16 mm	25 mm
b	30 mm	24 mm	30 mm
R_a	0.40 um	0.45 um	0.40 um
d_s		365 mm	365 mm
v_s		33 m/s	33 m/s
v_w		0.25 m/s	0.38 m/s
v_f		0.02 mm/s	0.013 mm/s
f_d		0.15 mm/r	0.12 mm/r
a_d		0.015 mm	0.015 mm
V.Imp. Indexes	HS,H,0,3	HS,H,AVK,3	HS,H,AVK,3
Imp Indexes	1080,0.40,25,0,0	1060,0.45,16,365,33	1080,0.40,25,365,33
Part No.			

Chapter 8 The Selection of Grinding Conditions Using Rule Based Reasoning

Case based reasoning is the preferred method for selection of grinding conditions. However, when the number of grinding cases are insufficient to cover the whole grinding problem space, rule base reasoning is a useful approach to provide a set of starting conditions. The approach is employed as one of the intelligent agents for selection of grinding conditions.

8.1 The Architecture of Rule Based Reasoning for Selection of Grinding Conditions

The basic architecture of the rule based reasoning agent for selection of grinding conditions is illustrated in Figure 8.1, which includes:

- A knowledge base containing a set of production rules
- An inference engine
- A memory containing the current state description

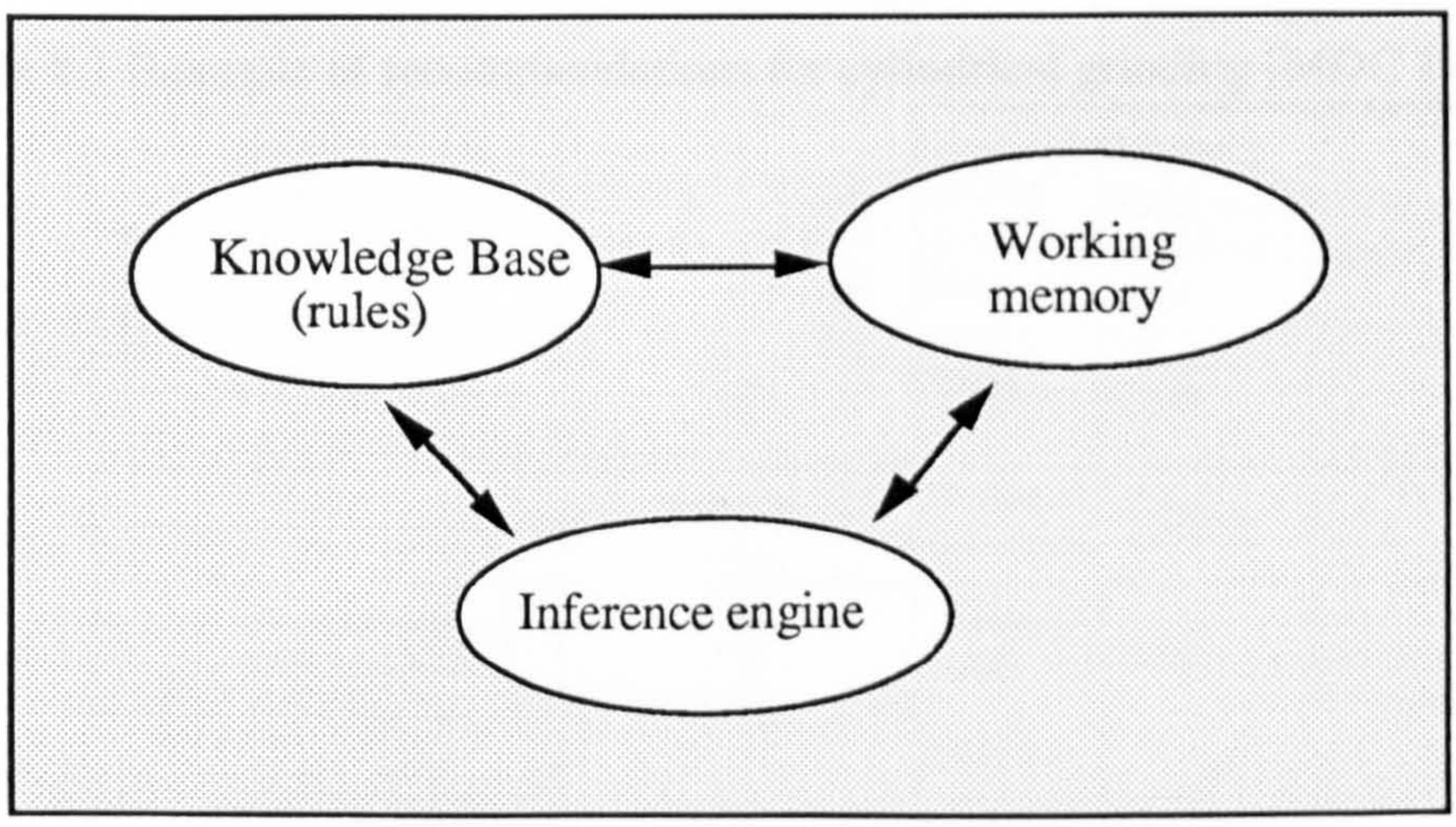

Figure 8.1. Main components of the rule based reasoning agent

The flow of information involved in the development of the rule based reasoning system is illustrated in Figure 8.2.

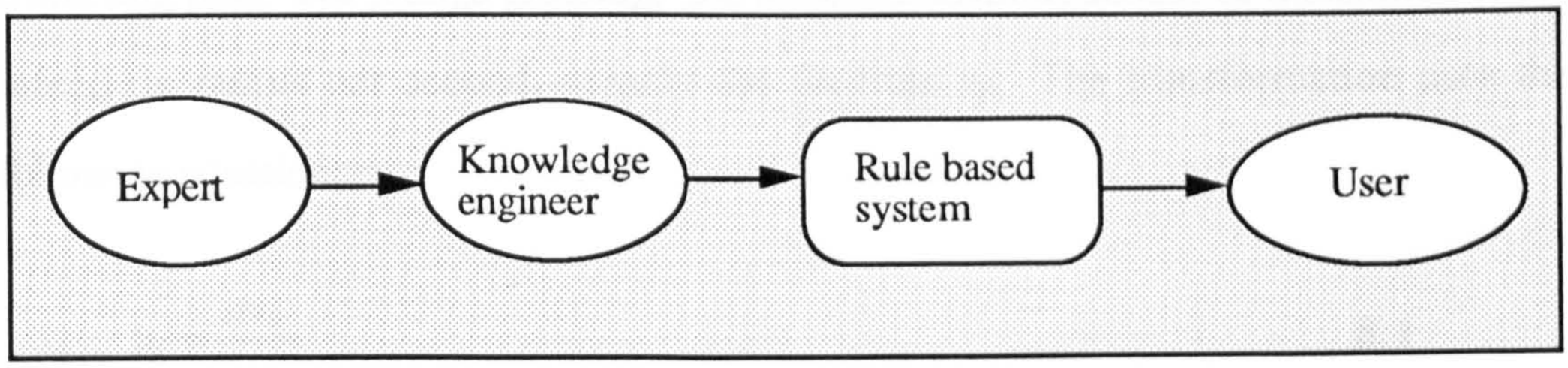

Figure 8.2 The information flow for developmentof the rule based system

8.2 Rule Base for Grinding Conditions

The knowledge base stores information about the grinding domain. It contains symbolic representation of expert rules based on judgement and experience.

The basic knowledge for selection of grinding conditions was collected and refined from The Machining Data Handbook [MDC 1980] and The Grinding Data Book[Universal 1992]. In these knowledge sources, knowledge mainly exists in databases or as some basic principles. An example of recommendations for cylindrical grinding with conventional wheels is illustrated in Table 8.1.

Table 8.1 Example of recommendations for cylindrical grinding [MDC 1980]

The data in the table can be encoded into rules. The roughness $R_a < 0.8$ μm is assumed for finish grinding, otherwise the operation is assumed to be rough grinding. The infeed values in the table are for traverse grinding, namely the infeed on diameter per

traversing pass. For plunge grinding, the recommendations should be transferred to the infeed on radius per second, namely the feedrate v_f. The transformation uses the following equation:

$$v_f = \frac{a v_w}{\pi d_w} \qquad 8.1$$

where a is equal to half the infeed on diameter per traverse.

Accordingly, one rule is:

IF	workpiece material is general carbon and alloy steels
AND	material hardness < 50 Rc
AND	wheel is conventional wheel
AND	roughness < 0.8 Ra
THEN	wheel speed = 28-33 m/s
AND	workspeed = 0.35-0.5 m/s
AND	feedrate $\leq 0.013/2 \times 0.35 \times 1000/(\pi \times d_w)$

In the rule, wheel specification is not included. The wheel selection in Table 8.1 is imprecise because the effects of the roughness requirements and the operation severity were not taken into account in the wheel recommendation. The neural network for wheel selection described in Chapter 5 can provide a better result.

The rules for dressing values are derived from Table 8.2.

Accordingly, two rules are:

IF roughness ≥ 0.8 μm

THEN dressing depth = 0.025 mm

AND dressing lead = 0.18 mm/rev.

IF roughness < 0.8 μm

THEN dressing depth = 0.012-0.019 mm

AND dressing lead = 0.10 mm/rev.

3 Oils-heavy duty

4 Emulsifiable oils - light duty (general purpose)

5 Emulsifiable oils - heavy duty

6 Chemicals and synthetics - light duty (general purpose)

7 Chemicals and synthetics - heavy duty

where number 3 is oil-based, others are water-based.

An example of a rule for coolant is:

IF workpiece material is general carbon and alloy steels

AND wheel is conventional wheel

AND material hardness < 48 Rc

THEN coolant = Emulsifiable oils - light duty /

Chemicals and synthetics - light duty

The three kinds of rules are expressed as follows:

IF workpiece material is general carbon and alloy steels

AND material hardness < 50 Rc

AND wheel is conventional wheel

AND roughness R_a< 0.8 μm

THEN wheel speed = 28-33 m/s

AND workspeed = 0.35-0.5 m/s

AND feedrate ≤ $0.013/2 \times 0.35 \times 1000/(\pi \times d_w)$

AND dressing depth = 0.012-0.019 mm

AND dressing lead = 0.1 mm/r

AND coolant = Emulsifiable oils - light duty /

Chemicals and synthetics - light duty

According to the above methods, a set of rules was developed. Some examples of the rules in the set are illustrated in the following.

Rule 5

IF workpiece material is tool steel

AND material hardness > 50 Rc

AND wheel is conventional wheel

AND roughness < 0.8 μm

THEN wheel speed = 28-30 m/s

AND workspeed = 0.3-0.5 m/s

AND feedrate ≤ $0.01/2 \times 0.3 \times 1000/(\pi \times d_w)$

AND dressing depth = 0.012-0.019 mm

AND dressing lead = 0.1mm/r

AND coolant = Emulsifiable oils - heavy duty /

Chemicals and synthetics - heavy duty

Rule 12

IF workpiece material is cost iron

AND material hardness < 50 Rc

AND wheel is conventional wheel

AND roughness < 0.8 μm

THEN wheel speed = 28-33 m/s

AND workspeed = 0.35-0.5 m/s

AND feedrate $\leq 0.025/2 \times 0.3 \times 1000/(\pi \times d_w)$

AND dressing depth = 0.012-0.019 mm

AND dressing lead = 0.1mm/r

AND coolant = Emulsifiable oils - light duty /

Chemicals and synthetics - light duty

Rule 13

IF workpiece material is cost iron

AND material hardness > 50 Rc

AND wheel is conventional wheel

AND roughness < 0.8 μm

THEN wheel speed = 28-33 m/s

AND workspeed = 0.35-0.5 m/s

AND feedrate $\leq 0.013/2 \times 0.35 \times 1000/(\pi \times d_w)$

AND dressing depth = 0.012-0.019 mm

AND dressing lead = 0.1mm/r

AND coolant = Emulsifiable oils - heavy duty /

Chemicals and synthetics - heavy duty

Rule 20

IF	workpiece material is superalloys
AND	wheel is conventional wheel
AND	roughness < 0.8 μm
THEN	wheel speed = 15-18 m/s
AND	workspeed = 0.25-0.5 m/s
AND	feedrate ≤ $0.005/2 \times 0.25 \times 1000/(\pi \times d_w)$
AND	dressing depth = 0.012-0.019 mm
AND	dressing lead = 0.1mm/r
AND	coolant = oils-heavy

Rule 34

IF	workpiece material is aluminum alloys
AND	wheel is conventional wheel
AND	roughness < 0.8 μm
THEN	wheel speed = 28-33 m/s
AND	workspeed = 0.25-0.77 m/s
AND	feedrate ≤ $0.013/2 \times 0.25 \times 1000/(\pi \times d_w)$
AND	dressing depth = 0.012-0.019 mm
AND	dressing lead = 0.1mm/r
AND	coolant = oils-light duty

8.3 Inference Engine

The inference engine provides the reasoning strategy for searching the knowledge base to determine which rules apply to the situation and makes the appropriate decision. Two forms of reasoning strategy are forward chaining and backward chaining[Adeli 1990]. In forward chaining, the system starts with known facts and searches to determine which rules are satisfied by the facts. Conclusions are reached on the basis of the rules which are true, and the search continues until a solution or set of solutions is found.

Backward chaining starts with a solution and works backwards to determine whether the required antecedents are satisfied. Backward chaining is commonly used for condition monitoring and problem diagnosis whereas forward chaining is more often used for planning and selecting values of process parameters to be employed. Some systems employ a combination of these techniques[Rowe 1994a].

The approach adopted for selection of grinding conditions was the forward chaining reasoning. The basic reasoning method is a pattern-matching algorithm. In a predetermined order, the condition portions of rules are compared with the current state of facts. When all the conditions of a rule are satisfied, then that rule becomes eligible for execution. Once a rule is executed, the conclusion part is added to the working memory.

For example, the following facts are the working memory.

Fact 1: workpiece material is tool steel

Fact 2: material hardness > 50 Rc

Fact 3: wheel is conventional wheel

Fact 4: roughness = 0.4 μm

Fact 5: workpiece diameter = 30 mm

Rule 5 can be executed because fact 1, fact 2, fact 3 and fact 4 match its LHS. As a result, and according to fact 5, the following facts are added to the working memory:

wheel speed = 28-30 m/s

workspeed = 0.3-0.5 m/s

feedrate ≤ 0.026 mm/s

dressing depth =0.012 - 0.019 mm

dressing lead =0.10 mm/n

coolant = Emulsifiable oils - heavy duty /

Chemicals and synthetics - heavy duty

Since there are no other rules to be executed, the process stops. Thus the above facts are the conclusions of the reasoning process.

Chapter 9 The Development of a Multi-Agent System for Selection of Grinding Conditions

9.1 Introduction

In Chapter 4, a multi-agent strategy for selection of grinding conditions was proposed. A concept framework of the system was proposed. The principle and structure of each agent, as an independent system, was described in Chapters 6, 7 and 8. This chapter deals with the implementation of the whole system by merging the agents using a blackboard model.

9.2 The System Structure and Principle

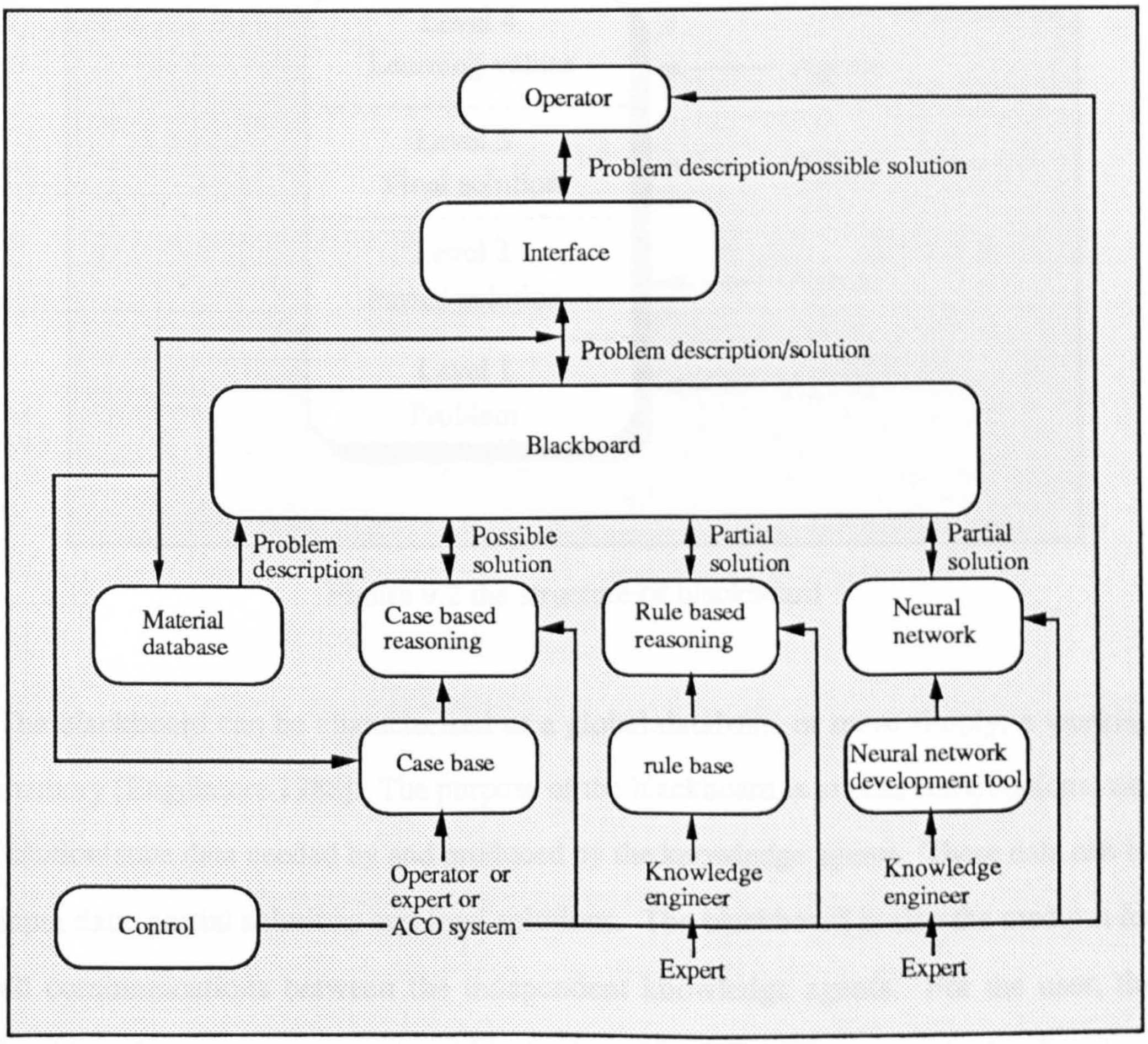

Figure 9.1 The structure

Figure 9.1 illustrates the structure of the multi-agent system modelled using the blackboard approach. The model consists of three basic parts, knowledge agents, blackboard and control unit.

9.2.1 The knowledge agents

The knowledge agents were described in previous Chapters.

9.2.2 The blackboard

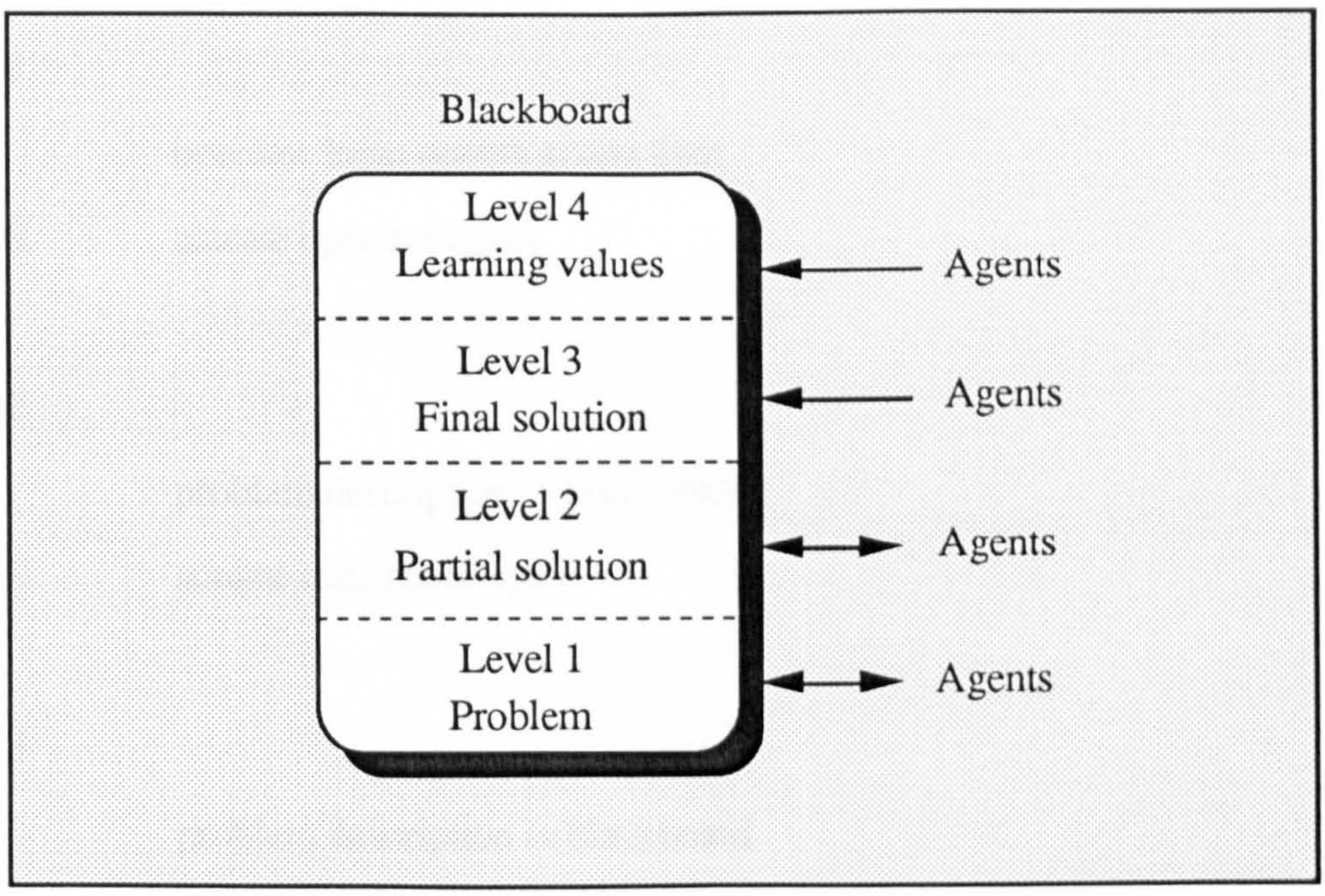

Figure 9.2 the structure of blackboard

The blackboard can be characterised as a global database, or more simply, a working memory [Engelmore 1988]. The purpose of the blackboard is to hold computational and solution-state data needed by and produced by the knowledge agents. These data can be input data, partial solutions and final solutions. The blackboard is also the medium for all communications between the independent knowledge agents. For the user, the blackboard is a computer screen which displays the input and output and other functions

are transparent. The blackboard has a hierarchical structure which consists of four level working memories as illustrated in figure 9.2

9.2.3 The control mechanism

The knowledge agents respond opportunistically to changes on the blackboard. The control mechanism is to monitor the changes on the blackboard and decide which knowledge agent to actuate next [Engelmore 1988]. In the proposed system, the control itself is a knowledge agent. The control mechanism uses a set of control rules to solve the problem of 'what next'. The rules are as follows:

Rule 1

IF	operator input occurs at any time
THEN	actuate operator agent

Rule 2

IF	problem description in blackboard
THEN	actuate case based agent

Rule 3

IF	problem description in blackboard
AND	no solution in blackboard after case based agent actuated
AND	wheel is not in blackboard
THEN	actuate neural network agent

Rule 4

IF	problem description in blackboard
AND	no solution in blackboard after case based agent actuated
AND	wheel is in blackboard
THEN	actuate rule based agent

Accordingly, the system control flowchart is illustrated in Figure 9.3

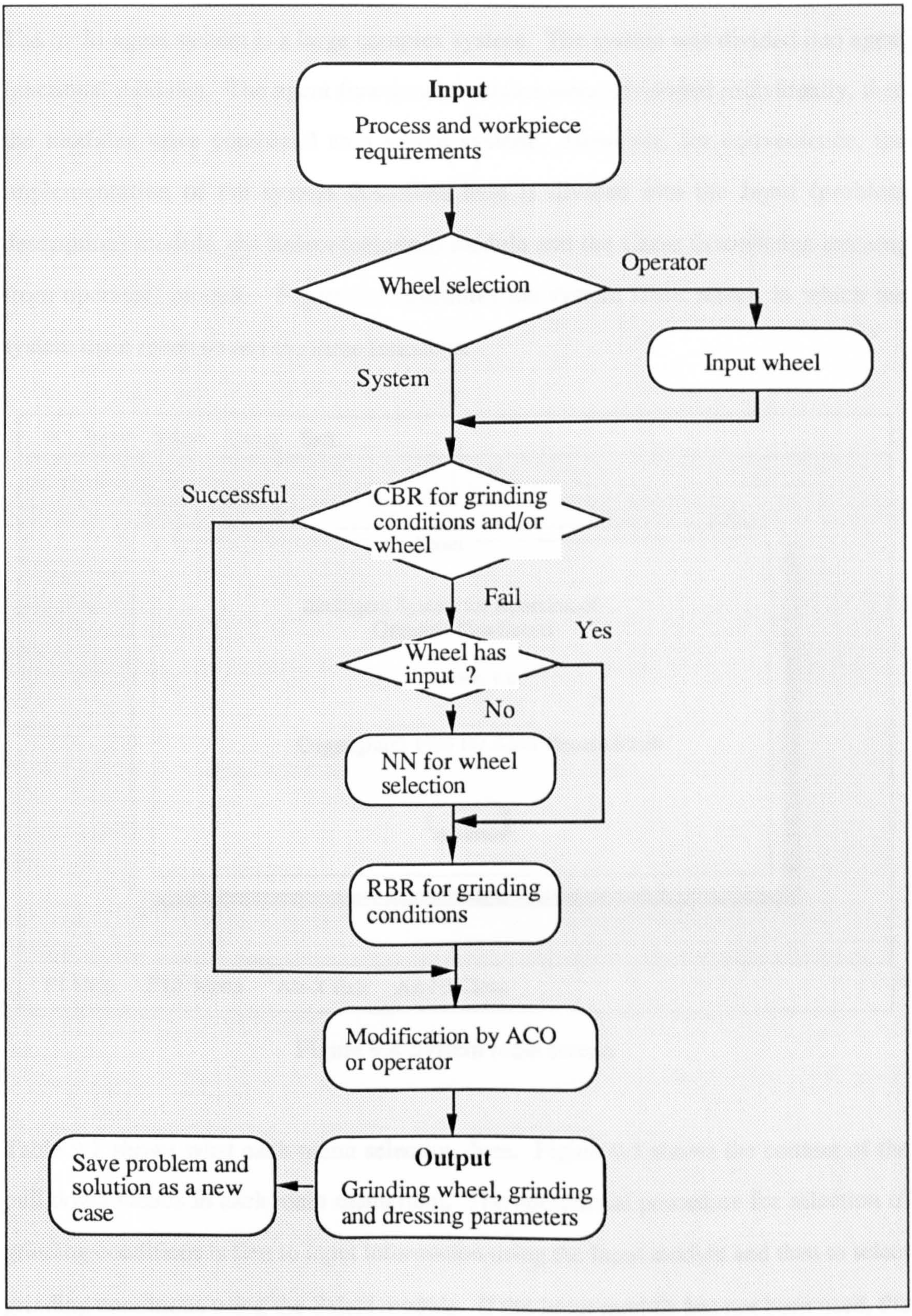

Figure 9.3 The control flowchart

9.3 The Implementation of the Multi-agent System

The multi-agent system is a large complex system. The system was divided into agent functional modules. The agent functional modules were developed individually, then the modules were combined as a whole system. However, for convenience, the implementation of the system described here is divided into the Input (problem description) module, the Select (solution) module and the Cases (knowledge learning from operator) module. Figure 9.4 illustrates the system main screen in which the system main menu shows the three functions.

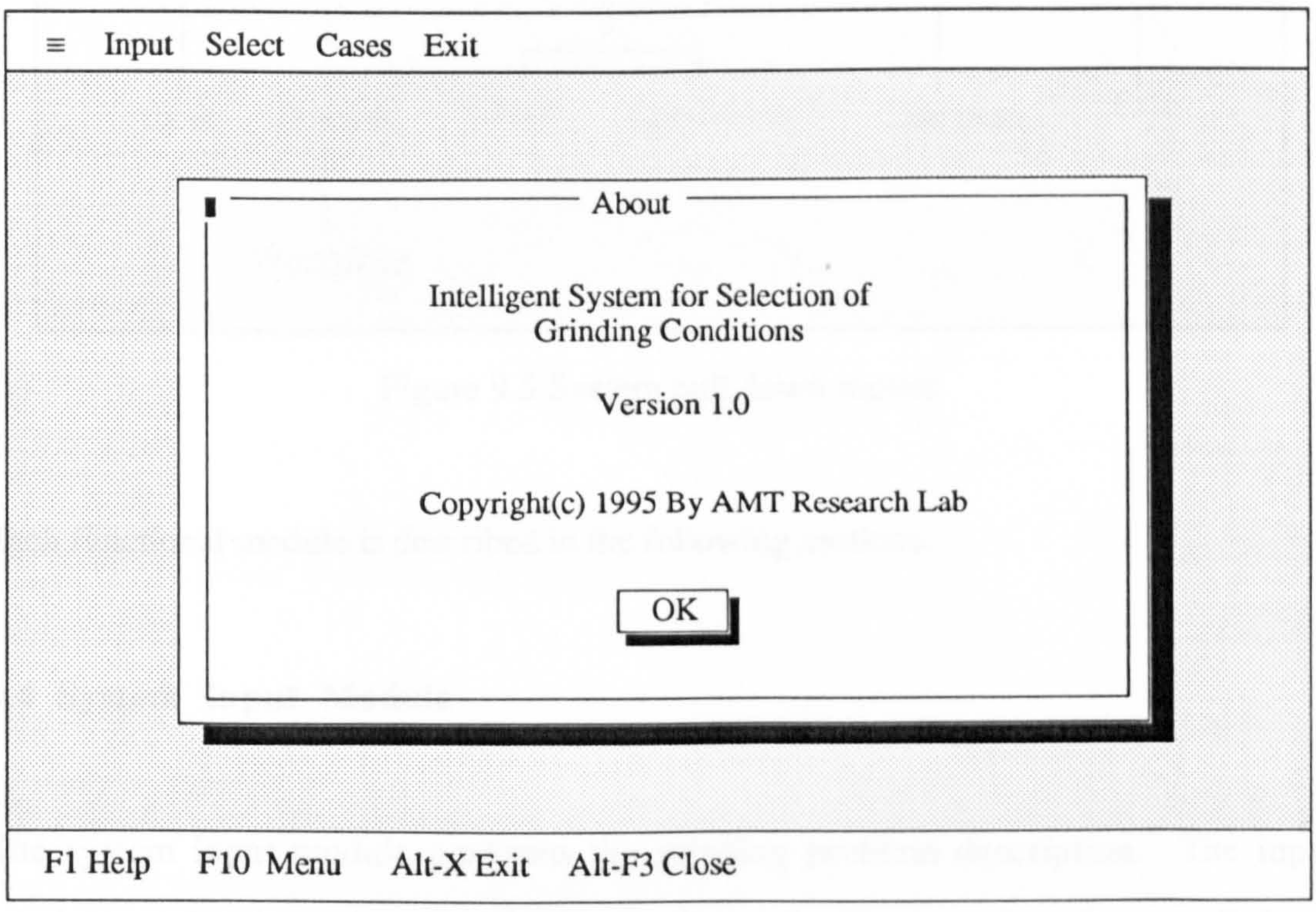

Figure 9.4 System main screen

Table 9.1 shows what each menu selection does. Figure 9.5 shows the content of the pull down menus in each main menu item. The operational procedure for selection of grinding conditions is first to input information using the Input module and then to select grinding conditions using the Select module. If the Input module has not been used, the Select module cannot be actuated.

Table 9.1 System main menu

Item	Purpose
≡	About the system
Input	Input process and workpiece information
Select	Select wheel and grinding and dressing parameters
Cases	Browse and edit the case base
Exit	Exits the system

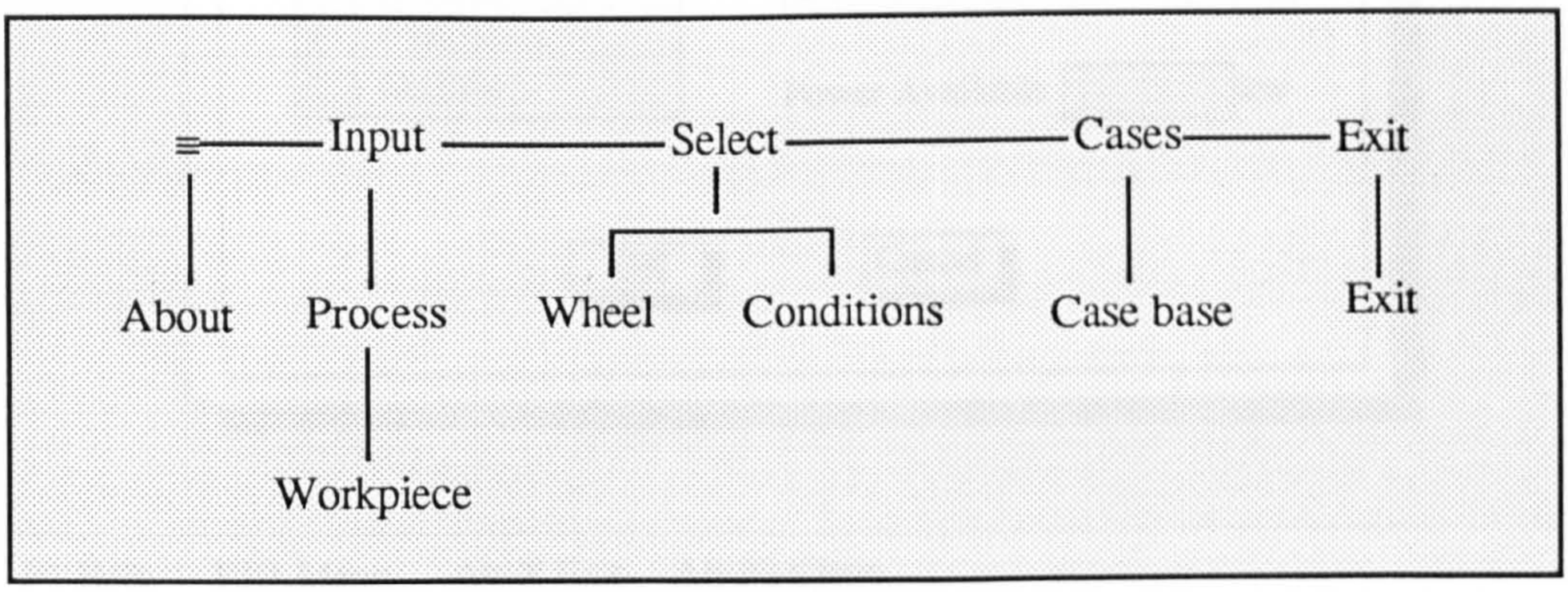

Figure 9.5 System pull down menus

Each functional module is described in the following sections.

9.4 System Input Module

The system input module performs the grinding problem description. The input requirement has been described in Chapter 3 and 7. The input item has a pull down menu which has two parts, process information input and workpiece information input. The two parts are separately completed in two input dialogue boxes.

9.4.1 Process information input

Process information describes the machine tool and the grinding method. Figure 9.6 shows the dialogue box for process information input. To date, the developed system

only works for external plunge grinding. However, if the system is developed further it will include other process types. Information on the power available is useful for an adaptive control system.

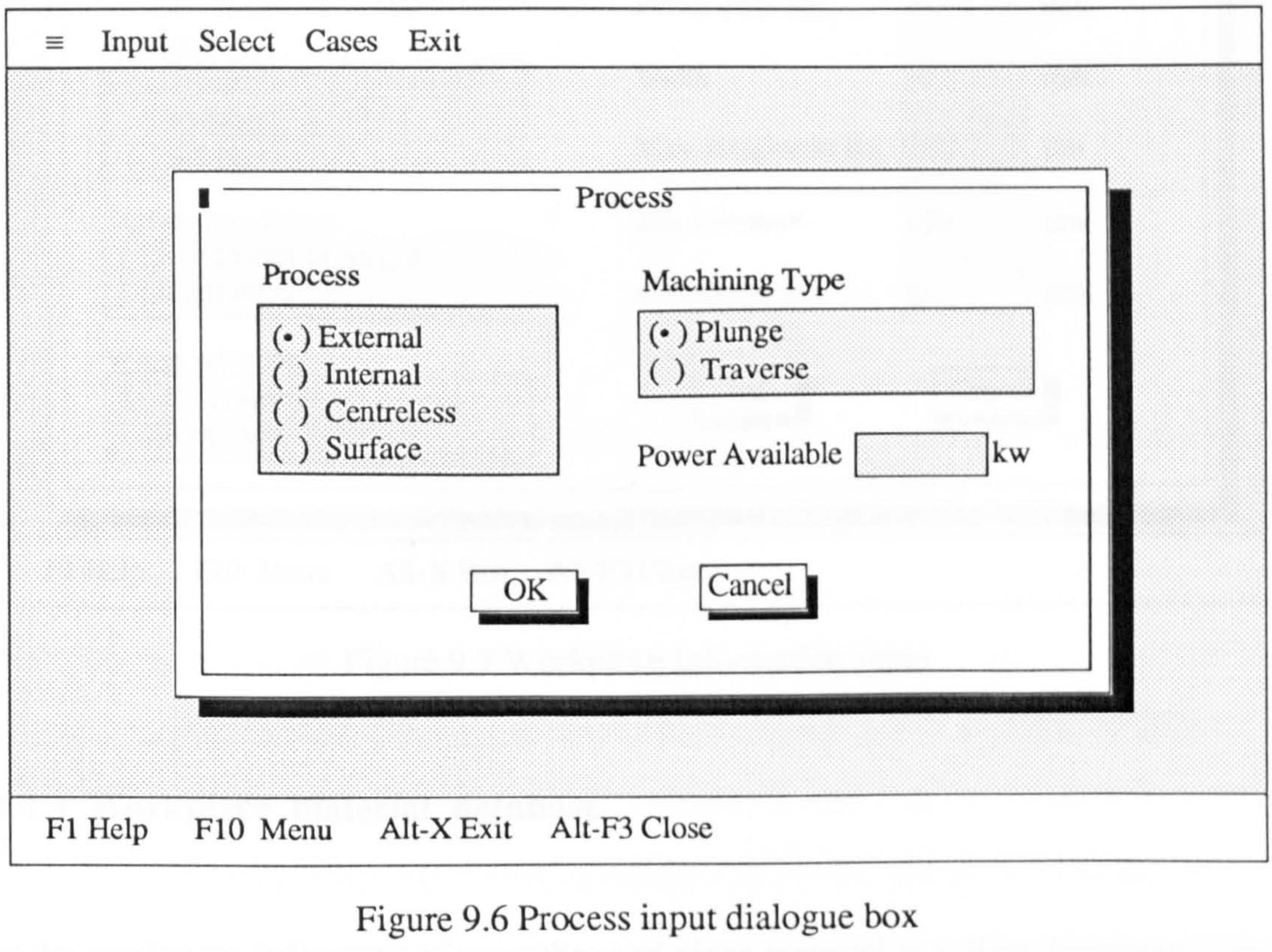

Figure 9.6 Process input dialogue box

9.4.2 Workpiece information input

The workpiece information inputs are illustrated in Figure 9.7. Some of the inputs are not essential for the Problem description but are useful for reference such as the workpiece width, size tolerance and roundness tolerance. Therefore, they are not required but are expected to be entered by the operator.

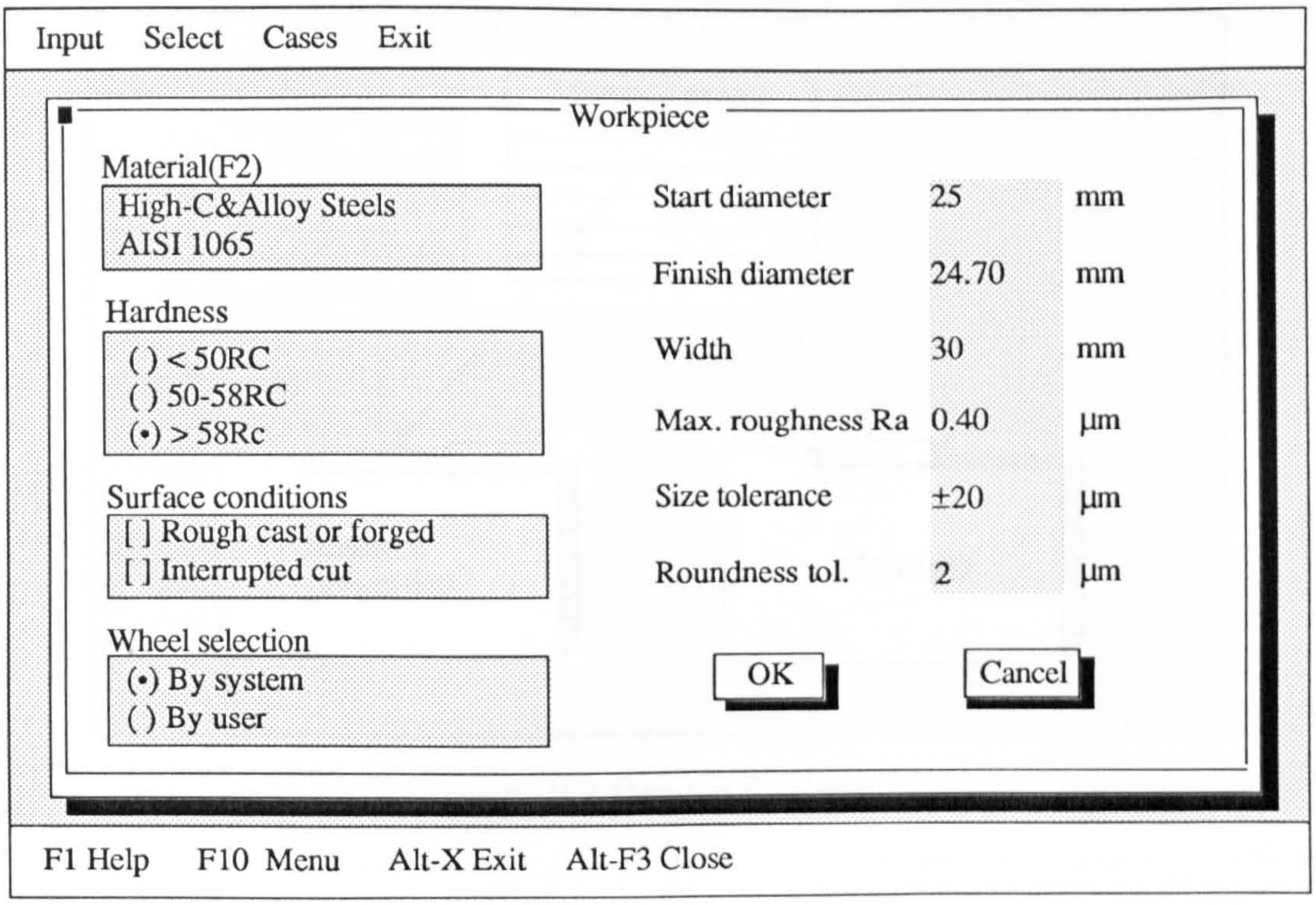

Figure 9.7 Workpiece information input

9.4.3 Workpiece material database

In the workpiece information input, the workpiece material is a Very Important input. There are many materials used in grinding. A database is therefore designed to manage the selection of materials.

Database structure

According to Chapter 7, materials are grouped according to their properties and individual material names are used to associate the materials in a group. For the convenience of the user, the material group, material properties and individual material names are arranged into an hierarchical database. Figure 9.8 shows the relationship between the material group, the material properties and individual materials.

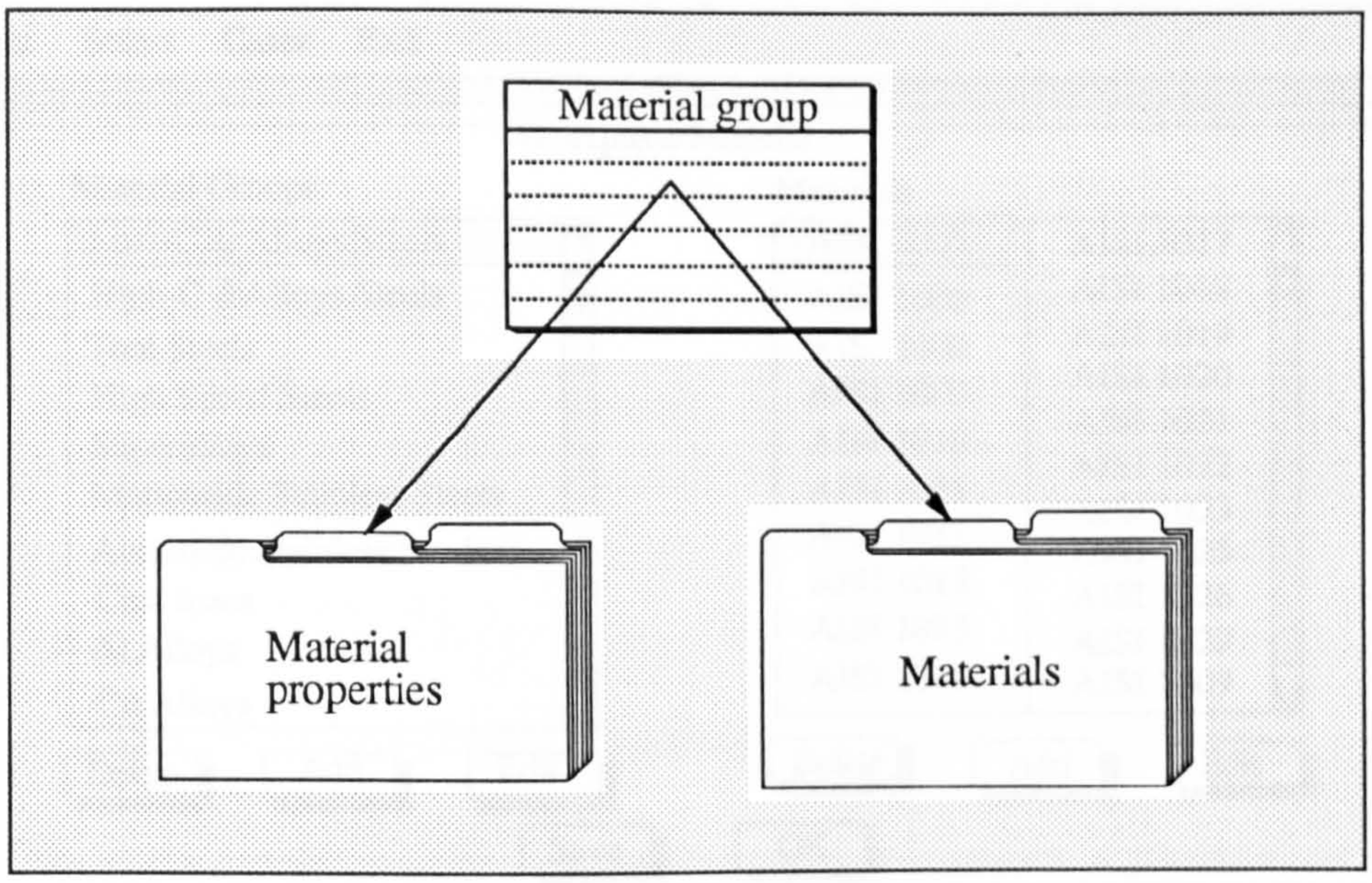

Figure 9.8 Material database

The material group includes the material catalogue illustrated in Table 7.3. The material properties element includes a set of data: Index, group name, thermal conductivity, specific heat, critical temperature, melting temperature and density. Material properties are employed by the adaptive control system[Rowe 1991]. Each material group has a set of properties. Individual materials are those materials belonging to the same material group.

The workpiece material input is actually a process of selection of a material from the database. Firstly, the material group is selected and the corresponding individual material names are displayed. Then, the individual material is selected. Figure 9.9 illustrates the selection screen. To the left of the picture is the material group. To the right of the picture are the individual materials corresponding to a selected material group. Two views are displayed using listboxes with scrollers. The user can use the mouse or a key to scroll the list in the listboxes. In the materials listbox, double clicking an item will select the material and its corresponding material group. Pressing the OK button will also select the highlighted items.

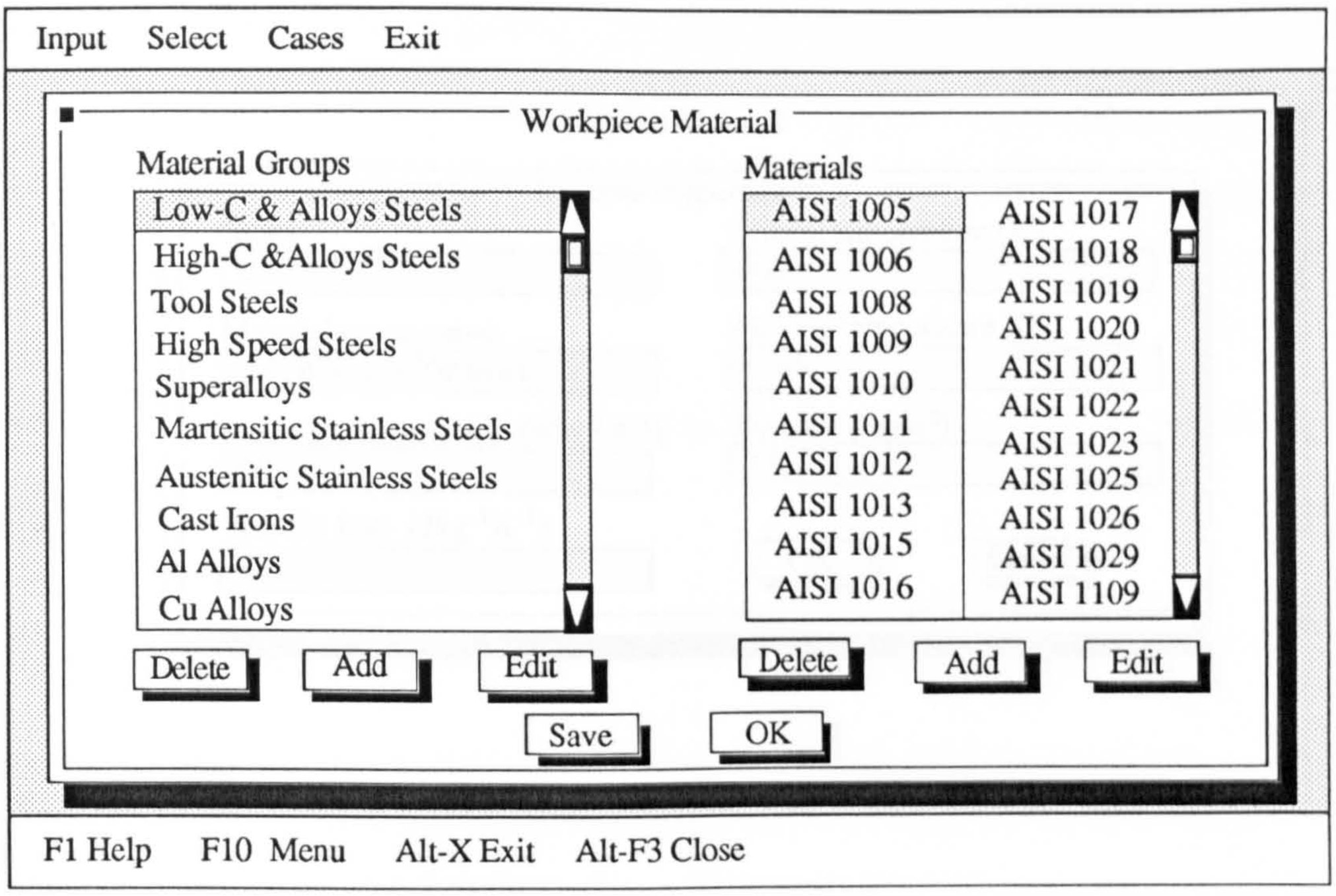

Figure 9.9 The selection of material

Database management

The database can be up-dated, by deleting from, adding to or editing the contents.

(i) Editing the material group and its properties

Normally, the material group and the material properties are modified simultaneously.

Therefore, the modification uses an edit interface as illustrated in figure 9.10.

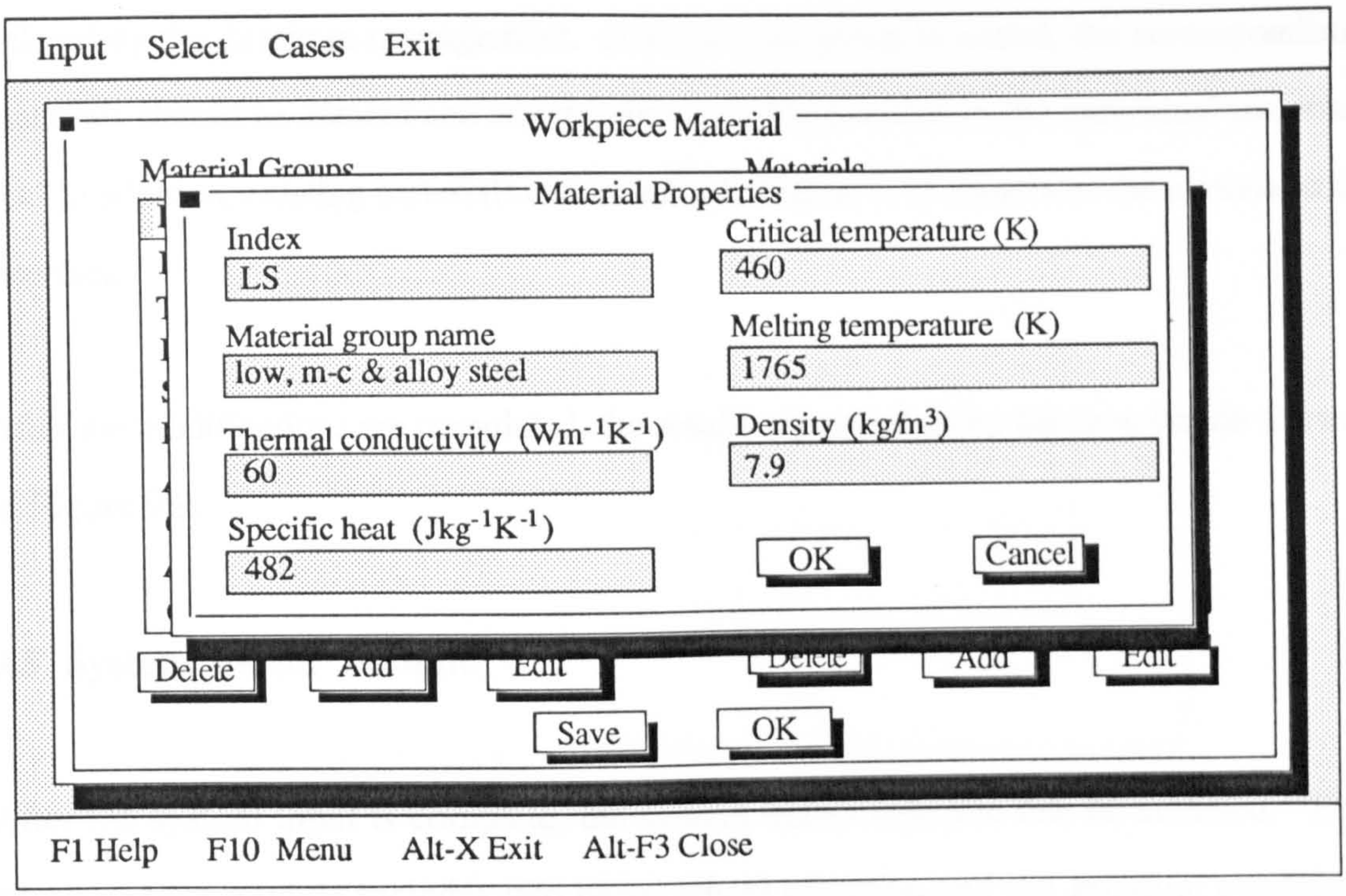

Figure 9.10 Material group and properties edit interface

(ii) Editing the individual material

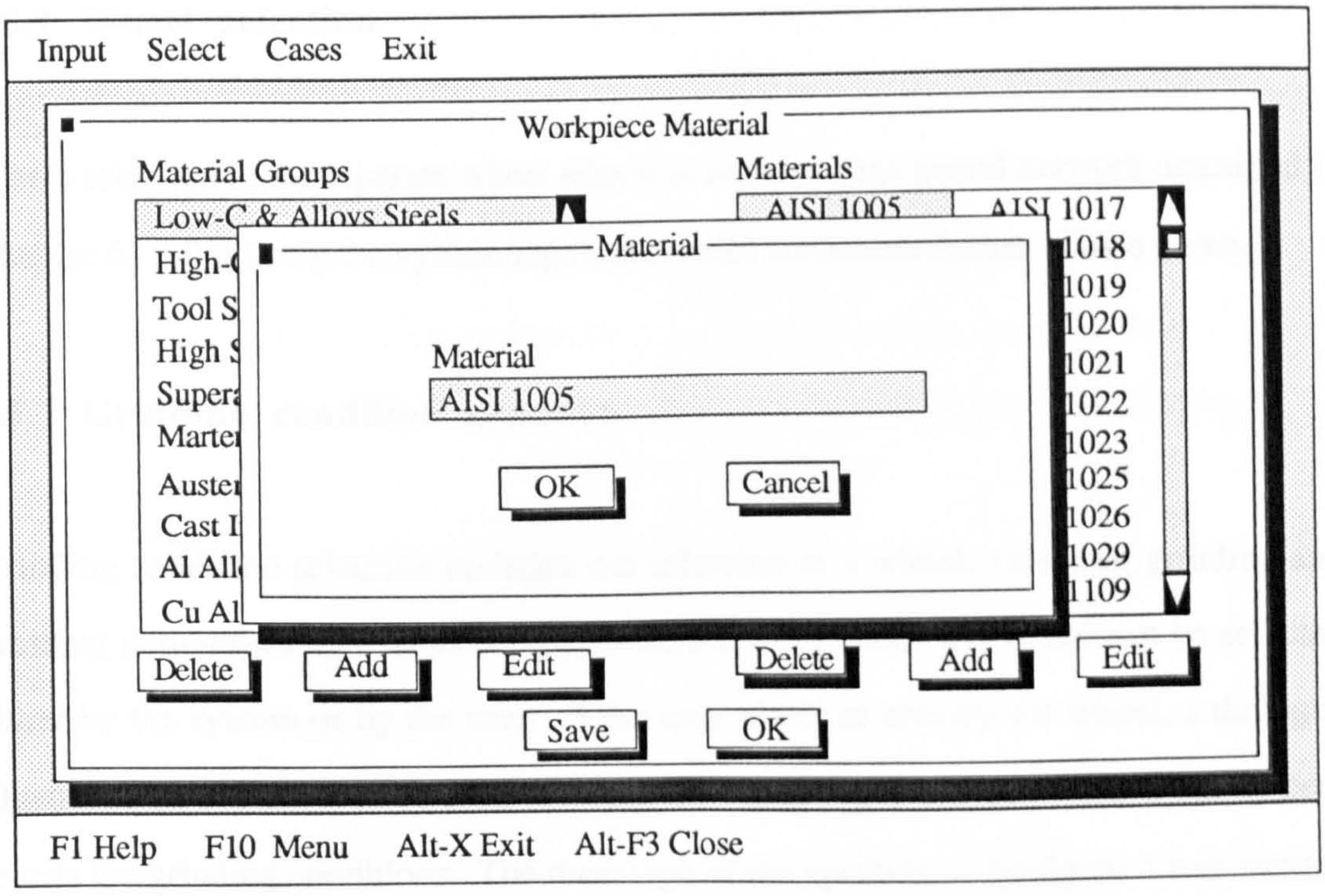

Figure 9.11 Material edit interface

If a material group name is deleted, the set of corresponding materials are automatically

deleted by the database management. If a material group is added, the corresponding materials should be created and inserted in a suitable position in the individual material set. In addition, existing materials can be edited. Figure 9.11 illustrates the material edit interface.

After the modifications are completed, the results are saved using the save button shown in Figure 9.9.

9.5 System Select Module

After the system Input is complete, the system Select function can be actuated. The system Select module includes two parts: wheel selection and the grinding condition selection. The grinding condition selection function includes the wheel selection. However, if the user wants only to select a wheel, the wheel selection item can be used.

9.5.1 Wheel selection

Wheel selection in the separate wheel selection item uses the neural network described in Chapter 6. After using the system input, the wheel recommendation will be given.

9.5.2 Grinding condition selection

Grinding condition selection includes the selection of a wheel, values of grinding and dressing parameters as well as coolant selection. However, the wheel can be selected either by the system or by the user. If the user wants to specify the wheel, a dialogue illustrated in Figure 9.12 is used to input the wheel specification before the system selects the grinding conditions. The meanings of the symbols in the figure are illustrated in Table 7.4.

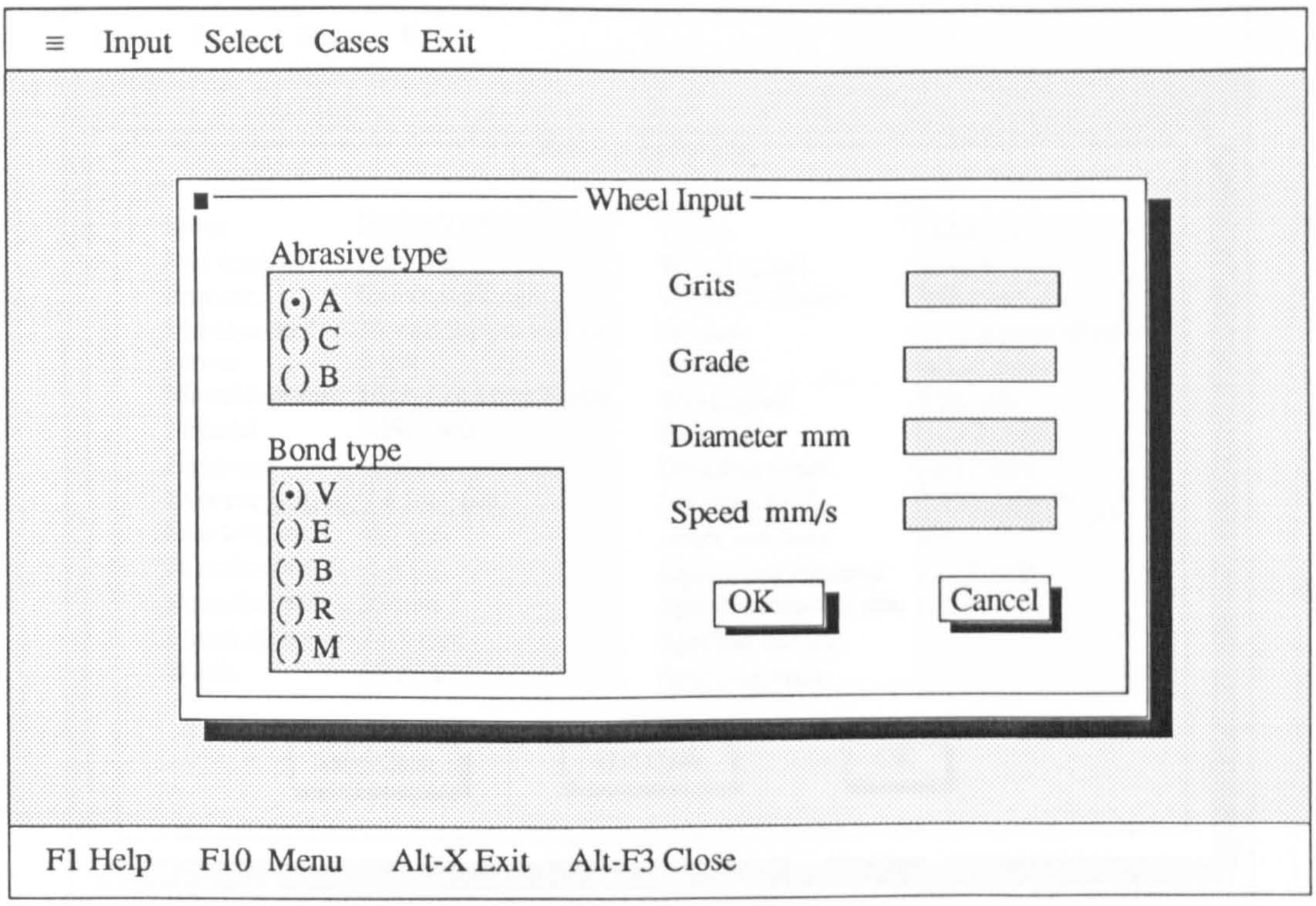

Figure 9.12 Operator input wheel

Following the inputs previously described, the system will actuate the case based reasoning agent. If the case based reasoning is successful, the recommendation of grinding conditions will be given as illustrated in Figure 9.13. If the user wants to have a look at the similar case from which the recommendation was derived, he can use the Old Case button to get the old case illustrated in Figure 9.14. The recommended grinding conditions can be modified by the user using the keyboard. If the recommended conditions are satisfied, they can be saved as a new case through the "Save Case" button.

If the case base reasoning agent fails to give the solution, the system automatically actuates the rule based reasoning agent and the neural network agent to give the recommendation as illustrated in Figure 9.15. However, if the user has specified the wheel, the system adopts the wheel specified by the user instead of by the neural network. The rule based reasoning only gives a range of the values of the operating parameters. After the user obtains satisfactory values, the values can be saved as a new case through the "Save Case" button.

Input Select Cases Exit

Recommendations

Index	HSHAVL3	Wheel	WA80JV
Part number	AMT 10	Wheel speed	33 m/s
Process	External plunge	Wheel diameter	365 mm
Machine tool	Jones&Shipman S10	Dresser	Single point diamond
Power	3 kw	Coolant	Water based
Material group	High-C&Alloy Steels	Workspeed	0.38 m/s
Material	AISI1060	Feedrate	0.013 mm/s
Hardness	62 Rc	Dressing depth	0.015 mm
Max roughness	0.4 Ra μm	Dressing lead	0.12 mm/rev
Size tolerance	-32 μm	Spark-out time	4 s
Roundness tol.	3.5 μm	Equivalant diameter	27.72 mm
Start diameter	25 mm	Specific removal rate	1 mm^2/s
Finish diameter	24.8 mm	Specific energy	
Width	30 mm	Grinding ratio	

Save Case Old Case OK

F1 Help F10 Menu Alt-X Exit Alt-F3 Close

Figure 9.13 Sample recommendations from the case based reasoning agent

Input Select Cases Exit

Old Case

Index	HSHAVL3	Wheel	WA80JV
Part number	AMT 05	Wheel speed	33 m/s
Process	External plunge	Wheel diameter	365 mm
Machine tool	Jones&Shipman S10	Dresser	Single point diamond
Power	3 kw	Coolant	Water based
Material group	High-C&Alloy Steels	Workspeed	0.25 m/s
Material	AISI1060	Feedrate	0.02 mm/s
Hardness	62 Rc	Dressing depth	0.015 mm
Max roughness	0.45 Raμm	Dressing lead	0.15 mm/rev
Size tolerance	±12 μm	Spark-out time	4 s
Roundness tol.	1.5 μm	Equivalant diameter	15.33 mm
Start diameter	16 mm	Specific removal rate	1 mm^2/s
Finish diameter	15.70 mm	Specific energy	60 J/mm^3
Width	24 mm	Grinding ratio	42

OK

F1 Help F10 Menu Alt-X Exit Alt-F3 Close

Figure 9.14 Old case

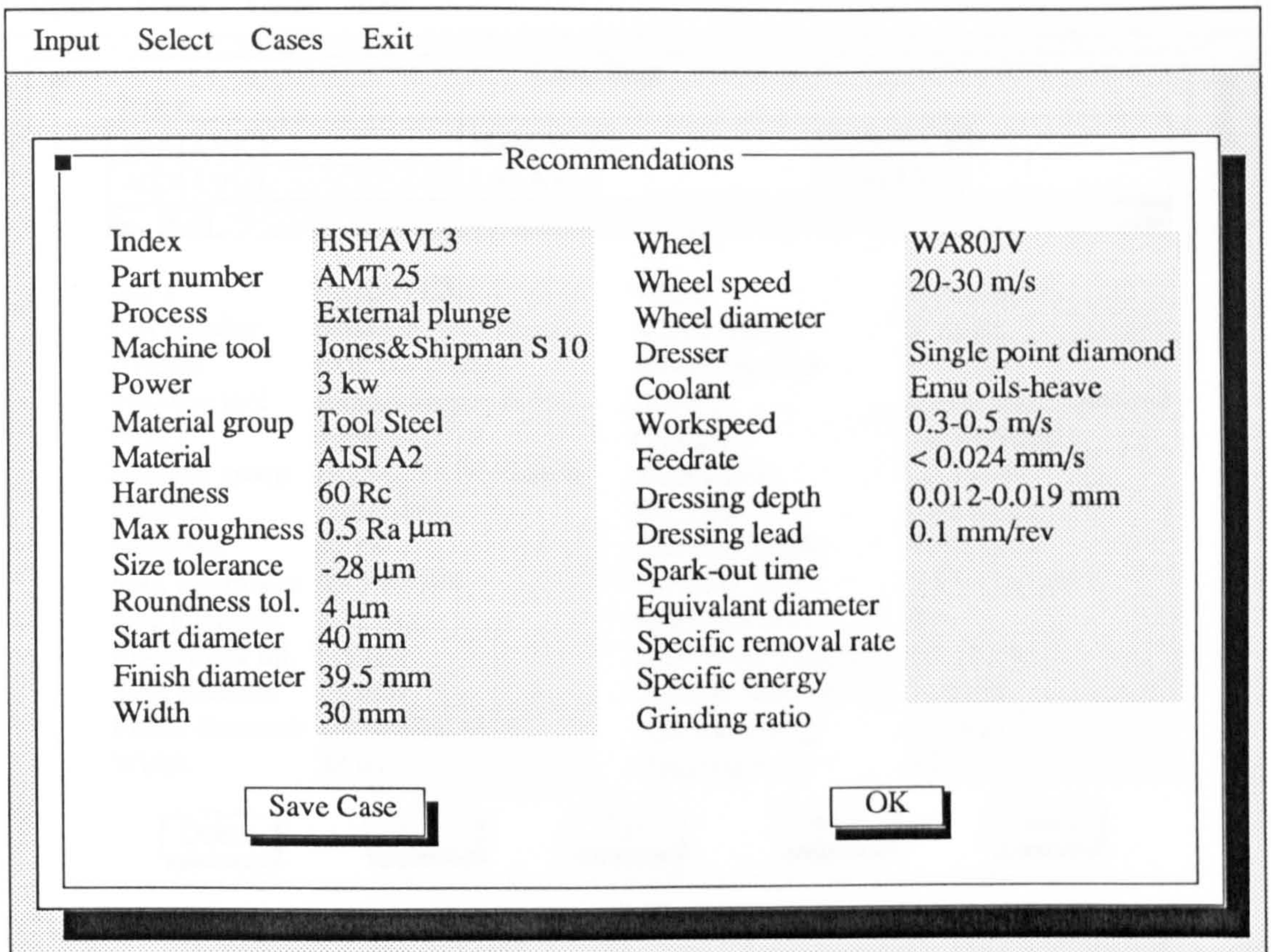

Figure 9.15 Sample recommendations from the rule based reasoning agent

9.6 Case Base Management

The multi-agent system learning is completed by individual knowledge agents. Neural network learning requires retraining using the neural network development tool and the procedure is completed by a knowledge engineer. Rule based reasoning learning requires the rules in the rule base to be edited and this is completed through the cooperation of experts and a knowledge engineer. Case based reasoning learning takes place by editing the cases in the case base. It can be achieved by an expert or by an operator without requiring the intervention of a knowledge engineer. Moreover, if the system is connected with an adaptive control CNC system the adaptive values can be fed back to the case base.

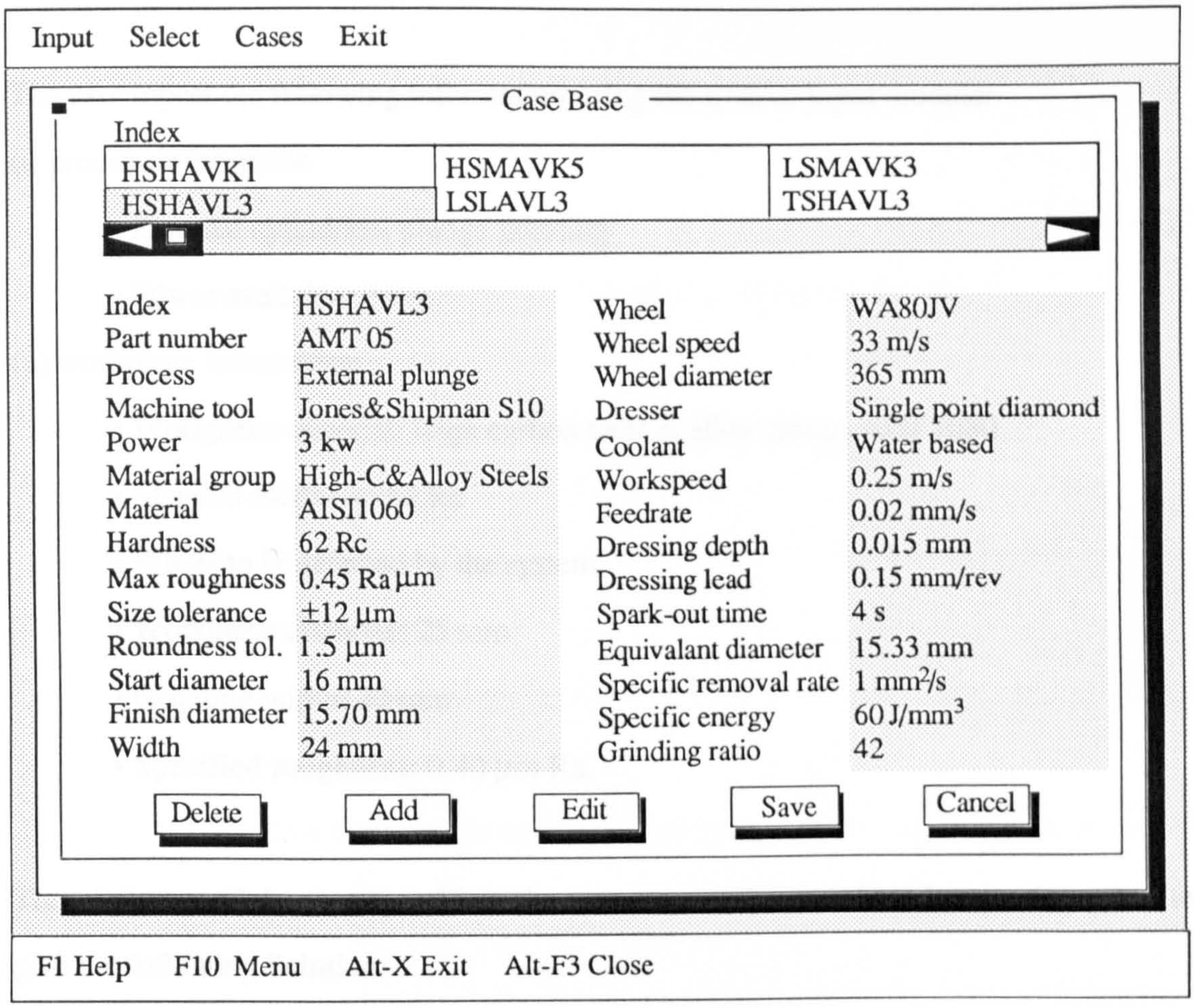

Figure 9.16 Case base

Figure 9.16 is a user interface of the case base. The user can edit a case directly through the interface and can browse through all the cases by means of an index search. The cases are sorted by the Indexes in alphabetical order. The indexes displayed in the top of the figure are restricted to Very Important Indexes. The user can delete, add and edit a case and save the case. Since these cases form an important body of knowledge on which the system relies, a password security system is built-in to prevent unauthorised editing.

9.7 Application Example

This section shows a comprehensive example which illustrates the operating procedure of the system.

The user inputs the following information using the system Input module:

(i) process information

- External cylindrical plunge grinding
- Power available: 3 kw

(ii) workpiece information

- Workpiece material: High carbon steel & alloy group, AISI 1080
- Material hardness: 62 Rc
- Wheel to be selected by the system
- Workpiece diameter: 25 mm
- Grinding width: 30 mm
- Specified roughness: 0.40 μm Ra.

Then Select module operates. First, the system uses the case based reasoning agent to give the following Solution:

- wheel: WA80JV,
- coolant: water based
- wheel diameter: d_s = 365 mm,
- wheel speed: v_s = 33 m/s,
- work speed: v_w = 0.38 m/s,
- feedrate: v_f = 0.013 mm/s,
- dressing lead: f_d = 0.12 mm/s,
- dressing depth: a_d = 0.015 mm

The results are derived from a similar case, which was illustrated in Table 7.4. If the similar case had not existed, other agents would have been actuated. For example, the neural network agent gives the wheel as:

WA80JV

Alternatives:

WA80KV

WA80IV

WA100JV

WA60JV

Then the rule based reasoning agent gives the other grinding conditions:

wheel speed = 28-33 m/s

workspeed = 0.35-0.5 m/s

feedrate ≤ 0.029 mm/s

dressing depth =0.012-0.019 mm

dressing lead =0.1 mm/rev

coolant = Emulsifiable-heavy

After the operator uses these results and modifies them if applicable, the solution can be saved as a new case.

Chapter 10 The Evaluation of the Strategy and the System

This chapter describes the initial evaluation of the system, in relation to:

(1) Grinding conditions recommended

(2) Use of the system and further development

10.1 Consideration of Evaluation Techniques

(i) Experimental evaluation

It was considered that although the most reliable approach to testing and validation of the selection system would be by experiment, this approach was not realistic. Figure 10.1 illustrates a possible experimental approach.

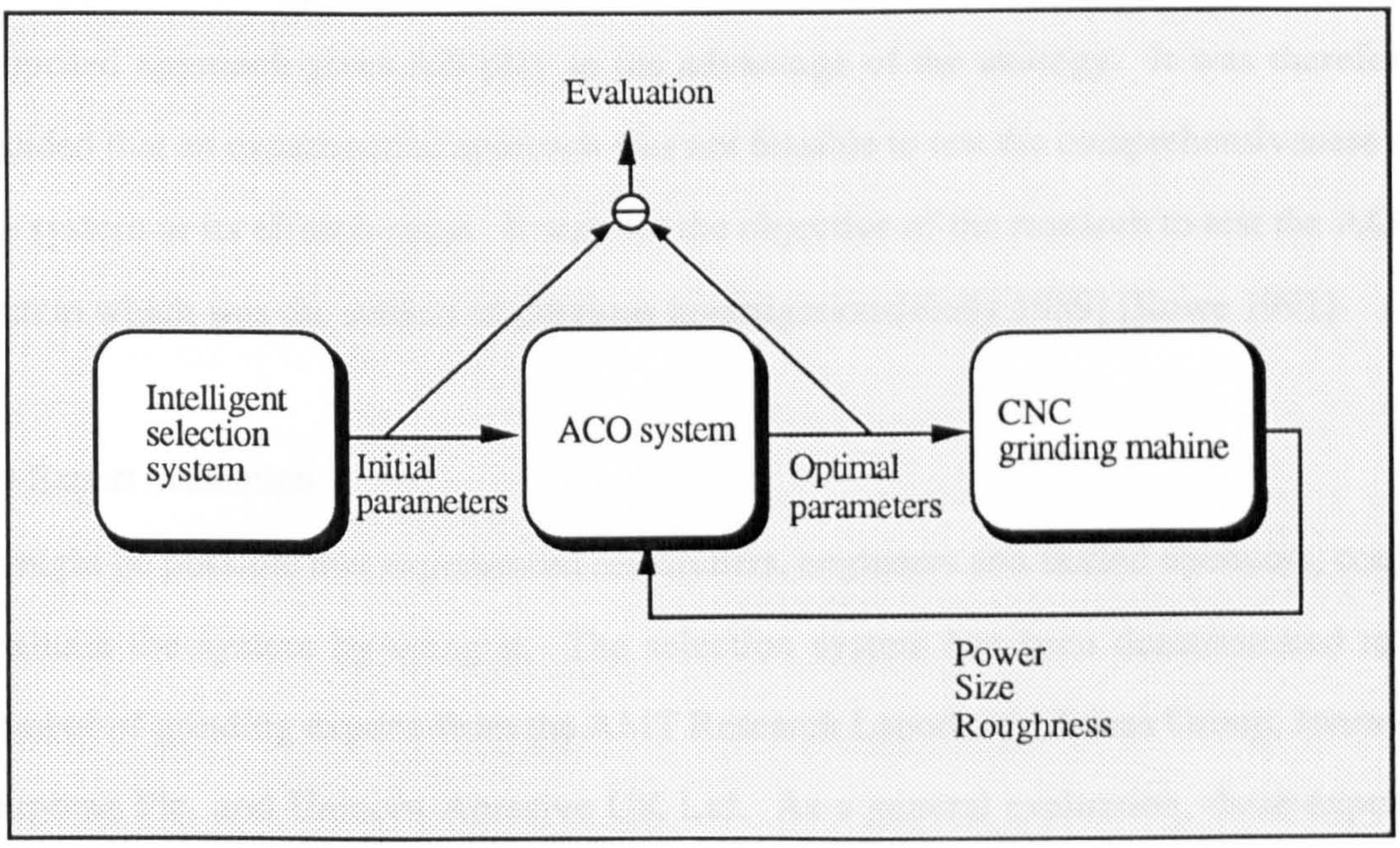

Figure 10.1 A possible approach to experimental evaluation of the system

A possible experimental procedure would be as follows:

(1) Select the grinding wheel, grinding coolant, the values of the dressing parameters and the values of the grinding parameters using the selection system.

(2) Set up the grinding machine with the recommended grinding conditions and optimise the grinding conditions from part to part based on the power limit, the burn limit and the roughness limit. The optimisation objective here is minimisation of the cycle time. The optimisation variables are v_f, v_w and f_d.

(3) Compare the recommended values and the optimised values, evaluate the results and store the optimum grinding conditions in the case base.

The above process may be repeated for different grinding problems, including different workpiece materials, different wheels and different grinding requirements. However, the experiments will be time consuming and expensive to cover a substantial range of grinding problems. It is therefore proposed that a better approach is that system tests and system learning are undertaken simultaneously while the system is used. This proposed approach gives full play to the advantage of the strategy. It was therefore decided that an experimental approach was not feasible to test the comprehensiveness of the system or its effectiveness. It was not the objective of the research to test the ACO system which was the subject of previous investigations[Kelly 1989] [Rowe 1991].

(ii) Expert evaluation

It might be possible that experienced researchers, engineers and skilled operators, could evaluate the system by using it. The selection system has been demonstrated to a number of grinding experts from the AMT Research Laboratory, Lucas Group, Jones & Shipman Plc. and Unicorn Abrasive UK Ltd. As a general evaluation, these experts thought the strategy was an original and reasonable approach which is easy to use. However, the system was not evaluated in detail.

(iii) Handbook evaluation

This approach evaluates the system comparing the system output and information derived from existing knowledge from handbooks. While the knowledge from

handbooks is less than optimal, this approach allows recommendations for a wide domain of grinding conditions to be tested against the system recommendations.

10.2 Comparison of the System Output and Data From the Handbooks

The system was evaluated by comparison of the system output and data from handbooks[MDC 1980][Universal 1992]. The data for the wheel specification and the values of dressing parameters were from The Grinding Data Book[Universal 1992]. The data for the values of the grinding parameters were from The Machining Data Handbook[MDC 1980]. The following is a description of the process employed for the comparison. To simplify the process, the Problem description only lists the variables which affect the grinding conditions. Common input information is: external cylindrical plunge grinding and single point diamond dressing.

Evaluation example 1.

Problem input:

- Workpiece material: Low & Medium carbon steel & alloy group, AISI 1050
- Material hardness: 50 R_c
- Wheel selection by the system
- Workpiece diameter: 54.8 mm
- Specified roughness: 0.55 μm Ra.

	System	Handbook
• wheel:	11A60K5V,	48A80KV,
• coolant:	water based	water based,
• wheel diameter:	d_s = 226 mm,	not available,
• wheel speed:	v_s = 26.13 m/s,	v_s = 28 - 33 m/s,
• work speed:	v_w = 0.47 m/s,	v_w = 0.35 - 0.5m/s,
• feedrate:	v_f = 0.02 mm/s,	v_f = < 0.013 mm/s,
• dressing lead:	f_d = 0.095 mm/s,	f_d = 0.10 mm/s,
• dressing depth:	a_d = 0.015 mm	a_d = 0.012 - 0.19 mm

The system recommended values derived from a similar case in case base. Case 4 in Table 10.1 is the nearest case. Since the roughness and the workpiece diameter were different from Case 4, the case was modified to meet the new requirement. The system recommended values mostly fell in the range of the recommendations of the handbook. However, the feedrate v_f recommended from the handbook tends to be conservative. The wheelspeed in the case was lower than handbook recommendation, perhaps because the wheel diameter in the case was rather smaller. The wheel specifications had some differences that may depend on which wheel was available. Since the old case was a successful case and the modification methods employed basic models which were proven to be reliable, the Solution provides a high confidence level. The user can modify the Solution according to his own knowledge.

Five example cases from different sources are illustrated in Table 10.1.

Table 10.1 Grinding cases

	Case 1 [Chen 1995]	Case 2 [Peters 1977]	Case 3 [Saka. 1992]	Case 4 [Zhu 1992]	Case 5 [Trmal 1974]
Machine tool	J&S S 10				
Material	High-c steels	AISI 52100	SCM435	Low-c steels	High-c steels
Hardness (R_c)	62	62	50-58	50-58	42
Max. roug.(R_a)	0.45 μm	1 μm	0.09 μm	0.40 μm	0.40 μm
Size tol.	±12 μm				
Roundness	1.5 μm				
Start diameter	16 mm	80 mm	50 mm	45.08 mm	35 mm
Finish diameter	15.70 mm			44.78 mm	
width	24 mm	20 mm	24 mm	12.05 mm	20 mm
wheel	WA80JV	EK60L7VX	A60L8V	11A60K5V	WA80MV
Wheel speed	33 m/s	45 m/s	40.9 m/s	26.13 m/s	30 m/s
Wheel diameter	365 mm	680 mm	415 mm	226 mm	300 mm
Dresser	SPD	SPD	SPD	SPD	SPD
Coolant	Water based	Oil EPS 12		Water based	
Workspeed	0.25 m/s	0.75 m/s	0.21 m/s	0.4 m/s	0.33 m/s
Feedrate	0.02 mm/s	0.013 mm/s	0.003 mm/s	0.024 mm/s	0.0087 mm/s
Dressing depth	0.015 mm	0.05 mm	0.02 mm	0.015 mm	0.02 mm
Dressing lead	0.15 mm/rev	0.2 mm/rev	0.03 mm/rev	0.05 mm/rev	0.167 mm/rev
Spark-out time	4 s		3 s		2 s
Specific energy	60 J/mm^3				
Grinding ratio	42				

These cases are good for recommending grinding conditions. However, these cases may not be optimal because they may be optimised for some specific aspect. In addition, it is obvious that five cases are far too few for effective use of the case based reasoning.

More comparisons are illustrated in Table 10.2. These examples were selected to cover different kinds of Problems, namely, all material groups, different material hardness, different roughness requirements and different workpiece diameters. The wheel can be specified by the system or by the user. To simplify the table, the Problem description is expressed by the Index.

Table 10.2 The comparison of the system output and handbooks output

No	Problem	Solution	Handbook	Similar case Solution
1	LS, M, 0, 3 1026, 0.45, 40	11A60K5V Water based v_s=26.13 m/s v_w=0.36 m/s v_f= 0.027 mm/s f_d=0.063 mm/r a_d=0.015 mm	48A80KV Emu oils - light v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.018 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	11A60K5V Water based v_s=26.13 m/s v_w=0.4 m/s v_f= 0.024 mm/s f_d=0.05 mm/r a_d=0.015 mm
2	LS, M, AVI*, 2 1025, 0.5, 40, 33, 220 * User specified wheel	A60LV Water based v_s=33 m/s v_w=0.44 m/s v_f= 0.034 mm/s f_d=0.078 mm/r a_d=0.015 mm	48A80KV Emu oils - heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.018 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	11A60K5V Water based v_s=26.13 m/s v_w=0.4 m/s v_f= 0.027 mm/s f_d=0.05 mm/r a_d=0.015 mm
3	HS, H, 0, 3 1080, 0.4, 25	WA80JV Water based v_s=33 m/s v_w=0.38 m/s v_f= 0.013 mm/s f_d=0.12 mm/r a_d=0.015 mm	WA80JV Oils-heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.029 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	WA80JV Water based v_s=33 m/s v_w=0.25 m/s v_f= 0.02 mm/s f_d=0.15 mm/r a_d=0.015 mm
4	HS, H, 0, 3 1060, 0.6, 20	WA80JV Water based v_s=33 m/s v_w=0.31 m/s v_f= 0.016 mm/s f_d=0.19 mm/r a_d=0.015 mm	WA80JV Oils-heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.036 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	WA80JV Water based v_s=33 m/s v_w=0.25 m/s v_f= 0.02 mm/s f_d=0.15 mm/r a_d=0.015 mm

No	Problem	Solution	Handbook	Similar case Solution
5	HS, H, 0, 1 1060, 0.95, 28.5	EK60L7VX Oil EPS 12 v_s=45 m/s v_w=0.29 m/s v_f= 0.036 mm/s f_d=0.18 mm/r a_d=0.05 mm	WA46JV Oils-heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.098 mm/s f_d=0.18 mm/r a_d=0.025 mm	EK60L7VX Oil EPS 12 v_s=45 m/s v_w=0.75 m/s v_f= 0.013 mm/s f_d=0.2 mm/r a_d=0.05 mm
6	HS, H, 0, 4 1070, 0.25, 38.5	WA100JV Oils-heavy duty v_s=28-33 m/s v_w=0.35-0.5 m/s v_f<0.019 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	WA100JV Oils-heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.019 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
7	HS, H, BBL, 3, 1070, 0.5, 38.5	B120T100B Emu oils - heavy v_s=25-38 m/s v_w=0.25-0.5 m/s v_f<0.052 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	B120T100B Emu oils - heavy v_s=25-38 m/s v_w=0.25-0.5 m/s v_f< 0.052 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and user required the CBN wheel
8	TS, H, 0, 2 A3, 0.5, 40	WA80JV Emu oils - heavy v_s=20-30 m/s v_w=0.3-0.5 m/s v_f< 0.012 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	WA80JV Emu oils - heavy v_s=20-30 m/s v_w=0.3-0.5 m/s v_f< 0.012 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
9	CI, M, 0, 2 ASTM A3, 0.6, 50	C60KV Emu oils - heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.014 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	C60KV Emu oils - heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.014 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based

No	Problem	Solution	Handbook	Similar case Solution
10	CI, L, 0, 1 ASTM A3, 1.0, 20	C46MV Emu oils - light v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.14 mm/s f_d=0.18 mm/r a_d=0.025 mm	C46MV Emu oils - light v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.14 mm/s f_d=0.18 mm/r a_d=0.025 mm	Rule based and neural network based
11	HT, H, 0, 2 M1, 0.6, 38	WA60JV Emu oils - heavy v_s=20-28 m/s v_w=0.3-0.5 m/s v_f< 0.01 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	WA60JV Emu oils - heavy v_s=20-28 m/s v_w=0.3-0.5 m/s v_f< 0.01 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
12	HT, M, 0, 2 T1, 0.95, 52	48A60KV Emu oils - heavy v_s=20-28 m/s v_w=0.3-0.5 m/s v_f< 0.023 mm/s f_d=0.18 mm/r a_d=0.025 mm	48A60KV Emu oils - heavy v_s=20-28 m/s v_w=0.3-0.5 m/s v_f< 0.023 mm/s f_d=0.18 mm/r a_d=0.025 mm	Rule based and neural network based
13	MS, L, 0, 2 ASTM A217, 0.7, 43	48A80JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.012 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	48A80JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.012 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
14	MS, L, 0, 2 ASTM A217, 0.95, 36	48A46JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.077 mm/s f_d=0.18 mm/r a_d=0.025 mm	48A46JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.077 mm/s f_d=0.18 mm/r a_d=0.025 mm	Rule based and neural network based

No	Problem	Solution	Handbook	Similar case Solution
15	AS, L, 0, 3 ASTM A296, 0.55, 28	C80JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.018 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	C80JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.018 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
16	AS, L, 0, 2 ASTM A296, 1, 35	C46JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.057 mm/s f_d=0.18 mm/r a_d=0.025 mm	C46JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.057 mm/s f_d=0.18 mm/r a_d=0.025 mm	Rule based and neural network based
17	AA, L, 0, 2 ASTM A296, 0.98, 40	C46JV Oils - light v_s=28-33 m/s v_w=0.25-0.77 m/s v_f< 0.05 mm/s f_d=0.18 mm/r a_d=0.025 mm	C46JV Oils - light v_s=28-33 m/s v_w=0.25-0.77 m/s v_f< 0.05 mm/s f_d=0.18 mm/r a_d=0.025 mm	Rule based and neural network based
18	NA, L, 0, 2 Nickel 200, 0.75, 34	C60JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.015 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	C60JV Emu oils - heavy v_s=28-33 m/s v_w=0.25-0.5 m/s v_f< 0.015 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based
19	CA, L, 0, 2 Nickel 200, 0.7, 45	C80JV Emu oils - heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.016 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	C80JV Emu oils - heavy v_s=28-33 m/s v_w=0.35-0.5 m/s v_f< 0.016 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based

No	Problem	Solution	Handbook	Similar case Solution
20	TA, L, 0, 2 Ti-8Mn, 0.65, 38	C80JV Special - light v_s=15-20 m/s v_w=0.35-0.5 m/s v_f< 0.019 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	C80JV Special-light v_s=15-20 m/s v_w=0.35-0.5 m/s v_f< 0.019 mm/s f_d=0.1 mm/r a_d=0.012-0.019 mm	Rule based and neural network based

Since the cases in the case base were mainly in the Carbon & Alloy Steel groups, most Solutions in the material groups in Table 10.2 were case based. The Solutions from case based agent mostly fell in the range of the recommendations of the handbook. Some differences in the system output and the Handbook output can be explained as in example 1.

Since there were insufficient cases in the case base, for other material groups, the system Solutions were obtained from the rule based reasoning agent and the neural network agent and the comparison showed the system Solutions fitted the handbook outputs very well.

10.3 Use of the System and Further Development

In use, the system has the following features:

1. The user is guided to the next step of operation.

2. The user can exit the system at any time.

The system employed the menus and the dialogue boxes as input and output interfaces. The choice-list method was used as much as possible for the user input so that the user can input information according to the format the system required. The material data base and the case base can browse, add, delete and modify easily. Moreover, the

system can be operated using the keyboard or the mouse. Therefore, the use of the system is convenient and the user can easily master the operational method. However, the system help function remains to be completed.

The learning of the case based reasoning agent and neural network agent can be completed without changing any programme code. The learning of the case base reasoning agent can be directly completed by the operator. The following is an example for the system learning test.

The example in No 8 in Table 10.1 gave an initial set of grinding conditions. The grinding conditions were then modified as follows:

- wheel: WA80JV,
- coolant: Emulsifiable oils - heavy duty
- wheel speed: $v_s = 30$ m/s,
- wheel diameter: $d_s = 300$ mm
- work speed: $v_w = 0.4$ m/s,
- feedrate: $v_f = 0.014$ mm/s,
- dressing lead: $f_d = 0.1$mm/r,
- dressing depth: $a_d = 0.015$ mm

The new grinding conditions were saved as a case in the case base and a similar grinding problem was given:

Problem input:

- Workpiece material: Tool steel, W1
- Material hardness: 60 Rc
- Wheel selection by the system
- Workpiece diameter: 32.5 mm
- Specified roughness: 0.4 μm Ra.

System output:

- wheel: WA80JV,
- coolant: Emulsifiable oils - heavy duty
- wheel speed: $v_s = 30$ m/s,
- wheel diameter: $d_s = 300$ mm
- work speed: $v_w = 0.35$ m/s,
- feedrate: $v_f = 0.017$ mm/s,
- dressing lead: $f_d = 0.064$ mm/r,
- dressing depth: $a_d = 0.015$ mm

The solution was based on the newly learned case. If the solution is satisfied or is modified, the case can be saved as a new case.

10.4 Discussion

It has been shown that it is possible to design a system for selection of grinding conditions using case based reasoning. Since a database is continually generated using successful cases, the system is capable of automatically updating itself taking account of the latest technology. Theoretical or empirical models of the grinding process are unnecessary because grinding cases are represented directly. The case base can be edited very easily so that new cases can be added into the knowledge base by the user without the assistance of a knowledge engineer. With use of the system, more grinding cases are obtained and more knowledge is learned, which shows that case based reasoning inherently incorporates an incremental learning mechanism. A disadvantage of case based reasoning is the need for initial grinding cases to cover the whole problem space. Case based reasoning must therefore be used in conjunction with other methods to set up initial grinding conditions in the first instance. It has been shown that the use of cases in the case base can be extended by inferencing using a kinematic model for 'near neighbour' cases . Other agents demonstrated for generating approximate conditions included a rule based system for grinding conditions and a neural network for

wheel selection. Finally, expert knowledge can be captured directly from the operator. It is therefore concluded that case based reasoning provides a flexible and practically ideal technique for selecting grinding conditions.

Since there is no model for selecting the grinding wheel, the neural network has been shown to be an appropriate method for wheel selection. Since explicit rules are not needed, a neural network system for wheel selection was easily and quickly developed using training data from catalogues and reference books. The system is flexible, allowing for further learning from new data to enlarge and improve the knowledge base. However, further learning requires the network to be retrained which requires the assistance of a knowledge engineer. Obviously, the learning mechanism inherent in case based reasoning is vastly superior to the learning process employed for a neural network. However, while the case based reasoning agent can cope with the comprehensive grinding problem as it is experienced and provides for any grinding situation, it would be a very large task to programme the case base for a comprehensive range of cases involving a wide range of grinding wheels. Using a neural network however, it was possible to capture information for selection of a wide range of grinding wheels with a relatively small training data set. Trials demonstrated that the system reliably predicts the same wheels as would be found from the handbooks used for the reference data. A major disadvantage of the wheel selection system was the lack of sufficient examples from recent practice to train the system, especially as there was a lack of data for CBN wheels. In principle, it is not difficult to train using a range of data from the main manufacturers to increase the scope of the neural network system and thus make it reasonably comprehensive.

Since there is an incomplete understanding of the grinding process, the rule based reasoning agent can only provide approximate values for grinding conditions. However, it has been shown that it is possible to design a rule based agent to suggest a starting point where case based reasoning fails to provide a solution. The evaluation of the system shows that the selection system currently relies mainly on the rule based

agent, except for problems in the Carbon & Alloy Steel groups.

It has been argued that a single technology cannot effectively solve the whole grinding problem. The multi-agent approach has shown the ability to integrate different intelligent technologies into one system. It has also been shown that the multi-agent system can automatically actuate different agents to provide the solution. The system is convenient to use and to extend. Much work is required to improve the system but the multi-agent system has shown the potential to be the best approach to solve the problem of the selection of grinding conditions.

Chapter 11 Conclusions

It has been shown that it is possible to design a system for selection of grinding conditions which is flexible and incorporates automatic updating of the database. It has been shown that in the initial training of the intelligent system, the need for knowledge acquisition is an impediment to the provision of a comprehensive system. Therefore, the incorporation of a learning ability is of the greatest importance in an intelligent system for grinding, to allow for updating using the latest information and the most modern technology.

Case based reasoning is a highly suitable approach for the selection of grinding conditions. Results from case based reasoning provide a high level of confidence. The most important advantage of case based reasoning is that technology can be relatively easily updated by incorporating new cases into the case base. However, it is difficult to find enough grinding cases to cover a sufficiently large problem space for the initial training of the system. Case based reasoning may fail to provide a solution for problems where there is a lack of relevant cases. Therefore, other techniques need to be used in association with case based reasoning.

It has been shown that a neural network can be trained for selection of a wide range of wheels using a relatively small set of training data. The modelling does not require a full understanding of the relationships between the grinding wheel and the grinding process. However, the application of expert knowledge to formulate rules helps to reduce the complexity of the network.

A rule based reasoning approach can provide approximately suitable values for grinding conditions although the conditions recommended may be far from optimal. However, a rule based system provides a starting point if case based reasoning fails to provide a solution .

A multi-agent approach has been proposed. The multi-agent approach combines different representational methods for the knowledge and inferencing techniques, including case based reasoning, rule based reasoning, neural network reasoning and operator input. The multi-agent approach combines the strengths and overcomes the weaknesses of the different agents employed, to generate hybrid solutions.

It has been shown that the intelligent system based on the multi-agent approach works as expected. The system can provide optimal, near optimal or approximately suitable grinding conditions depending on the situations and the agent employed. The system has a good learning ability. The user can update the cases in the case base without the help of a knowledge engineer. It is therefore expected that the effectiveness of the system will be increased with system application .

Although further work is required to complete a comprehensive system for a range of grinding processes, the system demonstrates the principle of a multi-agent system for selection of grinding conditions which can easily be extended. As demonstrated, the system already covers a useful range of external cylindrical grinding practice with conventional wheels.

Chapter 12 Suggestions for Further Work

The system should be applied in industry in order to further evaluate the strategy and the system and obtain more application cases.

The system is mainly developed for external cylindrical plunge grinding. Other types of grinding process require to be incorporated into the system. In addition, the selection of CBN wheels and diamond wheels should be investigated.

Some technologies for developing the system need further improvement. For example, the method of arrangement and management of the computer memory for an increasingly large case base should be investigated.

An adaptive control agent should be included into the system so that the system can have a self-learning ability.

References

Adily H, 1990, Knowledge Engineering, McGraw-Hill Publishing Company, New York

Amitay G, Malkin S, Koren Y, 1981, Adaptive control optimisation of grinding, Trans ASME, J of Eng for Ind, Vol 103, pp 103-108.

Ashby M F, Jones D R H, 1992, Engineering Materials 2: An Introduction to Microstructures, Processing and Design, Pergamon Press, UK

ASM, 1989, Metals Handbook; Vol 16: Machining, Ninth Edition, ASM International, USA

Balakrishana P, Devries M F, 1983, A review of computerised machinability data base systems, Proceedings of 10th NAMRC, pp 348-356

Barletta R, 1991, An introduction to case-based reasoning, AI Expert, August, pp 43-49

Brinksmeier E, Tönshoff H K, Inasaki I, Peddinghaus J, 1993, Basic parameters in grinding, Technical Reports, Annals of CIRP, Vol 42, No 2, pp 795-799

Botti V, Barber F, Grespo A, Garcia F A, Ripoll I, Gallardo D, Hernandez L, 1995, A temporal blackboard for a multi-agent environment, Data & Knowledge Engineering Vol 15, pp 189-211

Chen X, 1995, Strategy for the selection of grinding wheel dressing conditions, PhD Thesis

Chen Y T, Shin Y C, 1991, Surface grinding process advisory system with fuzzy logic, Control of Manufacturing Processes, ASME Winter Annual Meeting, DSC Division, Vol 28, PP 67-77.

Chester D L, 1990, Why two hidden layers are better than one , Proceedings of international Joint Conference: Neural Networks, Vol 1, Washington D C, pp 265-268

Chevrier V, 1994, Using the blackboard paradigm for real complex problems: the Abysse project, International Journal of Engineering Intelligent systems for Electrical Engineering and Communications, Vol 2, No 3, pp 163-174

Colding B N, Intelligent selection of machining parameters for metal cutting operations: the least expensive way to increase productivity, Robotics & Computer-

Integrated Manufacturing, Vol 9, No 4/5, pp 407-412

Dagli C H, 1994, Artificial Neural Networks for Intelligent Manufacturing, Chapman & Hall, London

Engelmore R, Morgan T, 1988, Blackboard System, Reading, MA: Addison-Wesley

Hornik K, 1989, Multilayer feedforward network are universal approximators, Neural Networks, Vol. 2, pp 359-366

Hunt V D, 1986, Artificial Intelligence and Expert Systems Source Book, Chapman Hall.

Inoue H, Suto T, Waida T, 1987, A knowledge-based system for grinding, Proceedings of the 6th International Conference on Production Engineering, Osaka, pp 377-382.

Kalpakjian S, 1991, Manufacturing Processes for Engineering Materials, Addison-Wesley Publishing Company, USA

Kelly S, Rowe W B, Moruzzi J L, 1989, Adaptive grinding control, Advanced Manufacturing Engineering (Butterworths), Vol 1, pp 287-295.

Ketler K, 1993, Case-based reasoning: an introduction, Expert Systems With Applications, Vol 6, pp 3-8

Kim G, Inasaki I, 1993, Establishment of optimum grinding conditions utilising the fuzzy regression model, Transaction of the Japan Society of Mechanical Engineers, Vol 59, pt 566, pp 3186-3192

King R I, Hahn R S, 1986, Hand of Modern Grinding Technology, Chapman and Hall, New York

König W, 1991, Artificial intelligence and simulation in grinding processes, 11th European Congress on Operational Research, Aachen.

Kolodner J, 1993, Case-Based Reasoning, Morgan Kaufmann Publishers, Inc., San Mateo

Kohonen T, 1988, An introduction to neural computing, Neural Networks vol. 1, pp 3-16

Liao T W, Chen L J, 1994, A neural network approach for grinding process:

modelling and optimisation, Int. J. Mach. Tools Manufact, Vol 34, No 7, pp 919-937.

Lippmann R, 1987, An introduction to computing with neural nets, IEEE ASSP Mag. April, pp 4-21

Matsushima K, Sata T, 1980, Development of intelligent machine tool, Journal of The Faculty of Engineering, The University of Tokyo, Vol 35, No 3, pp 299-314

Malkin S, Koren Y, 1980, Off-line grinding optimisation with a micro-computer, Annals of CIRP, Vol 29, pp 213-219

Malkin S, 1981, Grinding cycle optimisation, Annals of CIRP, Vol 30, No 1, pp 213-217

Malkin S, 1989, Grinding Technology: theory and applications of machining with abrasives, Ellis Horwood Limited, New York

Mani N, 1994, Towards modelling a multi-agent knowledge system in engineering design, Proceedings of the IEEE International Conference on System, Man and Cybernetics, Vol 1, pp 160-164

Masters T, 1993, Practical Neural Network Recipes in C++, Academic Press Inc., USA

MDC, Machinability Data Centre, 1980, Machining Data Handbook, 3rd Edition, Vol 2, Metcut Research Associates Inc., USA

Midha P S, Zhu C B, Trmal G J, 1990, An expert system for wheel selection for cylindrical grinding operations, Proceedings of 6th International Conference on Computer Aided Production Engineering, London, pp 445-450.

Midha P S, Zhu C B, Trmal G J, 1991, An application of the expert system approach to the production grinding process, Proceedings of the 4th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-91, Hawaii, July, pp 330-337.

Midha P S, Zhu C B, Trmal G J, 1991 Optimum selection of grinding parameters using process modelling and knowledge based system approach, Journal of Materials Processing Technology, Vol 28, pp 189-198.

Mirzai A R, 1990, Artificial intelligence: Concepts and Applications in Engineering, Chapman and Hall, London

Monostori L, Barschdorff D, 1992, Artificial neural networks in intelligent manufacturing, Robotics & Computer-integrated Manufacturing, Vol 9, No. 6, pp 421-437

Mott S, 1993, Case-based reasoning: market, applications, and fit with other technologies, Expert Systems With Applications, Vol. 6, pp 97-104

Nakajima T, 1978, A new standard for proper selection of grinding wheels in plunge grinding operation, Annals of CIRP, Vol 27, No 1, pp 249-253

Nakajima T, Tsukamoto S, Murakami D, Kurata K, 1992, Fuzzy in-process control plunge grinding techniques, Journal of the Japan Society of Precision Engineering, Vol 58, No 2, pp 313-318.

Nagasaka K, Kita Y, Kitaguchi Y, Tanibayashi A, 1991, The construction of expert systems for the grinding process, Journal of the Japan Society of Precision Engineering, Vol 57, No 9, pp 1661-1666.

NeuralDesk, User's Guide, 1992, Neural Computer Sciences, UK

Obradovic Z, Yan P, 1990, Small depth polynomial size neural networks, Neural Computation, Vol 2, pp 402-404

Occello M, Demazeau, 1994, Building real time agents using parallel blackboards and its use for mobile robotics, Proceedings of the IEEE International Conference on System, Man and Cybernetics, Vol 2, pp 1610-1615

O'connor L J, Lan M S, Partriridge D R, Lee J M F, 1992, A case based reasoning approach to automated weld-process design, Applied Artificial Intelligent, Vol 6, pp 315-330

Pattinson E J, Lyon J, 1975, The collection of data for the assessment of a grinding wheel dressing treatment, Proceedings of 15th Int. MTDR Conference, The Macmillan Press, pp 317-323

Peters J, Snoeys R, Decneut A, 1976, The proper selection of grinding conditions in cylindrical plunge grinding, Annals of CIRP, Vol 26, No 1, pp 387-394

Peters J, Aerens R, 1980, Optimisation procedure of three phase grinding cycle of a series without intermediate dressing, Annals of the CIRP, Vol 29, No 1, pp 195-200

Rangwala S S, Dornfeld D A, 1989, IEEE Transactions on Systems, Man, and

Cybernetics, Vol 19, No 2, pp 299-314

Rowe W B, Bell W F, Brough D, 1987, Limit chart for high removal rate centreless grinding, Int. J. Mach. Tools Manufact. Vol. 27, No. 1, pp 15-25

Rowe W B, Pettit J A, Boyle A, Moruzzi J L, 1988, Avoidance of thermal damage in grinding and prediction of the damage threshold, Annals of CIRP, Vol 37, No 1, pp 327-330.

Rowe W B, Allanson D R, Pettit J A, Moruzzi J L, Kelly S, 1991, Intelligent CNC for grinding, Proceedings of the Institution Mechanical Engineers, Vol 205, pp 233-239.

Rowe W B, Allanson D R, Thomas D A, Moruzzi J L, 1993, Intelligent CNC for grinding - Problems of accuracy control, Presented to CIRP Abrasive Processes Scientific Technical Committee in Paris.

Rowe W B, Li Y, Inasaki I, Malkin S, 1994a, Applications of artificial intelligence in grinding, Keynote Paper, Annals of CIRP, Vol 43, No 2, pp 521-531

Rowe W B, Thomas D A, Allanson D R, Moruzzi J L, 1994b, Intelligent CNC for Grinding, The Manufacturing Engineer, Proceedings of IEE.

Rowe W B, Chen X, Mills B, 1995, Towards an adaptive strategy for dressing in grinding operations, Proceedings of the 31st International MATADOR Conference, published by The Macmillan Press Ltd., 20th-21st April, pp 415-420

Rowe W B, Li Y, Mills B, Allanson D R, Application of intelligent CNC in grinding, Journal of Computer in Industry, 1996.

Rumelhart D E, Hinton G E, Williams R J, 1986, Learning representations by back-propagating errors, Nature, Vol 323, pp 533-536

Rumelhart D E, McClelland J, 1986, Parallel Distributed Processing, Vol 1, MIT, Cambridge, MA

Sakakura M, Inasaki I, 1992, Neural network approach to the decision-making process for grinding operations, Annals of the CIRP, Vol 41, No 1, pp 353-356.

Sakakura M, Inasaki I, 1993, Intelligent data base for grinding operations, Annals of CIRP, Vol 42, No 1, pp 379-382.

Sathyanarayanan G, Lin I J, Chen M K, 1992, Neural network modelling and

multi-objective optimisation of creep feed grinding of superalloys International Journal of Production Research, Vol 30, No 10, pp 2421-2438.

Schank R, 1982, Dynamic Memory: A Theory of Learning in Computer and People. Cambridge University Press, Cambridge

Shoureshi R, 1993, Intelligent control systems - Are they for real?, ASME Journal of Dynamic Systems, Vol 115, pp 392-407.

Snoeys R, Peters J, Decneut A, 1974, The significance of chip thickness in grinding, Annals of CIRP, Vol 23, pp 227-237

Tönshoff H K, Peters J, Inasaki I, Paul T, 1992, Modelling and simulation of grinding processes, Keynote Paper, Annals of CIRP, Vol 41, No 2, pp 677-688.

Tönshoff H K, Zinngrebe M, Kemmerling, 1986, Optimisation of internal grinding by microcomputer based force control, Annals of CIRP, Vol 35, No 1, pp 293-296.

Trmal G, Kaliszer H, 1974, Optimisation of a grinding process and criteria for wheel life, Proceedings of MTPR Conference, pp 311-315

Tsatsoulis C, Kashyap R L, 1993, Case-based reasoning and learning in manufacturing with the TOLTEC planner, IEEE Transactions on System, Man, and Cybernetics, Vol 23, No 4, July/August, pp 1010-1023

Ueda N, Matsuo T, Obuchi Y, Nomura H, 1988, Expert system for grinding, Proceedings of 3rd International Grinding Conference, Wisconsin.

Universal, 1992, Grinding Data Book, Universal Grinding Wheel Co. Ltd., UK

Venk S, Govind R, 1990, An expert system approach to optimisation of the centreless grinding process, Annals of CIRP Vol 39, No 1, pp 489-492.

Verkerk J, Pekelharing A J, 1979, The influence of the dressing operation on productivity in precision grinding, Keynote paper, Annals of the CIRP, Vol 28, No 2, pp 487-495

Watson I, 1995, Developing industrial awareness of case based reasoning technology in Europe, ECN, July, p5

Xiao G, Malkin S, Danai K, 1992 Intelligent control of cylindrical plunge grinding, Proceedings of the American Control Conference, Chicago, Vol 1, pp 391-

398.

Xiao G, Malkin S, Danai K, 1993, An autonomous system for cylindrical plunge grinding, Proceedings of NSF Design and Manufacturing Systems Conference, pp 399-403.

Yoon Y, Acree A D, Peterson L L, Development of a case based expert system: application to a service coordination problem, Expert Systems With Applications, Vol 6, pp 77-85

Zhu C B, Midha P S, Trmal G J, 1992a, Development of knowledge base abrasive wheel selection system using a proprietary expert system shell, Proceedings of 4th International Congress on Condition Monitoring and Diagnostic Engineering Manufacturing, Selis, France, pp 318-325.

Zhu C B, 1992b, Optimisation of the grinding process using process modelling and knowledge based system approach, PhD thesis, University of the West of England, Bristol.

Zurada J M, 1992, Artificial Neural Systems, West Publishing Company, St. Paul

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

                Feedforward Neural Network
                    Version 1.0

          Copyright ©1994 AMT Research Laboratory

*********************************************************/

/*********************************************************

        Source file 1:      User Interface

/*********************************************************/


#include <stdlib.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <stdarg.h>
#include <strstrea.h>

#define Uses_TEventQueue
#define Uses_TEvent
#define Uses_TProgram
#define Uses_TApplication
#define Uses_TKeys
#define Uses_TRect
#define Uses_TMenuBar
#define Uses_TSubMenu
#define Uses_TMenuItem
#define Uses_TStatusLine
#define Uses_TStatusItem
#define Uses_TStatusDef
#define Uses_TDeskTop
#define Uses_TView
#define Uses_TWindow
#define Uses_TFrame
#define Uses_TDialog
#define Uses_TButton
#define Uses_TLabel
#define Uses_TInputLine
#define Uses_TFileEditor
#define Uses_TFileDialog
#define Uses_TChDirDialog
#define Uses_TEditWindow
#define Uses_MsgBox
#define Uses_TScroller
#define Uses_TScrollBar

#include <tv.h>

const int cmWin          = 201;
const int cmSaasN        = 202;
const int cmMyabout      = 203;
const int cmTraining     = 204;
const int cmSetup        = 205;
const int cmTest         = 206;
const int cmTrain        = 207;
const int cmStop         = 208;

const cmNew              = 100;
const cmOpen             = 101;
const cmChangeDirectory  = 103;
const cmDosShell         = 104;
const cmShowClipboard    = 105;

extern void memory1(int,int,int,char *, char *);
extern void fileopenc(int hi1,int hi2,int hi3,char *,char *);
extern float Trainingc(int hi1,int hi2,int hi3,float cb,float ca);
extern void filewritec(int hi1,int hi2,int hi3,char *);
extern void memfreec(int,int,int);
extern char *Test(char *,char*,char*);

ushort execDialog( TDialog *d, void *data );
float mxer;
```

```
int p; unsigned long sum,n;
int first1, second, third;
float mom, lea, err1;

struct SetUpData
{
  char FstInputLine[10];
  char SndInputLine[10];
  char TrdInputLine[10];
  char MomInputLine[10];
  char LeaInputLine[10];
  char ErrInputLine[10];
};

SetUpData *setupData;

ushort errorHandler(int, ... );
```

```
class TClipboard {

   TEditWindow* window;

public:

   TClipboard();
   void select();
};

TClipboard::TClipboard()
{
   // setup an edit window to use as the clipboard
   window = new TEditWindow(TRect(4, 4, 50, 10),
                            0, 0);
   if (window) {
     // make the clipboard invisible by default
     window->hide();
     TProgram::deskTop->insert(window);

     // always disable undo-editing in the clipboard
     TEditor::clipboard = window->editor;
     TEditor::clipboard->canUndo = False;
   }
}

void TClipboard::select()
{
   window->select();
   window->show();
}


class TNeuApp : public TApplication
{
public:
   char fileName[MAXPATH];
   TClipboard clipboard;
   int windowNumber;
public:
   TNeuApp();
   ~TNeuApp();
   static TStatusLine *initStatusLine( TRect r );
   static TMenuBar *initMenuBar( TRect r );
   virtual void handleEvent( TEvent& event);
   char *FileOpen(char *,char * );
   void SaasNeu();
   void SaveD();
   void about();
   void SetUp();
   void Training();
   void Win(char *);
   void edit1();
   void fileNew();
   void fileOpen();
   void changeDirectory();
   void dosShell();

};
```

```
TNeuApp *neu;

TNeuApp::TNeuApp() :
    TProgInit( &TNeuApp::initStatusLine,
               &TNeuApp::initMenuBar,
               &TNeuApp::initDeskTop
             )
{
   setupData= new SetUpData;
    strcpy( setupData-> FstInputLine,"4");
    strcpy( setupData-> SndInputLine,"0");
    strcpy( setupData-> TrdInputLine,"0");
    strcpy( setupData-> MomInputLine,"0.9");
    strcpy( setupData-> LeaInputLine,"0.3");
    strcpy( setupData-> ErrInputLine,"0.04");
    first1=atoi(setupData->FstInputLine);
   second=atoi(setupData->SndInputLine);
   third=atoi(setupData->TrdInputLine);
   mom=atof(setupData->MomInputLine);
   lea=atof(setupData->LeaInputLine);
   err1=atof(setupData->ErrInputLine);

   windowNumber = 0;

   // disable all the standard menu commands that are
   // invalid when no files are loaded
   TCommandSet commands;
   commands += cmSave;
   commands += cmSaveAs;
   commands += cmCut;
   commands += cmPaste;
   commands += cmCopy;
   commands += cmClear;
   commands += cmUndo;
    disableCommands(commands);

   // establish a handler for errors during edit
   // operations. There can be only one such handler
   // in the system at the same time, since editorDialog
   // is a static data member of TEditor.
    TEditor::editorDialog = errorHandler;

   about();

}

TNeuApp::~TNeuApp()
{

}

void TNeuApp::handleEvent(TEvent& event)
{
    TApplication::handleEvent(event);
   if( event.what == evCommand )
          {
          switch( event.message.command )
            {
            case cmSetup:
             SetUp();
             break;
       case cmTraining:
            Training();
            break;
      case cmTest:
             char inp1[128];
             char outp1[128];
             char weight[128];
             char *ap;
             if((ap=FileOpen("*.neu","open test input file"))!=0){
             strcpy( inp1,ap );
             if((ap=FileOpen("*.neu","open test output file"))!=0){
             strcpy( outp1,ap );
             if((ap=FileOpen("*.w","open weight file"))!=0){
             strcpy( weight,ap );
             char *outtest1=Test(inp1,outp1,weight);
             Win(outtest1);
             }}}
             break;
```

```
case cmNew:
fileNew();
break;

case cmOpen:
fileOpen();
break;

case cmChangeDirectory:
changeDirectory();
break;

case cmDosShell:
dosShell();
break;

case cmShowClipboard:
clipboard.select();
break;

            case cmMyabout:
            about();
            break;

            default:
                return;
            }
        clearEvent( event );        // clear event after handling
        }
}

/*-------------------------------------------------------------------------
                        Main Menu
-------------------------------------------------------------------------*/

TMenuBar *TNeuApp::initMenuBar( TRect r )
{
 char i[]= {126,240,126,0};
 r.b.y = r.a.y + 1;  // set bottom line 1 line below top line
return new TMenuBar( r,
        *new TSubMenu( i, kbAltA )+
            *new TMenuItem( "~A~bout",cmMyabout,kbAltA,hcNoContext,"Alt-A")+
        *new TSubMenu( "~F~ile", kbAltF )+
        *new TMenuItem("~O~pen", cmOpen, kbF3) +
         *new TMenuItem("~N~ew", cmNew, kbNoKey) +
         *new TMenuItem("~S~ave", cmSave,
                        kbF2, hcNoContext, "F2") +
         *new TMenuItem("S~a~ve as...", cmSaveAs, kbNoKey) +
         newLine() +
         *new TMenuItem("~C~hange dir...",
                        cmChangeDirectory, kbNoKey) +
         *new TMenuItem("~D~OS shell", cmDosShell, kbNoKey) +
         *new TMenuItem("E~x~it", cmQuit, kbAltX,
                        hcNoContext, "Alt-X") +

  *new TSubMenu("~E~dit", kbAltE) +
 *new TMenuItem("~U~ndo", cmUndo, kbNoKey) +
 newLine() +
 *new TMenuItem("Cu~t~", cmCut, kbShiftDel,
                        hcNoContext, "Shift-Del") +
 *new TMenuItem("~C~opy", cmCopy, kbCtrlIns,
                        hcNoContext, "Ctrl-Ins") +
 *new TMenuItem("~P~aste", cmPaste, kbShiftIns,
                        hcNoContext, "Shift-Ins") +
 *new TMenuItem("~S~how clipboard",
                        cmShowClipboard, kbNoKey) +
 newLine() +
 *new TMenuItem("~C~lear", cmClear, kbCtrlDel,
                        hcNoContext, "Ctrl-Del") +
*new TSubMenu( "~T~raining", kbAltT )+
   *new TMenuItem( "~S~et up", cmSetup,kbAltS, hcNoContext,"Alt-S" )+
   *new TMenuItem( "~T~rain Network", cmTraining,kbAltP,
hcNoContext,"Alt-P" )+
*new TSubMenu( "T~e~st", kbAltE )+
   *new TMenuItem( "T~e~st", cmTest,kbAltE,hcNoContext,"Alt-E")+
*new TSubMenu( "~E~xit", kbAltF )+
   *new TMenuItem( "E~x~it", cmQuit, cmQuit, hcNoContext, "Alt-X" )
```

```
            // new dialog menu added here
          );
}

TStatusLine *TNeuApp::initStatusLine( TRect r )
{
   r.a.y = r.b.y - 1;     // move top to 1 line above bottom
   return new TStatusLine( r,
          *new TStatusDef( 0, 0xFFFF ) +
          // set range of help contexts
             *new TStatusItem( "~F1~ Help" , kbF1, cmHelp ) +
             *new TStatusItem( "~F10~ Menu" , kbF10, cmMenu ) +
             // define an item
             *new TStatusItem( "~Alt-X~ Exit", kbAltX, cmQuit ) +
             // and another one
             *new TStatusItem( "~Alt-F3~ Close", kbAltF3, cmClose )
             // and another one
          );
}
```

```
class TTLabel:public TLabel{
public:
TTLabel(const TRect& bounds, const char *aText, TView *aLink):
TLabel(bounds, aText, aLink){};
virtual void handleEvent(TEvent& event);
};

class TTInputLine:public TInputLine{
public:
TTInputLine(const TRect& bounds, int aMaxLen):
TInputLine(bounds, aMaxLen){};

virtual TPalette& getPalette() const;
};

TPalette& TTInputLine::getPalette() const
{
  const char *cpInputLine="\x1A\x1A\x1B\x1C";
  static TPalette palette(cpInputLine, sizeof(cpInputLine)-1);
  return palette;
}
```

```
class TrainDialog:public TDialog
{
  public:
  TrainDialog(const TRect& bounds, const char *aTitle);
  virtual void handleEvent(TEvent& event);
};
void TrainDialog::TrainDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&TrainDialog::initFrame)
{
}

void TrainDialog::handleEvent(TEvent& event)

{
   n=n+1;
    TDialog::handleEvent(event);

if( event.what == evCommand )
 {
       switch( event.message.command )
       {
       case cmTrain:
       p=1;
       break;
       case cmStop:
       p=0;
        default:
           return;
         }

       clearEvent( event );          // clear event after handling
       }

       if(mxer>err1&&n>3&&p==1)
       {
        sum=sum+1;
         mxer=Trainingc(first1, second, third, mom, lea);
```

```
        }

        if(n>3)
        {
        textbackground(7);
        textcolor(4);
        gotoxy(43,11);
        cprintf("%5lu    ",sum);
        gotoxy(43,13);
        cprintf(" %f",mxer);
        }
    }


class SaveDialog:public TDialog
{
  public:
  SaveDialog(const TRect& bounds, const char *aTitle);
  virtual void handleEvent(TEvent& event);
};

void SaveDialog::SaveDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&TrainDialog::initFrame)
{
}

void SaveDialog::handleEvent(TEvent& event)

{

    TDialog::handleEvent(event);

   if( event.what == evCommand )
    {
          switch( event.message.command )
          {
          case cmSaasN:
          neu->SaasNeu();
          endModal(cmSaasN);
          break;
```

```
          default:
             return;
           }

          clearEvent( event );          // clear event after handling
          }
    }

ushort execDialog( TDialog *d, void *data )
{
   TView *p = TProgram::application->validView( d );
   if( p == 0 )
          return cmCancel;
   else
          {
          if( data != 0 )
             p->setData( data );
          ushort result = TProgram::deskTop->execView( p );
          if( result != cmCancel && data != 0 )
             p->getData( data );
          TObject::destroy( p );
          return result;
          }
}

void TNeuApp::SaveD()
{
   SaveDialog *pd = new SaveDialog( TRect(20, 8, 60, 16), "Save..." );
   if( pd )
   {
    pd->insert(new TStaticText(TRect(8, 2, 32, 3),
                          "Do you wish to save it ?"
                ));

   pd->insert( new TButton( TRect( 7, 5, 17, 7 ), "~S~ave", cmSaasN,
                          bfDefault ));

   pd->insert( new TButton( TRect( 21, 5, 31, 7 ), "~N~o", cmCancel,
                          bfNormal ));
```

```
    deskTop->execView( pd );
    destroy(pd);

 } }

char * TNeuApp::FileOpen(char *openName, char *titleName)
{

    strcpy( fileName, openName );

    if( execDialog( new TFileDialog( openName, titleName,
              "~N~ame", fdOpenButton, 100 ), fileName) != cmCancel )

     return fileName;
     else return 0;
     }

void TNeuApp::SaasNeu()
{
   char fileName[MAXPATH];
    strcpy( fileName, "*.w" );

    if( execDialog( new TFileDialog( "*.w", "Save Neural Net As",
              "~N~ame", fdOKButton,100 ), fileName) != cmCancel )
     filewritec(first1,second,third,fileName);
 }

void TNeuApp::fileNew()
{
    TEditWindow* window =
    (TEditWindow*)
     validView(new TEditWindow(deskTop->getExtent(),
              0, ++windowNumber) );
    if (window)
      deskTop->insert(window);

    if (windowNumber > 1) {
      // more than one window open:
      // enable tiling and cascading
     TCommandSet commands;
      commands += cmTile;
      commands += cmCascade;
       enableCommands(commands);
     }
   }

void TNeuApp::fileOpen()
{
    char fileName [132];

    TFileDialog* dialog =
    (TFileDialog*) validView(new TFileDialog("*.*",
          "Open file",
          "~N~ame",
          fdOpenButton,
          100) );

    if (dialog) {
      int result = execView(dialog);
      if (result != cmCancel) {
            dialog->getFileName(fileName);
            TRect r = deskTop->getExtent();
        TEditWindow *window =
        (TEditWindow*)
         validView(new TEditWindow(deskTop->getExtent(),
                fileName,
                ++windowNumber) );
        deskTop->insert(window);
        destroy(dialog);
       }
     }
   }

void TNeuApp::changeDirectory()
{
    TChDirDialog* dialog =
    (TChDirDialog*)
    validView(new TChDirDialog(cdNormal, 0) );
    if (dialog) {
     execView(dialog);
```

```
    destroy(dialog);
  }
}

void TNeuApp::dosShell()
{
   // suspend all TurboVision event handling
   suspend();

   // clear the screen
   system("cls");

   // give reentry instructions
   cout << "Type EXIT to return...";

   // invoke a new COMMAND.COM shell, no
   // matter where it's located
   system(getenv("COMSPEC"));

   // restart the TurboVision event handling
   resume();

   // ... and repaint the TurboVision screen
   redraw();
}

void TNeuApp::about()
{
   TDialog *pd = new TDialog( TRect( 18, 5, 62, 17), "About" );
   if (!pd)
    return;

   // add some centered text to the dialog box
   pd->insert(new TStaticText(TRect(3, 1, 41, 8),
                        "\n\003Feedforward Neural Network"
                         "\n\n\n\003Version 1.0"
                         "\n\n\n\n\003Copyright(c) 1994 By AMT Research Lab") );

   // add an OK button
   pd->insert(new TButton(TRect(15, 9, 27, 11),
                        " OK", cmOK, bfDefault) );
   deskTop->execView(pd);
   destroy(pd);
}

/*-------------------------------------------------------------------------------------------
                        System Set-up Window
-------------------------------------------------------------------------------------------*/

void TNeuApp::SetUp()
{
   TDialog *pd = new TDialog( TRect(20, 5, 60, 19), "System Setup" );
   if( pd )
   {
    pd->insert(new TStaticText(TRect(5, 2, 34, 3),
                        "Number of the Hidden Neurons"
              ) );
   TView* b = new TTInputLine( TRect( 23, 3, 34, 4 ), 10 );
   pd->insert( b );
   pd->insert( new TLabel( TRect( 4, 3, 23, 4 ),
                        "First Layer(1-30):", b ));

   b = new TTInputLine( TRect( 24, 4, 34, 5 ), 10 );
   pd->insert( b );
   pd->insert( new TLabel( TRect( 4, 4, 24, 5 ),
                        "Second Layer(0-30):", b ));

   b = new TTInputLine( TRect( 23, 5, 34, 6 ), 10 );
   pd->insert( b );
   pd->insert( new TLabel( TRect( 4, 5, 23, 6 ),
                        "Third Layer(0-30):", b ));


   b = new TTInputLine( TRect( 28, 7, 34, 8 ), 10 );
   pd->insert( b );
   pd->insert( new TLabel( TRect( 4, 7, 28, 8 ),
                        "Momentum Factor(0-0.9):", b ));

   b = new TTInputLine( TRect( 26, 8, 34, 9 ), 10 );
```

```
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 8, 26, 9 ),
                    "Learning Rate(0-0.9):", b ));

b = new TTInputLine( TRect( 21, 9, 34, 10 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 9, 21, 10 ),
                    "Error Threashod:", b ));

pd->insert( new TButton( TRect( 9, 11, 19, 13 ), "~O~K", cmOK,
                      bfDefault ));
 pd->insert( new TButton( TRect( 21, 11, 31, 13 ), "~C~ancel", cmCancel,
bfNormal ));

pd->setData(setupData);
int control=deskTop->execView( pd );
        if( control != cmCancel )
{
 pd->getData(setupData);
  first1=atoi(setupData->FstInputLine);
  second=atoi(setupData->SndInputLine);
  third=atoi(setupData->TrdInputLine);
     mom=atof(setupData->MomInputLine);
     lea=atof(setupData->LeaInputLine);
     err1=atof(setupData->ErrInputLine);

    }
    }

    destroy(pd);
}


/*-------------------------------------------------------------------------------
                    Training Process Monitor

-------------------------------------------------------------------------------
*/
```

```
void TNeuApp::Training()
{
 char inp1[MAXPATH];
 char outp1[MAXPATH];
 char *ap;
 n=0;p=1;
 mxer=10.0;
 sum=0;
 if((ap=FileOpen("*.neu","open training input file"))!=0)
 {
 strcpy( inp1,ap );
 if((ap=FileOpen("*.neu","open training output file"))!=0)
 {
 strcpy( outp1,ap );
  memory1(first1,second,third,inp1,outp1);
  fileopenc(first1,second,third,inp1,outp1);

 TrainDialog *pd = new TrainDialog( TRect(19, 5, 61, 19), "Training" );
  if( pd )
  {
    pd->insert(new TStaticText(TRect(8, 4, 22, 5),
                        "Current Epoch"
                ) );
  pd->insert(new TStaticText(TRect(8, 6, 22, 7),
                        "Current Error"
                ) );
 pd->insert( new TButton( TRect( 1, 9, 11, 11 ), "~T~rain", cmTrain,
                      bfDefault ));
 pd->insert( new TButton( TRect( 11, 9, 21, 11 ), "~S~top", cmStop,
bfNormal ));
 pd->insert( new TButton( TRect( 21, 9, 31, 11 ), "~O~K", cmOK,
                      bfDefault ));
 pd->insert( new TButton( TRect( 31, 9, 41, 11 ), "~C~ancel", cmCancel,
bfNormal ));


int control=deskTop->execView( pd );
        if( control != cmCancel )
          {
           SaveD();
```

```
            }
    memfreec(first1,second,third);
   destroy(pd);
}}}
}

ushort errorHandler(int message_type, ... )
{
    va_list arg;

   char buffer [80];
    ostrstream text(buffer, sizeof(buffer) );

    switch (message_type) {

     case edOutOfMemory:
       return messageBox( "Insufficient memory",
                          mfError | mfOKButton );
    case edReadError:
           va_start(arg, message_type);
          text << "Error reading file "
           << va_arg(arg, char*)
           << "."
           << ends;
         va_end(arg);
         return messageBox(buffer, mfError | mfOKButton);

 case edWriteError:
           va_start(arg, message_type);
          text << "Error writing file "
           << va_arg(arg, char*)
           << "."
           << ends;
          va_end(arg);
          return messageBox(buffer, mfError | mfOKButton);

case edCreateError:
         va_start(arg, message_type);
         text << "Error creating file "
          << va_arg(arg, char*)
           << "."
           << ends;
         return messageBox(buffer, mfError | mfOKButton);

case edSaveModify:
         va_start(arg, message_type);
         text << va_arg(arg, char*)
          << " has been modified. "
          << "Do you wish to save it ?"
          << ends;
         return messageBox(buffer,
                    mfInformation | mfYesNoCancel);

case edSaveUntitled:
         return messageBox("Save untitled file ?",
                    mfInformation | mfYesNoCancel);

case edSaveAs:
    {
     va_start( arg, dialog );
     return execDialog( new TFileDialog( "*.*",
                          "Save file as",
                          "~N~ame",
                           fdOKButton, 101 ), va_arg( arg, char* ) );
    }
   default:
    return 0;
   }
}


/*-----------------------------------------------------------------------------
                                Test Window

-----------------------------------------------------------------------------*/


const int maxLineLengthW = maxViewWidth+1;
const int maxLines      = 212;
char *lines[maxLines];
```

```
int lineCount = 0;
static short winNumber = 0;


void readFile( const char *fileName )
{
   ifstream fileToView( fileName );
   if( !fileToView )
           {
           cout << "Invalid file name..." << endl;
           exit( 1 );
           }
   else
           {
           char buf[maxLineLengthW];

           while( lineCount < maxLines &&
                fileToView.getline( buf, maxLineLengthW ) != 0 )
             {
              lines[lineCount] = newStr( buf );
              lineCount++;
             }
           }
}

void deleteFile()
{
   for( int i = 0; i < lineCount; i++ )
           delete lines[i];
}

class TTestWindow : public TWindow   // define a new window class
{

public:
    TTestWindow( const TRect& bounds, const char *aTitle, short aNumber );
   void makeInterior();
};

class TInterior : public TScroller
{
public:
   TInterior( const TRect& bounds, TScrollBar *aHScrollBar,
              TScrollBar *aVScrollBar ); // constructor
  virtual void draw();           // override TView::draw
};

TInterior::TInterior( const TRect& bounds, TScrollBar *aHScrollBar,
                TScrollBar *aVScrollBar ) :
     TScroller( bounds, aHScrollBar, aVScrollBar )
{
  growMode = gfGrowHiX | gfGrowHiY;
   options = options | ofFramed;
   setLimit( maxLineLengthW, maxLines );
}

void TInterior::draw()       // modified for scroller
{
   ushort color = getColor(0x0301);
  for( int i = 0; i < size.y; i++ )
          // for each line:
{
TDrawBuffer b;
b.moveChar( 0, ' ', color, size.x );
// fill line buffer with spaces
int j = delta.y + i;       // delta is scroller offset
if( lines[j] )
  {
   char s[maxLineLengthW];
   if( delta.x > strlen(lines[j] ) )
           s[0] = EOS;
  else
           {
           strncpy( s, lines[j]+delta.x, size.x );
           s[size.x] = EOS;
           }
   b.moveStr( 0, s, color );
  }
writeLine( 0, i, size.x, 1, b);
}
```

```
}

void TTestWindow::makeInterior()
{
   TScrollBar *vScrollBar =
         standardScrollBar( sbVertical | sbHandleKeyboard );
   TScrollBar *hScrollBar =
         standardScrollBar( sbHorizontal |  sbHandleKeyboard );
   TRect r = getClipRect();    // get exposed view bounds
   r.grow( -1, -1 );           // shrink to fit inside window frame
    insert( new TInterior( r, hScrollBar, vScrollBar ));
}

void TNeuApp::Win(char * outName)
{
  readFile( outName);
   TRect r( 8, 4, 72, 20 );           // set initial size and position
   TTestWindow *window = new TTestWindow ( r, "Test Result Window",
++winNumber );
   deskTop->insert(window);    // put window into desktop and draw it
}

TTestWindow::TTestWindow( const TRect& bounds, const char *aTitle,
            short aNumber) :
          TWindow( bounds, aTitle, aNumber),
          TWindowInit( &TTestWindow::initFrame )
{
   makeInterior(); // creates scrollable interior and inserts into window
}


/*-------------------------------------------------------------------------------
                              Main Programme

-------------------------------------------------------------------------------
*/

int main()
{
   TNeuApp myApp;
   myApp.run();
   deleteFile();
   return 0;
}
```

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

                Feedforward Neural Network
                     Version 1.0

          Copyright ©1994 AMT Research Laboratory

*************************************************************/

/*************************************************************

          Source file 2:        Network

/*************************************************************/

  #include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <dos.h>
#include <STRING.H>

#ifdef __cplusplus

  float max (float value1, float value2);

  float max(float value1, float value2)
 {
    return ( (value1 > value2) ? value1 : value2);
 }

#endif

void memory1(int,int,int,char *, char *);
void fileopenc(int hi1,int hi2,int hi3,char *,char *);
float Trainingc(int hi1,int hi2,int hi3,float cb,float ca);
void filewritec(int hi1,int hi2,int hi3,char *);
void memfreec(int,int,int);

 int  i,j,k,h,hi3a,x;int  in,out,ch,r2,m2,m,s;

 FILE *fp;

/*-------------------------------------------------------------------------------------------
                              Dynamically Arrange Memory
-------------------------------------------------------------------------------------------*/


  float  **intra,**outtr,*bh1,*bh2,*bh3,*bo,**winh1,**wh1h2,**wh2h3,**wh3o,
  **err,**abserr,**dth3,**dth2,**dth1,**dto,**dtw,*dbi,*dbh3_1,*dbo_1,*dtm,
  **dtow_1,**dtiw_1,**dtiw_2,**dtiw_3,*dbh1_1,*dbh2_1,  **net_h1,  **oh1,
    **net_h2,  **oh2,**net_h3,**oh3,**net_out,**outn,  *eveerr;

void memory1(int hi1,int hi2,int hi3,char *inp1, char *outp1)
{
 if(hi2==0)
 hi3a=hi1;
 else
 {
 if(hi3==0)
 hi3a=hi2;
 else
 hi3a=hi3;
 }
 in=0;k=0;r2=0;out=0;
 if((fp=fopen(inp1,"r"))==NULL)
 {
 putch(7);exit(0);
 }
   for(j=0;j<=30000-1;j++)
 {
    ch=fgetc(fp);
  if(ch==44||ch==10)
  {
  if(ch==10)
```

```
 k=k+1;
 else
 in=in+1;
 }
 if(feof(fp)!=0)
 break;
 }
 fclose(fp);
 in=in/k;

 if((fp=fopen(outp1,"r"))==NULL)
{
getch();exit(0);
}
 for(j=0;j<=30000-1;j++)
{
 ch=fgetc(fp);
 if(ch==44||ch==10)
 {
 if(ch==10)
 r2=r2+1;
 else
 out=out+1;
 }
 if(feof(fp)!=0)
 break;}
 fclose(fp);
 out=out/r2;
m=(int)max(hi1,hi2);
m=(int)max(m,hi3);
s=(int)max(m,in);
s=(int)max(s,out);
intra=(float**)calloc(in,sizeof(float*));
for(i=0;i<=in-1;i++)    *(intra+i)=(float *)calloc(k,sizeof(float));
outtr=(float**)calloc(out,sizeof(float*));
for(i=0;i<=out-1;i++)    *(outtr+i)=(float *)calloc(k,sizeof(float));

bh1=(float*)calloc(hi1,sizeof(float));
if(hi2!=0)
bh2=(float*)calloc(hi2,sizeof(float));
if(hi3!=0)
 bh3=(float*)calloc(hi3,sizeof(float));
 bo =(float*)calloc(out,sizeof(float));

 winh1=(float**)calloc(in,sizeof(float*));
 for(i=0;i<=in-1;i++)    *(winh1+i)=(float *)calloc(hi1,sizeof(float));
 dtiw_3=(float**)calloc(in,sizeof(float*));
 for(i=0;i<=in-1;i++)    *(dtiw_3+i)=(float *)calloc(hi1,sizeof(float));

 if(hi2!=0){
 wh1h2=(float**)calloc(hi1,sizeof(float*));
 for(i=0;i<=hi1-1;i++)    *(wh1h2+i)=(float *)calloc(hi2,sizeof(float));
 dtiw_2=(float**)calloc(hi1,sizeof(float*));
 for(i=0;i<=hi1-1;i++)    *(dtiw_2+i)=(float *)calloc(hi2,sizeof(float));
 }
 if(hi3!=0){
 wh2h3=(float**)calloc(hi2,sizeof(float*));
 for(i=0;i<=hi2-1;i++)    *(wh2h3+i)=(float *)calloc(hi3,sizeof(float));
 dtiw_1=(float**)calloc(hi2,sizeof(float*));
 for(i=0;i<=hi2-1;i++)    *(dtiw_1+i)=(float *)calloc(hi3,sizeof(float));
 }
 wh3o=(float**)calloc(hi3a,sizeof(float*));
 for(i=0;i<=hi3a-1;i++)    *(wh3o+i)=(float *)calloc(out,sizeof(float));
 dtow_1=(float**)calloc(hi3a,sizeof(float*));
 for(i=0;i<=hi3a-1;i++)    *(dtow_1+i)=(float *)calloc(out,sizeof(float));

 err=(float**)calloc(out,sizeof(float*));
 for(i=0;i<=out-1;i++)    *(err+i)=(float *)calloc(k,sizeof(float));
 aberr=(float**)calloc(out,sizeof(float*));
 for(i=0;i<=out-1;i++)    *(aberr+i)=(float *)calloc(k,sizeof(float));

 dth3=(float**)calloc(hi3a,sizeof(float*));
 for(i=0;i<=hi3a-1;i++)    *(dth3+i)=(float *)calloc(k,sizeof(float));
 dto=(float**)calloc(out,sizeof(float*));
 for(i=0;i<=out-1;i++)    *(dto+i)=(float *)calloc(k,sizeof(float));
 dth2=(float**)calloc(m,sizeof(float*));
 for(i=0;i<=m-1;i++)    *(dth2+i)=(float *)calloc(k,sizeof(float));
 dth1=(float**)calloc(m,sizeof(float*));
 for(i=0;i<=m-1;i++)    *(dth1+i)=(float *)calloc(k,sizeof(float));
 dtw=(float**)calloc(s,sizeof(float*));
```

```
for(i=0;i<=s-1;i++)    *(dtw+i)=(float *)calloc(s,sizeof(float));
dbi=(float*)calloc(s,sizeof(float));
if(hi3!=0) dbh3_1=(float*)calloc(hi3,sizeof(float));
if(hi2!=0) dbh2_1=(float*)calloc(hi2,sizeof(float));
dbh1_1=(float*)calloc(hi1,sizeof(float));
dbo_1=(float*)calloc(out,sizeof(float));
dtm=(float*)calloc(k,sizeof(float));
net_h1=(float**)calloc(hi1,sizeof(float*));
for(i=0;i<=hi1-1;i++)    *(net_h1+i)=(float *)calloc(k,sizeof(float));
if(hi2!=0){
net_h2=(float**)calloc(hi2,sizeof(float*));
for(i=0;i<=hi2-1;i++)    *(net_h2+i)=(float *)calloc(k,sizeof(float));
}
if(hi3!=0){
net_h3=(float**)calloc(hi3,sizeof(float*));
for(i=0;i<=hi3-1;i++)
 *(net_h3+i)=(float *)calloc(k,sizeof(float));
}
oh1=(float**)calloc(hi1,sizeof(float*));
for(i=0;i<=hi1-1;i++)    *(oh1+i)=(float *)calloc(k,sizeof(float));
oh2=(float**)calloc(m,sizeof(float*));
for(i=0;i<=m-1;i++)    *(oh2+i)=(float *)calloc(k,sizeof(float));
oh3=(float**)calloc(m,sizeof(float*));

   for(i=0;i<=m-1;i++)    *(oh3+i)=(float *)calloc(k,sizeof(float));
net_out=(float**)calloc(out,sizeof(float*));
for(i=0;i<=out-1;i++)    *(net_out+i)=(float *)calloc(k,sizeof(float));
outn=(float**)calloc(out,sizeof(float*));
for(i=0;i<=out-1;i++)    *(outn+i)=(float *)calloc(k,sizeof(float));
eveerr=(float*)calloc(k,sizeof(float));

if(outn==NULL)
{
 printf("Not enough memory");
 getch();
 exit(0);
 }
}
```

```
/*-----------------------------------------------------------------------------------------------
                        Read Data from Data Files
-----------------------------------------------------------------------------------------------*/


void fileopenc(int hi1,int hi2,int hi3, char *inp1,char *outp1)
{
 if((fp=fopen(inp1,"r"))==NULL)
 {
 getch();exit(0);
 }
 for(i=0;i<=k-1;i++)
 {
  for(j=0;j<=in-1;j++)
  {
   fscanf(fp,"%e,",&intra[j][i]);
   //printf("%e, ",intra[j][i]);
  }
  if(feof(fp)!=0)
  break;
 }
 fclose(fp);

 if((fp=fopen(outp1,"r"))==NULL)
 {
 getch();exit(0);
 }
 for(i=0;i<=k-1;i++)
 {
  for(j=0;j<=out-1;j++)
  {
   fscanf(fp,"%e,",&outtr[j][i]);
  // printf("%e, ",outtr[j][i]);
  }
 }
 fclose(fp);
 if(hi2==0)
 hi3a=hi1;
 else
```

```
{
if(hi3==0)
 hi3a=hi2;
else
 hi3a=hi3;
}
randomize();

  for(i=0;i<=in-1;i++)
{
 for(j=0;j<=hi1-1;j++)
 {
   winh1[i][j]=1.0/(x=random(100),x+100);
  dtiw_3[i][j]=0;
 }
}
 for(i=0;i<=hi1-1;i++)
{
 for(j=0;j<=hi2-1;j++)
 {
   wh1h2[i][j]=1.0/(x=random(100),x+100);
  dtiw_2[i][j]=0;
 }
  dbh1_1[i]=0;
  bh1[i]=1.0/(x=random(100),x+100);
}
 for(i=0;i<=hi2-1;i++)
{
 for(j=0;j<=hi3-1;j++)
 {
   wh2h3[i][j]=1.0/(x=random(100),x+100);
  dtiw_1[i][j]=0;
  dbh3_1[j]=0;
   bh3[j]=1.0/(x=random(100),x+100);
}
 dbh2_1[i]=0;
  bh2[i]=1.0/(x=random(100),x+100);
}
 for(i=0;i<=hi3a-1;i++)
{
  for(j=0;j<=out-1;j++)
  {
    wh3o[i][j]=1.0/(x=random(100),x+100);
    bo[j]=1.0/(x=random(100),x+100);
   dtow_1[i][j]=0;
   dbo_1[j]=0;
  }
 }
 }


/*-------------------------------------------------------------------------------------------
                                    Training Module

-------------------------------------------------------------------------------------------*/


float Trainingc(int hi1,int hi2,int hi3,float cb,float ca)
{
 float mxer;
  // do
  {
 mxer=0.0;
  for(i=0;i<=k-1;i++)
  {
   for(j=0;j<=hi1-1;j++)
   {
    net_h1[j][i]=0;
    for(h=0;h<=in-1;h++)
    {
      net_h1[j][i]=intra[h][i]*winh1[h][j]+net_h1[j][i];
    }
    net_h1[j][i]=net_h1[j][i]+bh1[j];
    oh1[j][i]=1.0/(1.0+exp(-net_h1[j][i]));
   }
  }
  hi3a=hi3;
  if(hi2==0)
  {
   for(i=0;i<=k-1;i++)
```

```
{
 for(j=0;j<=hi1-1;j++)
 {
  oh3[j][i]=oh1[j][i];
  hi3a=hi1;
 }
 }
}
else
{

for(i=0;i<=k-1;i++)
{
 for(j=0;j<=hi2-1;j++)
 {
  net_h2[j][i]=0.0;
  for(h=0;h<=hi1-1;h++)
  {
    net_h2[j][i]=oh1[h][i]*wh1h2[h][j]+net_h2[j][i];
  }
  net_h2[j][i]=net_h2[j][i]+bh2[j];
    oh2[j][i]=1.0/(1.0+exp(-net_h2[j][i]));
  }
 }
 if(hi3==0)
 {
  for(i=0;i<=k-1;i++)
  {
   for(j=0;j<=hi2-1;j++)
   {
    oh3[j][i]=oh2[j][i];
    hi3a=hi2;
   }
  }
}
else
{
 for(i=0;i<=k-1;i++)
 {
  for(j=0;j<=hi3-1;j++)
  {
   net_h3[j][i]=0.0;
   for(h=0;h<=hi2-1;h++)
   {
     net_h3[j][i]=oh2[h][i]*wh2h3[h][j]+net_h3[j][i];
   }
    net_h3[j][i]=net_h3[j][i]+bh3[j];
    oh3[j][i]=1.0/(1.0+exp(-net_h3[j][i]));
  }
 }
}
}
 for(i=0;i<=k-1;i++)
 { eveerr[i]=0;
  for(j=0;j<=out-1;j++)
  {
   net_out[j][i]=0.0;
   for(h=0;h<=hi3a-1;h++)
   {
     net_out[j][i]=oh3[h][i]*wh3o[h][j]+net_out[j][i];
   }
    net_out[j][i]=net_out[j][i]+bo[j];
    outn[j][i]=1.0/(1.0+exp(-net_out[j][i]));
   err[j][i]=outtr[j][i]-outn[j][i];
   abserr[j][i]=err[j][i]*err[j][i];
   eveerr[i]=eveerr[i]+abserr[j][i];

    dto[j][i]=outn[j][i]*(1-outn[j][i])*err[j][i];
  }
   eveerr[i]=sqrt(eveerr[i]/(out));
   mxer=mxer+eveerr[i];
 }
 mxer=mxer/k;
 for(i=0;i<=k-1;i++)
 {
  dtm[i]=0.0;
  for(j=0;j<=hi3a-1;j++)
  {
   for(h=0;h<=out-1;h++)
   {
```

```
   dtm[i]=dto[h][i]*wh3o[j][h]+dtm[i];
  }
    dth3[j][i]=oh3[j][i]*(1-oh3[j][i])*dtm[i];
 }
}
 if(hi3==0)
 {
  for(i=0;i<=k-1;i++)
  {
   for(j=0;j<=hi2-1;j++)
   {
    dth2[j][i]=dth3[j][i];
   }
  }
 }
 else
{
 for(i=0;i<=k-1;i++)
 {
  dtm[i]=0.0;
  for(j=0;j<=hi2-1;j++)
  {
   for(h=0;h<=hi3-1;h++)
   {
    dtm[i]=dth3[h][i]*wh2h3[j][h]+dtm[i];
   }
   dth2[j][i]=oh2[j][i]*(1-oh2[j][i])*dtm[i];
 }
}
}
if(hi2==0)
{
 for(i=0;i<=k-1;i++)
 {
  for(j=0;j<=hi1-1;j++)
  {
   dth1[j][i]=dth3[j][i];
  }
 }
}
 else
{
 for(i=0;i<=k-1;i++)
 {
  dtm[i]=0.0;
  for(j=0;j<=hi1-1;j++)
  {
   for(h=0;h<=hi2-1;h++)
   {
    dtm[i]=dth2[j][i]*wh1h2[j][h]+dtm[i];
   }
   dth1[j][i]=oh1[j][i]*(1-oh1[j][i])*dtm[i];
  }
 }
}

  for(j=0;j<=hi3a-1;j++)
 {
  for(i=0;i<=out-1;i++)
  {
   dtw[j][i]=0.0;
   for(h=0;h<=k-1;h++)
   {
     dtw[j][i]=dto[i][h]*oh3[j][h]+dtw[j][i];
   }
    wh3o[j][i]=wh3o[j][i]+ca*dtw[j][i]+cb*dtow_1[j][i];
    dtow_1[j][i]=ca*dtw[j][i]+cb*dtow_1[j][i];
  }
 }
 for(j=0;j<=hi2-1;j++)
 {
  for(i=0;i<=hi3-1;i++)
  {
   dtw[j][i]=0.0;
   for(h=0;h<=k-1;h++)
   {
    dtw[j][i]=dth3[i][h]*oh2[j][h]+dtw[j][i];
   }
    wh2h3[j][i]=wh2h3[j][i]+ca*dtw[j][i]+cb*dtiw_1[j][i];
    dtiw_1[j][i]=ca*dtw[j][i]+cb*dtiw_1[j][i];
```

```
 }
}
for(j=0;j<=hi1-1;j++)
{
 for(i=0;i<=hi2-1;i++)
 {
  dtw[j][i]=0.0;
  for(h=0;h<=k-1;h++)
  {
    dtw[j][i]=dth2[i][h]*oh1[j][h]+dtw[j][i];
  }
   wh1h2[j][i]=wh1h2[j][i]+ca*dtw[j][i]+cb*dtiw_2[j][i];
   dtiw_2[j][i]=ca*dtw[j][i]+cb*dtiw_2[j][i];
 }
}
 for(j=0;j<=in-1;j++)
{
 for(i=0;i<=hi1-1;i++)
 {
  dtw[j][i]=0.0;
   for(h=0;h<=k-1;h++)
   {
     dtw[j][i]=dth1[i][h]*intra[j][h]+dtw[j][i];
   }
    winh1[j][i]=winh1[j][i]+ca*dtw[j][i]+cb*dtiw_3[j][i];
    dtiw_3[j][i]=ca*dtw[j][i]+cb*dtiw_3[j][i];
 }
}
for(i=0;i<=out-1;i++)
{
 dbi[i]=0.0;
 for(j=0;j<=k-1;j++)
 {
  dbi[i]=dto[i][j]+dbi[i];
 }
  bo[i]=bo[i]+ca*dbi[i]+cb*dbo_1[i];
  dbo_1[i]=ca*dbi[i]+cb*dbo_1[i];
}
for(i=0;i<=hi3-1;i++)
{
 dbi[i]=0.0;
 for(j=0;j<=k-1;j++)
 {
  dbi[i]=dth3[i][j]+dbi[i];
 }
  bh3[i]=bh3[i]+ca*dbi[i]+cb*dbh3_1[i];
  dbh3_1[i]=ca*dbi[i]+cb*dbh3_1[i];
}
 for(i=0;i<=hi2-1;i++)
{
 dbi[i]=0.0;
  for(j=0;j<=k-1;j++)
 {
  dbi[i]=dth2[i][j]+dbi[i];
 }
   bh2[i]=bh2[i]+ca*dbi[i]+cb*dbh2_1[i];
   dbh2_1[i]=ca*dbi[i]+cb*dbh2_1[i];

}
 for(i=0;i<=hi1-1;i++)
 {
  dbi[i]=0.0;
  for(j=0;j<=k-1;j++)
  {
   dbi[i]=dth1[i][j]+dbi[i];
  }
    bh1[i]=bh1[i]+ca*dbi[i]+cb*dbh1_1[i];
    dbh1_1[i]=ca*dbi[i]+cb*dbh1_1[i];
  }
 }
 return(mxer);
}

/*-------------------------------------------------------------------------------------
                        Write Data to Data File

-------------------------------------------------------------------------------------*/

void filewritec(int hi1,int hi2,int hi3,char *weight)
{
```

```
if((fp=fopen(weight,"w"))==NULL)
{
printf("Cannot open weight.txt file\n");
getch();exit(0);
}
fprintf(fp,"%d %d %d %d %d %d ",in,hi1,hi2,hi3,hi3a,out);
for(j=0;j<=hi1-1;j++)
{
for(h=0;h<=in-1;h++)
{
fprintf(fp,"%e ",winh1[h][j]);
}
for(i=0;i<=hi1-1;i++)
{
fprintf(fp,"%e ",bh1[i]);
}
for(j=0;j<=hi2-1;j++)
{
for(h=0;h<=hi1-1;h++)
{
fprintf(fp,"%e ",wh1h2[h][j]);
}
for(j=0;j<=hi3-1;j++)
{
for(h=0;h<=hi2-1;h++)
{
fprintf(fp,"%e ",wh2h3[h][j]);
}
for(j=0;j<=hi3a-1;j++)
{
for(h=0;h<=hi3-1;h++)
{
fprintf(fp,"%e ",wh3oh[h][j]);
}
for(i=0;i<=out-1;i++)
{
fprintf(fp,"%e ",bo[i]);
}
fclose(fp);
}

/*------------------------------------------------------------------------------
                        Free Memory
------------------------------------------------------------------------------*/

void memfreec(int hi1,int hi2,int hi3)
{
for(i=0;i<=in-1;i++) free(*(inta+i));free(inta);
for(i=0;i<=out-1;i++)free(*(outa+i));free(outa);
free(bh1);free(dbh1_1);
if(hi2!=0){free(bh2);free(dbh2_1);}
if(hi3!=0){free(bh3);free(dbh3_1);}
free(bo);
for(i=0;i<=in-1;i++)free(*(winh1+i));free(winh1);
for(i=0;i<=in-1;i++)free(*(diw_3+i));free(diw_3);
if(hi2!=0){
for(i=0;i<=hi1-1;i++)free(*(wh1h2+i));free(wh1h2);
for(i=0;i<=hi1-1;i++)free(*(diw_2+i));free(diw_2);
}
if(hi3!=0){
for(i=0;i<=hi2-1;i++)free(*(wh2h3+i));free(wh2h3);
for(i=0;i<=hi2-1;i++)free(*(diw_1+i));free(diw_1);
}
for(i=0;i<=hi3a-1;i++)free(*(wh3o+i));free(wh3o);
for(i=0;i<=hi3a-1;i++)free(*(diow_1+i));free(diow_1);
```

```
for(i=0;i<=out-1;i++)free(*(err+i));free(err);
for(i=0;i<=out-1;i++)free(*(abserr+i));free(abserr);
for(i=0;i<=hi3a-1;i++)free(*(dth3+i));free(dth3);
for(i=0;i<=m-1;i++)free(*(dth2+i));free(dth2);
for(i=0;i<=m-1;i++)free(*(dth1+i));free(dth1);
for(i=0;i<=out-1;i++)free(*(dto+i)); free(dto);
for(i=0;i<=s-1;i++)free(*(dtw+i));free(dtw);
free(dbi);
free(dbo_1);free(dtm);
for(i=0;i<=hi1-1;i++)free(*(net_h1+i));free(net_h1);
if(hi2!=0){for(i=0;i<=hi1-1;i++)free(*(net_h2+i));free(net_h2);}
if(hi3!=0){ for(i=0;i<=hi1-1;i++)free(*(net_h3+i));free(net_h3);}
for(i=0;i<=hi1-1;i++)free(*(oh1+i));free(oh1);
for(i=0;i<=m-1;i++)free(*(oh2+i));free(oh2);
for(i=0;i<=m-1;i++)free(*(oh3+i));free(oh3);
for(i=0;i<=out-1;i++)free(*(net_out+i));free(net_out);
for(i=0;i<=out-1;i++)free(*(outn+i));free(outn);
}
/*-------------------------------------------------------------------------------
                        Test Module
-------------------------------------------------------------------------------*/

char *Test(char*inp1,char*outp1,char*weight)
{
 int  i,j,h,k,in,hi1,hi2,hi3,hi3a,out,x,m,ch;

FILE *fp;
 float  **intest,*net_h1,*oh1,*net_h2,*oh2,*net_h3,*oh3,
 *net_out,**outtest,**winh1,**wh1h2,**wh2h3,**wh3o,
 *bh1,*bh2,*bh3,*bo;

k=0;
if((fp=fopen(inp1,"r"))==NULL)
{ printf("Cannot open weight.txt file\n");
  putch(7);exit(0);
}
  for(j=0;j<=30000-1;j++)
 {
  ch=fgetc(fp);
  if(ch==44||ch==10)
 {
  if(ch==10)
  k=k+1;
  else
  in=in+1;
 }
  if(feof(fp)!=0)
  break;}
  fclose(fp);
if((fp=fopen(weight,"r"))==NULL)
{
printf("Cannot open weight.txt file\n");
getch();exit(0);
}
fscanf(fp,"%d %d %d %d %d %d ",&in,&hi1,&hi2,&hi3,&hi3a,&out);
m=(int)max(hi1,hi2);
m=(int)max(m,hi3);


intest=(float**)calloc(in,sizeof(float*));
for(i=0;i<=in-1;i++)    *(intest+i)=(float *)calloc(k,sizeof(float));
outtest=(float**)calloc(out,sizeof(float*));
for(i=0;i<=out-1;i++)    *(outtest+i)=(float *)calloc(k,sizeof(float));

bh1=(float*)calloc(hi1,sizeof(float));
if(hi2!=0)
bh2=(float*)calloc(hi2,sizeof(float));
if(hi3!=0)
bh3=(float*)calloc(hi3,sizeof(float));
bo =(float*)calloc(out,sizeof(float));

winh1=(float**)calloc(in,sizeof(float*));
for(i=0;i<=in-1;i++)    *(winh1+i)=(float *)calloc(hi1,sizeof(float));

if(hi2!=0){
```

```
 wh1h2=(float**)calloc(hi1,sizeof(float*));
for(i=0;i<=hi1-1;i++)    *(wh1h2+i)=(float *)calloc(hi2,sizeof(float));
}

if(hi3!=0){
 wh2h3=(float**)calloc(hi2,sizeof(float*));
for(i=0;i<=hi2-1;i++)    *(wh2h3+i)=(float *)calloc(hi3,sizeof(float));
}

 wh3o=(float**)calloc(hi3a,sizeof(float*));
for(i=0;i<=hi3a-1;i++)    *(wh3o+i)=(float *)calloc(out,sizeof(float));

 net_h1=(float*)calloc(hi1,sizeof(float));

if(hi2!=0)
 net_h2=(float*)calloc(hi2,sizeof(float));

if(hi3!=0)
 net_h3=(float*)calloc(hi3,sizeof(float));

 oh1=(float*)calloc(hi1,sizeof(float));
 oh2=(float*)calloc(m,sizeof(float));
 oh3=(float*)calloc(m,sizeof(float));

 net_out=(float*)calloc(out,sizeof(float));

if(net_out==NULL)
{
 printf("Not enough memory");
 getch();
 exit(0);
}

for(j=0;j<=hi1-1;j++)
{
 for(h=0;h<=in-1;h++)
 {
  fscanf(fp,"%e ",&winh1[h][j]);
 }
}
for(j=0;j<=hi2-1;j++)
{
 for(h=0;h<=hi1-1;h++)
 {
  fscanf(fp,"%e ",&wh1h2[h][j]);
 }
}
for(j=0;j<=hi3-1;j++)
{
 for(h=0;h<=hi2-1;h++)
 {
   fscanf(fp,"%e ",&wh2h3[h][j]);
 }
}
for(j=0;j<=out-1;j++)
{
 for(h=0;h<=hi3a-1;h++)
 {
   fscanf(fp,"%e ",&wh3o[h][j]);
 }
}
 for(i=0;i<=hi1-1;i++)
 {
  fscanf(fp,"%e ",&bh1[i]);
 }
 for(i=0;i<=hi2-1;i++)
 {
   fscanf(fp,"%e ",&bh2[i]);
 }
  for(i=0;i<=hi3-1;i++)
 {
   fscanf(fp,"%e ",&bh3[i]);
 }
  for(i=0;i<=out-1;i++)
 {
   fscanf(fp,"%e ",&bo[i]);
 }
fclose(fp);
if((fp=fopen(inp1,"r"))==NULL)
{
```

```
printf("Cannot open input file\n");
getch();exit(0);
}
for(h=0;h<=k-1;h++)
{
 for(j=0;j<=in-1;j++)
{  fscanf(fp,"%e,",&intest[j][h]); }
 if(feof(fp)!=0)
 break;
}
fclose(fp);

for(i=0;i<=k-1;i++)
{
for(j=0;j<=hi1-1;j++)
{
  net_h1[j]=0;
  for(h=0;h<=in-1;h++)
 {
    net_h1[j]=intest[h][i]*winh1[h][j]+net_h1[j];
 }
   net_h1[j]=net_h1[j]+bh1[j];
   oh1[j]=1.0/(1.0+exp(-net_h1[j]));
 }
 hi3a=hi3;
 if(hi2==0)
 {
   for(j=0;j<=hi1-1;j++)
  {
   oh3[j]=oh1[j];
   hi3a=hi1;
  }
 }
else
{
 for(j=0;j<=hi2-1;j++)
 {
   net_h2[j]=0.0;
   for(h=0;h<=hi1-1;h++)
  {
     net_h2[j]=oh1[h]*wh1h2[h][j]+net_h2[j];
  }
    net_h2[j]=net_h2[j]+bh2[j];
    oh2[j]=1.0/(1.0+exp(-net_h2[j]));
 }
  if(hi3==0)
  {
   for(j=0;j<=hi2-1;j++)
   {
    oh3[j]=oh2[j];
    hi3a=hi2;
   }
  }
 else
 {
  for(j=0;j<=hi3-1;j++)
  {
    net_h3[j]=0.0;
    for(h=0;h<=hi2-1;h++)
   {
     net_h3[j]=oh2[h]*wh2h3[h][j]+net_h3[j];
   }
    net_h3[j]=net_h3[j]+bh3[j];
    oh3[j]=1.0/(1.0+exp(-net_h3[j]));
  }
 }
 }
  for(j=0;j<=out-1;j++)
  {
   net_out[j]=0.0;
   for(h=0;h<=hi3a-1;h++)
   {
     net_out[j]=oh3[h]*wh3o[h][j]+net_out[j];
   }
    net_out[j]=net_out[j]+bo[j];
    outtest[j][i]=1.0/(1.0+exp(-net_out[j]));
 }
 }
  if((fp=fopen(outp1,"w"))==NULL)
  {
```

```
printf("Cannot open output file\n");
getch();exit(0);
}
fprintf(fp,"\n%s\n","Test Results:");
for(i=0;i<=k-1;i++)
{
 for(j=0;j<=out-1;j++)
 {  fprintf(fp,"%f,",outtest[j][i]);
 }
  fprintf(fp,"%s\n","");
}
fclose(fp);

for(i=0;i<=in-1;i++) free(*(intest+i));free(intest);
for(i=0;i<=out-1;i++)free(*(outtest+i));free(outtest);
free(bh1);
if(hi2!=0){free(bh2);}
if(hi3!=0){free(bh3);}
free(bo);
for(i=0;i<=in-1;i++)free(*(winh1+i));free(winh1);
 if(hi2!=0){
 for(i=0;i<=hi1-1;i++)free(*(wh1h2+i));free(wh1h2);
 }
 if(hi3!=0){
 for(i=0;i<=hi2-1;i++)free(*(wh2h3+i));free(wh2h3);
 }
 for(i=0;i<=hi3a-1;i++)free(*(wh3o+i));free(wh3o);
 free(net_h1);
 if(hi2!=0){free(net_h2);}
 if(hi3!=0){free(net_h3);}
 free(oh1);
 free(oh2);
 free(oh3);
 free(net_out);
 free(eveerr);
return outp1;}
```

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

                    Feedforward Neural Network
                          Version 1.0

            Copyright ©1994 AMT Research Laboratory

*************************************************************/

/*************************************************************

          Source file 3:        Application Module

/*************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <math.h>

#ifdef __cplusplus

  float max (float value1, float value2);
  float min (float value1, float value2);

  float max(float value1, float value2)
  {
    return ( (value1 > value2) ? value1 : value2);
  }
  float min(float value1, float value2)
  {
    return ( (value1 > value2) ? value1 : value2);
  }

#endif
```

```
void appl(float *inapp,float *outapp,char *weight)
{
 int  i,j,h,in,hi1,hi2,hi3,hi3a,out,m;
 FILE *fp;
 float  *net_h1,*oh1,*net_h2,*oh2,*net_h3,*oh3,  *net_out,  **winh1,  **wh1h2,
**wh2h3,  **wh3o,  *bh1,*bh2,*bh3,*bo;

 if((fp=fopen(weight,"r"))==NULL)
 {
 printf("Cannot open weight.txt file\n");
 getch();exit(0);
 }
 fscanf(fp,"%d %d %d %d %d %d ",&in,&hi1,&hi2,&hi3,&hi3a,&out);
 m=(int)max(hi1,hi2);
 m=(int)max(m,hi3);

 bh1=(float*)calloc(hi1,sizeof(float));
 if(hi2!=0)
 bh2=(float*)calloc(hi2,sizeof(float));
 if(hi3!=0)
 bh3=(float*)calloc(hi3,sizeof(float));
 bo =(float*)calloc(out,sizeof(float));

 winh1=(float**)calloc(in,sizeof(float*));
 for(i=0;i<=in-1;i++)    *(winh1+i)=(float *)calloc(hi1,sizeof(float));

 if(hi2!=0){
 wh1h2=(float**)calloc(hi1,sizeof(float*));
 for(i=0;i<=hi1-1;i++)    *(wh1h2+i)=(float *)calloc(hi2,sizeof(float));
 }

 if(hi3!=0){
 wh2h3=(float**)calloc(hi2,sizeof(float*));
 for(i=0;i<=hi2-1;i++)    *(wh2h3+i)=(float *)calloc(hi3,sizeof(float));
 }

 wh3o=(float**)calloc(hi3a,sizeof(float*));
 for(i=0;i<=hi3a-1;i++)    *(wh3o+i)=(float *)calloc(out,sizeof(float));

 net_h1=(float*)calloc(hi1,sizeof(float));
```

```
if(hi2!=0)
 net_h2=(float*)calloc(hi2,sizeof(float));

if(hi3!=0)
 net_h3=(float*)calloc(hi3,sizeof(float));

 oh1=(float*)calloc(hi1,sizeof(float));
 oh2=(float*)calloc(m,sizeof(float));
 oh3=(float*)calloc(m,sizeof(float));

 net_out=(float*)calloc(out,sizeof(float));

if(net_out==NULL)
{
 printf("Not enough memory");
 getch();
 exit(0);
}
 for(j=0;j<=hi1-1;j++)
{
 for(h=0;h<=in-1;h++)
{
  fscanf(fp,"%e ",&winh1[h][j]);
}
}
for(j=0;j<=hi2-1;j++)
{
 for(h=0;h<=hi1-1;h++)
{
  fscanf(fp,"%e ",&wh1h2[h][j]);
}
}
for(j=0;j<=hi3-1;j++)
{
 for(h=0;h<=hi2-1;h++)
{
   fscanf(fp,"%e ",&wh2h3[h][j]);
}
}
for(j=0;j<=out-1;j++)
{
 for(h=0;h<=hi3a-1;h++)
{
   fscanf(fp,"%e ",&wh3o[h][j]);
}
}
 for(i=0;i<=hi1-1;i++)
{
  fscanf(fp,"%e ",&bh1[i]);
}
 for(i=0;i<=hi2-1;i++)
{
   fscanf(fp,"%e ",&bh2[i]);
}
  for(i=0;i<=hi3-1;i++)
{
   fscanf(fp,"%e ",&bh3[i]);
}
  for(i=0;i<=out-1;i++)
{
   fscanf(fp,"%e ",&bo[i]);
}
fclose(fp);

for(j=0;j<=hi1-1;j++)
{
  net_h1[j]=0;
  for(h=0;h<=in-1;h++)
 {
    net_h1[j]=inapp[h]*winh1[h][j]+net_h1[j];
 }
  net_h1[j]=net_h1[j]+bh1[j];
  oh1[j]=1.0/(1.0+exp(-net_h1[j]));
}
 hi3a=hi3;
 if(hi2==0)
 {
   for(j=0;j<=hi1-1;j++)
```

```
 {
  oh3[j]=oh1[j];
  hi3a=hi1;
  }
 }
else
{
 for(j=0;j<=hi2-1;j++)
 {
  net_h2[j]=0.0;
  for(h=0;h<=hi1-1;h++)
 {
    net_h2[j]=oh1[h]*wh1h2[h][j]+net_h2[j];
 }
  net_h2[j]=net_h2[j]+bh2[j];
  oh2[j]=1.0/(1.0+exp(-net_h2[j]));
}
 if(hi3==0)
{
  for(j=0;j<=hi2-1;j++)
  {
   oh3[j]=oh2[j];
   hi3a=hi2;
  }
 }
else
{
 for(j=0;j<=hi3-1;j++)
 {
  net_h3[j]=0.0;
  for(h=0;h<=hi2-1;h++)
 {
    net_h3[j]=oh2[h]*wh2h3[h][j]+net_h3[j];
 }
  net_h3[j]=net_h3[j]+bh3[j];
  oh3[j]=1.0/(1.0+exp(-net_h3[j]));
 }
}
}
 for(j=0;j<=out-1;j++)
 {
  net_out[j]=0.0;
  for(h=0;h<=hi3a-1;h++)
 {
   net_out[j]=oh3[h]*wh3o[h][j]+net_out[j];
 }
  net_out[j]=net_out[j]+bo[j];
  outapp[j]=1.0/(1.0+exp(-net_out[j]));
}

free(bh1);
if(hi2!=0){free(bh2);}
if(hi3!=0){free(bh3);}
free(bo);
for(i=0;i<=in-1;i++)free(*(winh1+i));free(winh1);
if(hi2!=0){
for(i=0;i<=hi1-1;i++)free(*(wh1h2+i));free(wh1h2);
}
if(hi3!=0){
for(i=0;i<=hi2-1;i++)free(*(wh2h3+i));free(wh2h3);
}
for(i=0;i<=hi3a-1;i++)free(*(wh3o+i));free(wh3o);
free(net_h1);
if(hi2!=0){free(net_h2);}
if(hi3!=0){free(net_h3);}
free(oh1);
free(oh2);
free(oh3);
free(net_out);
}
```

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

                        Wheel Selection
                        Version 2.0

            Copyright ©1995 AMT Research Laboratory

*************************************************************/

/*************************************************************

        Source file 1:      User Interface
                            Encoding

/*************************************************************/

#include <stdlib.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <stdarg.h>
#include <strstrea.h>

#define Uses_TProgram
#define Uses_TApplication
#define Uses_TKeys
#define Uses_TRect
#define Uses_TMenuBar
#define Uses_TSubMenu
#define Uses_TMenuItem
#define Uses_TStatusLine
#define Uses_TStatusItem
#define Uses_TStatusDef
#define Uses_TDeskTop
#define Uses_TView
#define Uses_TDialog
#define Uses_TButton
#define Uses_TLabel
#define Uses_TInputLine
#define Uses_TEditWindow
#define Uses_MsgBox

#define Uses_TSItem
#define Uses_TCheckBoxes
#define Uses_TRadioButtons

#include <tv.h>

extern void appl(float *inapp,float *outapp,char *weight);
void encode();
void output();

char recom[20];
float roughness, gs, gd;
float inputData[5],outputData[4];

                // assign new command values
const int cmWheel               = 201;
const int cmMyabout             = 203;
const int cmProcess             = 204;
const int cmMore                = 205;

struct WheelData
{

   ushort radioMaterialData;
   ushort checkBoxSeverityData;
  ushort radioHardData;
   ushort radioGrindingData;
   ushort radioWheelTypeData;
   char Roughness[128];

};

WheelData *wheelData;

struct processData
```

```
{
 ushort radioData;
};

processData *ProcessData;

class WTStaticText:public TStaticText{
public:
WTStaticText(const TRect& bounds, const char *aText):
TStaticText(bounds, aText){};
virtual TPalette& getPalette() const;
};

TPalette& WTStaticText::getPalette() const
{
 const char *cpStaticText="\x13";
 static TPalette palette(cpStaticText, sizeof(cpStaticText)-1);
 return palette;
}

class DisplayDialog:public TDialog
{
 public:
 DisplayDialog(const TRect& bounds, const char *aTitle);
 virtual void handleEvent(TEvent& event);
};

class TWheelApp : public TApplication
{

public:
   TWheelApp();
   ~TWheelApp();
    static TStatusLine *initStatusLine( TRect r );
    static TMenuBar *initMenuBar( TRect r );
    virtual void handleEvent( TEvent& event);
    void Wheel();
    void Wheelsurface();
    void about();
    void display();
    void Process();
    void More();
};

TWheelApp::TWheelApp() :
    TProgInit( &TWheelApp::initStatusLine,
               &TWheelApp::initMenuBar,
               &TWheelApp::initDeskTop
             )
{
   wheelData= new WheelData;
   wheelData-> radioMaterialData=0;
   wheelData-> radioHardData=0;
    wheelData->checkBoxSeverityData=0;
    wheelData->radioGrindingData=0;
    wheelData->radioWheelTypeData=0;
    strcpy( wheelData->Roughness ,"0.30");

   ProcessData= new processData;
   ProcessData->radioData=0;
   about();
}

TWheelApp *wa;


TWheelApp::~TWheelApp()
{
}

void TWheelApp::handleEvent(TEvent& event)
{
    TApplication::handleEvent(event);
   if( event.what == evCommand )
          {
          switch( event.message.command )
            {
             case cmMyabout:
             about();
```

```
        break;
        case cmProcess:
                Process();
                break;
        case cmWheel:
                if(ProcessData->radioData==0)
                Wheel();
                if(ProcessData->radioData==1)
                Wheelsurface();
                break;

        default:
                return;
        }
      clearEvent( event );          // clear event after handling
      }
}

void DisplayDialog::DisplayDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&DisplayDialog::initFrame)
{
}

void DisplayDialog::handleEvent(TEvent& event)

{
  TDialog::handleEvent(event);

  if( event.what == evCommand )
  {
        switch( event.message.command )
        {
        case cmMore:
        wa->More();
        break;
        default:
           return;
         }
```

```
                clearEvent( event );          // clear event after handling
                }
  }

/*-----------------------------------------------------------------------------------------------
                                Main Menu

-----------------------------------------------------------------------------------------------*/

TMenuBar *TWheelApp::initMenuBar( TRect r )
{
   char i[]= {126,240,126,0};
   r.b.y = r.a.y + 1;  // set bottom line 1 line below top line
   return new TMenuBar( r,
          *new TSubMenu( i, kbAltA )+
             *new TMenuItem( "~A~bout",cmMyabout,kbAltA,hcNoContext,"Alt-A")+
          *new TSubMenu( "~P~rocess", kbAltP )+
          *new TMenuItem( "~P~rocess", cmProcess,kbAltT,hcNoContext,"Alt-P")+
          *new TSubMenu( "~W~heel", kbAltW )+
           *new TMenuItem("~W~heel", cmWheel, kbAltW,
                          hcNoContext, "Alt-W") +
          *new TSubMenu( "~E~xit", kbAltF )+
            *new TMenuItem( "E~x~it", cmQuit, cmQuit, hcNoContext, "Alt-X" )
          );
}

TStatusLine *TWheelApp::initStatusLine( TRect r )
{
   r.a.y = r.b.y - 1;     // move top to 1 line above bottom
   return new TStatusLine( r,
          *new TStatusDef( 0, 0xFFFF ) +
            *new TStatusItem( "~F1~ Help" , kbF1, cmHelp ) +
            *new TStatusItem( "~F10~ Menu" , kbF10, cmMenu ) +
            *new TStatusItem( "~Alt-X~ Exit", kbAltX, cmQuit ) +
             *new TStatusItem( "~Alt-F3~ Close", kbAltF3, cmClose )
          );
}

void TWheelApp::about()
{
```

```
   TDialog *pd = new TDialog( TRect( 18, 5, 62, 17), "About" );
    if (!pd)
     return;

    // add some centered text to the dialog box
    pd->insert(new TStaticText(TRect(3, 1, 41, 8),
                               "\n\003Intelligent Wheel Selection"
                               "\n\n\n\003Version 2.0"
                               "\n\n\n\n\003Copyright(c) 1994 By AMT Research Lab") );
     pd->insert(new TButton(TRect(15, 9, 27, 11),
                            " OK", cmOK, bfDefault) );

   deskTop->execView(pd);
   destroy(pd);
}

/*-------------------------------------------------------------------------------
                              Process Input

-------------------------------------------------------------------------------*/

void TWheelApp::Process()
{

 TDialog *pd = new TDialog( TRect(24, 5, 58, 17), "Process Types" );
  if( pd )
  {
  TView *b = new TRadioButtons( TRect( 9, 3, 24, 6),
             new TSItem( "~E~xternal",
             new TSItem( "~S~urface",
             new TSItem( "~I~nternal",0 )
             )));
          pd->insert( b );
          pd->insert( new TLabel( TRect( 8, 2, 21, 3), "Process Types", b ));
           pd->insert( new TButton( TRect( 5, 8, 15, 10 ), "~O~K", cmOK,
                       bfDefault ));
          pd->insert( new TButton( TRect( 20, 8, 30, 10 ), "~C~ancel", cmCancel,
    bfNormal ));
    pd->setData(ProcessData);
     int control=deskTop->execView( pd );
          if( control != cmCancel )
          pd->getData(ProcessData);
    destroy(pd);
    }

}


/*-------------------------------------------------------------------------------
                              System Input

-------------------------------------------------------------------------------*/

void TWheelApp::Wheel()
{
 int situation=0;
 TDialog *pd = new TDialog( TRect(5, 1, 75, 21), "System Input" );
  if( pd )
  {
 TView *b = new TRadioButtons( TRect( 3, 3, 36, 10),
            new TSItem( "~G~eneral Steels",
            //new TSItem( "High ~S~peed Steels",
            new TSItem( "~T~ool Steels",
            new TSItem( "~H~igh Alloy Steels",
             new TSItem( "~M~artensitic Stainless Steels",
             new TSItem( "~A~ustenitic Stainless Steels",
            new TSItem( "~C~ast Iron",
            new TSItem( "~N~on-ferrous Metals",0 )
            )))));
          pd->insert( b );
          pd->insert( new TLabel( TRect( 2, 2, 22, 3), "Workpiece Materials ", b ));

          b = new TCheckBoxes( TRect( 3, 12, 36, 18),
             new TSItem( "Rough cast or forged",
             new TSItem( "Interrupted cut",
             new TSItem( "Wide wheel,light pressure",
             new TSItem( "Narrow wheel,high pressure",
             new TSItem( "Large diameter workpiece",
             new TSItem( "Small diameter workpiece",0)
```

```
   ))))));
pd->insert( b );
pd->insert( new TLabel( TRect( 2, 11, 36, 12), "Severity of Operation ",
 b ));

b = new TRadioButtons( TRect( 39, 3, 62, 6),
   new TSItem( "<50 Rc",
   new TSItem( "50-58 Rc",
   new TSItem( ">58 Rc", 0 )
   )));
pd->insert( b );

pd->insert( new TLabel( TRect( 38, 2, 57, 3), "Workpiece Hardness ", b ));

b = new TRadioButtons( TRect( 39, 8, 62, 10),
   new TSItem( "Traverse grinding",
   new TSItem( "Plunge grinding", 0 )
   ));
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 7, 57, 8), "Grinding Methods ", b ));

        b = new TRadioButtons( TRect( 39, 12, 62, 14),
           new TSItem( "Conventional",
           new TSItem( "CBN", 0 )
           ));
        pd->insert( b );
        pd->insert( new TLabel( TRect( 38, 11, 62, 12), "Grinding Wheel Types", b ));

b = new TInputLine( TRect( 58, 15, 65, 16 ), 128 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 15, 58, 16 ),
                   "Workpiece Roughness", b ));
pd->insert( new TLabel( TRect( 65, 15, 68, 16 ),
                   "Ra", b ));

 pd->insert( new TButton( TRect( 40, 17, 50, 19 ), "~O~K", cmOK,
                      bfDefault ));
 pd->insert( new TButton( TRect( 53, 17, 63, 19 ), "~C~ancel", cmCancel,
bfNormal ));
pd->setData(wheelData);
   int control=deskTop->execView( pd );
          if( control != cmCancel )
   {
    pd->getData(wheelData);
     roughness=atof(wheelData->Roughness);
     situation=1;
    }
    }
   destroy(pd);
   if( situation != 0 )
   {
     encode();
       appl(inputData,outputData,"wn10.w");
     display();
   }
}

/*-------------------------------------------------------------------------------------------
                                    Input Encode
-------------------------------------------------------------------------------------------*/

 void encode()
 {
  float a1=0.0;float a2=0.0;float a3=0.0; float a4=0.0; float a6=0.0;
  float b1=0.0;float b2=0.0; float b5=0.0; float b6=0.0;
  float ax=0.0;float ai=0.0;float bx=0.0;float bi=0.0;
 if(wheelData->radioMaterialData==0||
 wheelData->radioMaterialData==1)
 inputData[4]=0.0;
 if(wheelData->radioMaterialData>=4)
 inputData[4]=1.0;

 if(wheelData->radioHardData==0)
 inputData[0]=0.3;
 if(wheelData->radioHardData==1)
 inputData[0]=0.5;
 if(wheelData->radioHardData==2)
 inputData[0]=0.7;
```

```
if(wheelData->radioMaterialData!=0&&
wheelData->radioMaterialData!=1&&
wheelData->radioMaterialData!=5)
inputData[0]=0.7;

if(roughness>=0.9)
inputData[1]=0.3;
else if(roughness<0.9&&roughness>=0.7)
inputData[1]=0.4;
else if(roughness<0.7&&roughness>=0.4)
inputData[1]=0.5;
else if(roughness<0.4&&roughness>=0.2)
inputData[1]=0.6;
else if(roughness<0.2)
inputData[1]=0.7;

inputData[2]=0.1;inputData[3]=0.1;
if(ProcessData->radioData==0){
if(wheelData->checkBoxSeverityData&1){a1=1;b1=-1;}
if(wheelData->checkBoxSeverityData&2){a2=2;b2=1;}
if(wheelData->checkBoxSeverityData&4){a3=-1;}
if(wheelData->checkBoxSeverityData&8){a4=1;}
if(wheelData->checkBoxSeverityData&16){a6=-1;b6=-1;}
if(wheelData->checkBoxSeverityData&32){a6=1;b6=1;}
}
if(ProcessData->radioData==1){
if(wheelData->checkBoxSeverityData&1){a1=1;b1=-1;}
if(wheelData->checkBoxSeverityData&2){a3=-1;}
if(wheelData->checkBoxSeverityData&4){a4=1;}
if(wheelData->checkBoxSeverityData&8){a6=-1;b6=-1;}
if(wheelData->checkBoxSeverityData&16){a6=1;b6=1;}
a6=0;b6=0;
}

if(wheelData->radioGrindingData==1)
b5=1;

ax=max(a1,a2);ax=max(ax,a3);ax=max(ax,a4);ax=max(ax,a6);
ai=min(a1,a2);ai=min(ai,a3);ai=min(ai,a4);ai=min(ai,a6);
bx=max(b1,b2);bx=max(bx,b5);bx=max(bx,b6);
bi=min(b1,b2);bi=min(bi,b5);bi=min(bi,b6);
inputData[2]=(ax+ai)/10.0+0.1;inputData[3]=(bx+bi)/10.0+0.1;
}

/*-----------------------------------------------------------------------------------------------
                                    Output Encode
-----------------------------------------------------------------------------------------------*/

void output()
{
 char wheeltype[4],gritsize[4],grade[2],bond[2];
 if(wheelData->radioMaterialData==2||wheelData->radioMaterialData==3)
  outputData[0]=0.4;

  if(wheelData->radioWheelTypeData==0){
  if(ProcessData->radioData==1)
  strcpy( wheeltype,"MA");
  else {if(outputData[0]<0.26)
   strcpy( wheeltype,"11A");
  else if(outputData[0]<0.5)
   strcpy( wheeltype,"48A");
  else if(outputData[0]<0.65)
  strcpy( wheeltype,"51A");}
  if(outputData[0]>=0.65&&outputData[0]<0.74)
  strcpy( wheeltype,"WA");
  if(outputData[0]>0.75)
  strcpy( wheeltype ,"C");
  }
   else strcpy( wheeltype ,"B");

  if(wheelData->radioWheelTypeData==1)
  outputData[1]=outputData[1]+0.2;

  if(outputData[1]<1.15 && outputData[1]>0.05)
  {
  if(outputData[1]<0.15 )
  strcpy( gritsize,"24");
  else if(outputData[1]<0.25 )
  strcpy( gritsize,"30");
```

```
 else if(outputData[1]<0.35 )
  strcpy( gritsize,"36");
 else if(outputData[1]<0.45 )
  strcpy( gritsize,"46");
 else if(outputData[1]<0.55 )
 strcpy( gritsize,"60");
 else if(outputData[1]<0.65 )
 strcpy( gritsize,"80");
 else if(outputData[1]<0.75 )
  strcpy( gritsize,"100");
 else if(outputData[1]<0.85 )
  strcpy( gritsize,"120");
else
  strcpy( gritsize,"150");
}
if(wheelData->radioWheelTypeData==1)
outputData[1]=outputData[1]-0.2;

if(wheelData->radioWheelTypeData==0){
 if(ProcessData->radioData==1)
 outputData[2]=outputData[2]+0.2;//for surface grinding;
 if(outputData[2]<0.15)
  strcpy( grade,"P");
 else if(outputData[2]>0.15 && outputData[2]<0.25)
  strcpy( grade,"O");
 else if(outputData[2]>0.25 && outputData[2]<0.35)
  strcpy( grade,"N");
 else if(outputData[2]>0.35 && outputData[2]<0.45)
  strcpy( grade,"M");
 else if(outputData[2]>0.45 && outputData[2]<0.55)
  strcpy( grade,"L");
 else if(outputData[2]>0.55 && outputData[2]<0.65)
  strcpy( grade,"K");
 else if(outputData[2]>0.65 && outputData[2]<0.75)
  strcpy( grade,"J");
 else if(outputData[2]>0.75 && outputData[2]<0.85)
  strcpy( grade,"I");
 else
  strcpy( grade,"H");
  if(ProcessData->radioData==1)
 outputData[2]=outputData[2]-0.2;

 strcpy(recom ,"\003");
 strcat(recom,wheeltype);
 strcat(recom,gritsize);
 strcat(recom,grade);
 strcat(recom,"V");
 }

 else {
 if(outputData[2]<0.35 )
  strcpy( grade,"X");
 else if(outputData[2]<0.45)
  strcpy( grade,"V");
 else if(outputData[2]>0.45 && outputData[2]<0.55)
  strcpy( grade,"U");
 else if(outputData[2]>0.55 && outputData[2]<0.65)
  strcpy( grade,"T");
 else if(outputData[2]>0.65 && outputData[2]<0.75)
  strcpy( grade,"S");
 else if(outputData[2]>0.75 && outputData[2]<0.85)
  strcpy( grade,"R");
 else
  strcpy( grade,"Q");

  strcpy(recom ,"\003");
  strcat(recom,wheeltype);
  strcat(recom,gritsize);
  strcat(recom,grade);
  strcat(recom,"100B");
 }
}

/*-------------------------------------------------------------------------------
                              Output Display

-------------------------------------------------------------------------------*/

void TWheelApp::display()
```

```
{
  DisplayDialog *pd = new DisplayDialog( TRect( 18, 3, 62, 19), "OUTPUT" );
  if (!pd)
   return;

  // add some centered text to the dialog box
  pd->insert(new TStaticText(TRect(3, 1, 41, 3),
                        "\n\003The Recommendation is"
                        "\n\n\n\003"
                       ));
  output();
   pd->insert(new WTStaticText(TRect(15, 4, 29, 5),
                       recom
                       ));

  pd->insert(new TStaticText(TRect(3, 5, 41, 7),
                        "\n\003Alternatives are"
                        "\n\n\n\003"
                       ));
 gd=outputData[2];
 outputData[2]=gd-0.1;
output();
 pd->insert(new WTStaticText(TRect(15, 8, 29, 9),
                       recom
                       ));

outputData[2]=gd+0.1;
output();
 pd->insert(new WTStaticText(TRect(15, 9, 29, 10),
                       recom
                       ));

outputData[2]=gd;

gs=outputData[1];
 outputData[1]=gs+0.1;
output();
outputData[1]=gs;
 pd->insert(new WTStaticText(TRect(15, 10, 29, 11),
                       recom
                                  ));

 outputData[1]=gs-0.1;
output();

 pd->insert(new WTStaticText(TRect(15, 11, 29, 12),
                       recom
                       ));

pd->insert(new TButton(TRect(10, 13, 22, 15),
                       "More",cmMore, bfDefault) );

pd->insert(new TButton(TRect(24, 13, 36, 15),
                       "OK", cmOK, bfDefault) );

  deskTop->execView(pd);
  destroy(pd);
}
void TWheelApp::More()
{
 TDialog *pd = new TDialog( TRect( 8, 4, 70, 17), "More Suggestions" );
   if (!pd)
    return;
   pd->insert(new TStaticText(TRect(2, 1, 60, 6),
         "\nThe abrasive type can be changed to a basicl"
         " similar type. For example, the type 48A can be"
         " replaced by 51A or 51A can be replaced by 48A."
         " The 11A can be replaced by the 48A. However, Type"
         " WA for grinding hardened steel and Type"
         ));
   pd->insert(new TStaticText(TRect(2, 6, 60, 9),
         "C for grinding low tensile strength materials such as"
         " cast iron and non-ferrous materials are not recommended"
         " replacements for ensuring the grinding performance."
         ));
    pd->insert(new TButton(TRect(25, 10, 35, 12),
                       " OK", cmOK, bfDefault) );
  deskTop->execView(pd);
  destroy(pd);
```

```
}

/*-------------------------------------------------------------------------------
                        Surface Grinding
-------------------------------------------------------------------------------*/
void TWheelApp::Wheelsurface()
{
 int situation=0;
 TDialog *pd = new TDialog( TRect(3, 1, 77, 21), "System Input" );
  if( pd )
  {
   TView *b = new TRadioButtons( TRect( 3, 3, 36, 10),
              new TSItem( "~G~eneral Steels",
              //new TSItem( "High ~S~peed Steels",
              new TSItem( "~T~ool Steels",
              new TSItem( "~H~igh Alloy Steels",
              new TSItem( "~M~artensitic Stainless Steels",
              new TSItem( "~A~ustenitic Stainless Steels",
              new TSItem( "~C~ast Iron",
              new TSItem( "~N~on-ferrous Metals",0 )
             )))))));
pd->insert( b );
pd->insert( new TLabel( TRect( 2, 2, 21, 3), "Workpiece Materials ", b ));

b = new TCheckBoxes( TRect( 3, 12, 36, 17),
    new TSItem( "Rough cast or forged",
    new TSItem( "Wide wheel,light pressure",
    new TSItem( "Narrow wheel,high pressure",
    new TSItem( "Large surface area",
    new TSItem( "Small surface area",0)
   )))));
pd->insert( b );
pd->insert( new TLabel( TRect( 2, 11, 36, 12), "Severity of Operation ",
 b ));

b = new TRadioButtons( TRect( 39, 3, 62, 6),
    new TSItem( "<50 Rc",
    new TSItem( "50-58 Rc",
    new TSItem( ">58 Rc", 0 )
   )));
pd->insert( b );

b = new TRadioButtons( TRect( 39, 8, 65, 10),
    new TSItem( "Horizontal Spindle",
    new TSItem( "Vertical Spindle", 0 )
   ));
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 7, 57, 8), "Grinding Methods ", b ));

b = new TRadioButtons( TRect( 39, 12, 62, 14),
    new TSItem( "Conventional",
    new TSItem( "CBN", 0 )
   ));
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 11, 62, 12), "Grinding Wheel Types", b ));

 b = new TInputLine( TRect( 57, 15, 64, 16 ), 128 );
pd->insert( b );
  pd->insert( new TLabel( TRect( 38, 15, 57, 16 ),
              "Workpiece Roughness", b ));
  pd->insert( new TLabel( TRect( 64, 15, 67, 16 ),
              "Ra", b ));

 pd->insert( new TButton( TRect( 40, 17, 50, 19 ), "~O~K", cmOK,
              bfDefault ));
 pd->insert( new TButton( TRect( 53, 17, 63, 19 ), "~C~ancel", cmCancel,
bfNormal ));

pd->setData(wheelData);

int control=deskTop->execView( pd );
        if( control != cmCancel )
{
 pd->getData(wheelData);
 //diameter=atof(wheelData->WorkDia );
  roughness=atof(wheelData->Roughness);
 situation=1;
}
}
```

```
    destroy(pd);
    if( situation != 0 )
    {
     encode();
       appl(inputData,outputData,"wn10.w"); // call neural network
     display();
    }
}

/*-------------------------------------------------------------------------------
                              Main Programme

-------------------------------------------------------------------------------*/
int main()
{

  TWheelApp myApp;
  myApp.run();

  return 0;
}

/*************************************************************

          Source file 2:       Neural Network Application Module

*************************************************************/

// See Feedforward Neural Network Source file 3.
```

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

                Selection of Grinding Conditions
                        Version 1.0

        Copyright ©1995 AMT Research Laboratory

*************************************************************/

/*************************************************************

        Source file 1:      Headfile for Creating Classes

*************************************************************/

#include <tv.h>

const int cmMyFileOpen    = 200;  // assign new command values
const int cmMyNewWin      = 201;
const int cmNewDialog     = 202;
const int cmMyabout       = 203;
const int cmProcess       = 204;
const int cmWorkpiece     = 205;
const int cmProType       = 206;
const int cmWheelS        = 207;
const int cmCoolant       = 219;
const int cmDress         = 220;
const int cmGrinding      = 208;
const int cmCase          = 209;
const int cmDelete        = 210;
const int cmAdd           = 211;
const int cmEdit          = 212;
const int cmItemSelect    = 213;
const int cmNext1         = 214;
const int cmPre           = 215;
const int cmDeletec       = 216;
const int cmAddc          = 217;
const int cmEditc         = 218;
const int cmDelete1       = 219;
const int cmAdd1          = 220;
const int cmEdit1         = 221;
const int cmSaveMat       = 222;
const int cmSaveCase      = 223;
const int cmMatchInd      = 224;
const int cmSaveNew       = 225;
const int cmOldCase       = 226;
const int cmMore          = 227;
const int cmChangItem1    = 1001;
const int cmChangItem2    = 1002;

int cc, ccount, bestnum;
char *hard,*wt,*bt;
float Vfn, Vwn, fdn, Den,Sr;

extern void appl(float *inapp, float *outapp, char *weight);
void encode(); char *getWheel();
float roughness; void recValueCase(int); void recValueRule(int);
  FILE *Mbase;
    float inputData[5], outputData[4];

struct WheelData
{

    ushort radioMaterialData;
    ushort checkBoxSeverityData;
   ushort radioHardData;
    ushort radioGrindingData;
    ushort radioWheelTypeData;
    char Roughness[128];
};
  WheelData *wheelData;

struct WTS
{
 unsigned wts;
};
WTS *Wts;

 struct IndexData
```

```
{
 TSortedCollection *data;
 ushort select;
};
IndexData * IndexData1;

struct CaseData
{
   char Index[30];
   char Number[20];
   char Process[20];
   char Machine[30];
   char Power[10];
   char materialtype[30];
   char material[20];
   char Hardness[10];
   char Roughness[10];
   char Tolerance[10];
   char Roundness[10];
   char Dw1[10];
   char Dw2[10];
   char Width[10];
   char Wheel[20];
   char Vs[10];
   char Ds[10];
   char Dresser[30];
   char Coolant[30];
   char Vw[10];
   char Vf[10];
   char ad[10];
   char fd[10];
   char SOT[10];
   char ED[10];
   char SR[10];
   char ec[10];
   char G[10];
};
CaseData *CaseData1, *CaseData2, *CaseData3, *recvalue;

struct WorkPieceData
{
   char MaterialTypeLine[30];
   char MaterialLine[30];
   ushort SurConCheckData;
   ushort HardnessRadioData;
   ushort WheelRadioData;
   char DiameterLine[30];
   char Dw2[30];
   char Width[30];
   char RoughnessLine[30];
   char ToleranceLine[30];
   char Roundness[30];
   char Partnumber[30];
};
WorkPieceData *WorkPieceData1;

struct adddata
{
  char MaterialLine[30];
};
adddata *addData;

struct ProcessData
{
ushort radioButtonData1;
ushort radioButtonData2;
char PowerLine[30];
};
ProcessData *ProData;

struct MATBase
{
  char index[40];
  char material[40];
  char thermal[40];
  char specific[40];
  char melting[40];
  char critical[40];
  char density[40];
};
```

```
MATBase *MatBase[30],*MatBase1;

struct INDEX
{
  char   vipi[10];
  char   mater[10];
  char   wheeli[10];
  float  roughi;
  float  diaWorki;
  float  diaWheeli;
  float  Vsi;
};
INDEX *Index;

struct Materials
{
TCollection *data;
ushort select;
};
Materials *Materials1;

struct MM
{
 TCollection *Data;
 ushort select;
};
MM *List;
MM *Materials_data[30];

struct UserWheel
{
 ushort wheeltype;
 ushort bondtype;
 char grits[10];
 char grade[10];
 char diameter[10];
 char speed[10];
};
UserWheel *userwheel;
```

```
class TMaterials:public TObject,public TStreamable
{
public:

TMaterials(int );
static const char *const name;
static TStreamable *build();
virtual void write(opstream& os);
virtual void *read(ipstream& is);
private:
virtual const char *streamableName() const
{ return name;}

protected:
TMaterials(StreamableInit){}
};

TMaterials::TMaterials(int item){
cc=item;
}

const char *const TMaterials:: name="TMaterials";
TStreamable *TMaterials::build()
{
 return new TMaterials(streamableInit);
}
TStreamableClass RMaterials(TMaterials::name,
                            TMaterials::build,
                            __DELTA(TMaterials)
                            );

inline ipstream& operator >> ( ipstream& is, TMaterials& cl )
   { return is >> (TStreamable&)cl; }
inline ipstream& operator >> ( ipstream& is, TMaterials*& cl )
   { return is >> (void *&)cl; }

inline opstream& operator << ( opstream& os, TMaterials& cl )
   { return os << (TStreamable&)cl; }
inline opstream& operator << ( opstream& os, TMaterials* cl )
```

```
  { return os << (TStreamable *)cl; }

  class TwpieceCollection:public TCollection
{
public:
TwpieceCollection(ccIndex aLimit, ccIndex aDelta):
TCollection(aLimit,aDelta){};
static const char *const name;
static TStreamable *build();
private:
virtual const char *streamableName() const
{ return name;}
void *readItem(ipstream&is);
void writeItem(void *obj, opstream&os);
protected:
TwpieceCollection(StreamableInit):
TCollection(streamableInit){}
};
void TwpieceCollection::writeItem(void *obj, opstream&os)
{os.writeString((const char *)obj);
}
void *TwpieceCollection::readItem(ipstream&is)
{
 return is.readString();
}
const char *const TwpieceCollection:: name="TwpieceCollection";
TStreamable *TwpieceCollection::build()
{
 return new TwpieceCollection(streamableInit);
}
TStreamableClass RwpieceCollection(TwpieceCollection::name,
                                   TwpieceCollection::build,
                                   __DELTA(TwpieceCollection)
                                  );


class TCaseCollection:public TStringCollection
{
public:
TCaseCollection(ccIndex aLimit, ccIndex aDelta):
TStringCollection(aLimit,aDelta){};
virtual ccIndex insert(CaseData *item);
virtual ccIndex indexOf(void *item);
virtual void *keyOf1(void *item);
virtual void *keyOf(void *item);
static const char *const name;
static TStreamable *build();
private:
virtual const char *streamableName() const
{ return name;}
void *readItem(ipstream&is);
void writeItem(void *obj, opstream&os);
protected:
TCaseCollection(StreamableInit):
TStringCollection(streamableInit){}
};

const char *const TCaseCollection:: name="TCaseCollection";
TStreamable *TCaseCollection::build()
{
 return new TCaseCollection(streamableInit);
}
TStreamableClass RCaseCollection(TCaseCollection::name,
                                 TCaseCollection::build,
                                 __DELTA(TCaseCollection)
                                );
inline ipstream& operator >> ( ipstream& is, TCaseCollection& cl )
  { return is >> (TStreamable&)cl; }
inline ipstream& operator >> ( ipstream& is, TCaseCollection*& cl )
  { return is >> (void *&)cl; }

inline opstream& operator << ( opstream& os, TCaseCollection& cl )
  { return os << (TStreamable&)cl; }
inline opstream& operator << ( opstream& os, TCaseCollection* cl )
  { return os << (TStreamable *)cl; }

TCaseCollection *CaseCollect;
TMaterials *Materials2;
  TListBox* Box, *Box1;
TSortedListBox *Box2;
```

```
TDialog* pd,*pd2,*pdr;
TScrollBar *sr,*sr1, *sr2;
TInputLine *d;
FILE *fp;
TStringCollection *TIndexCollection;

void TMaterials::write(opstream& os)
{
for (int i=0; i<= cc; i++)
os<<Materials_data[i]->Data;
}

void *TMaterials::read(ipstream& is)
{
 for (int i=0; i<= cc; i++)
 is>>Materials_data[i]->Data;
 return this;
}

class TTLabel:public TLabel{
public:
TTLabel(const TRect& bounds, const char *aText, TView *aLink):
TLabel(bounds, aText, aLink){};
virtual void handleEvent(TEvent& event);
};

class TTInputLine:public TInputLine{
public:
TTInputLine(const TRect& bounds, int aMaxLen):
TInputLine(bounds, aMaxLen){};
virtual void handleEvent(TEvent& event);
virtual TPalette& getPalette() const;
};


TPalette& TTInputLine::getPalette() const
{
  const char *cpInputLine="\x1A\x1A\x1B\x1C";
  static TPalette palette(cpInputLine, sizeof(cpInputLine)-1);
  return palette;
}

class Casedisplay:public TInputLine{
public:
Casedisplay(const TRect& bounds, int aMaxLen):
TInputLine(bounds, aMaxLen){};
virtual TPalette& getPalette() const;
};
TTInputLine *d1;

TPalette& Casedisplay::getPalette() const
{
  const char *cpInputLine="\x1A\x1A\x1B\x1C";
  static TPalette palette(cpInputLine, sizeof(cpInputLine)-1);
  return palette;
}

class WTStaticText:public TStaticText{
public:
WTStaticText(const TRect& bounds, const char *aText):
TStaticText(bounds, aText){};
virtual TPalette& getPalette() const;
};

TPalette& WTStaticText::getPalette() const
{
  const char *cpStaticText="\x13";
  static TPalette palette(cpStaticText, sizeof(cpStaticText)-1);
  return palette;
}

class TTDialog:public TDialog
{
 public:
 TTDialog(const TRect& bounds, const char *aTitle);
 virtual void handleEvent(TEvent& event);
};
void TTDialog::TTDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&TTDialog::initFrame)
{
```

```
}
class CaseDialog:public TDialog
{
  public:
  CaseDialog(const TRect& bounds, const char *aTitle);
  virtual void handleEvent(TEvent& event);
  void getData(void*);
  void setData(void*);
};
void CaseDialog::CaseDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&TTDialog::initFrame)
{

}
void CaseDialog::getData(void *rec)
{
   ushort i = 0;
   ushort j =0;
   if (last != 0 )
    {
         TView* v = last;
    do {
       v->getData( ((char *)rec) + i );
       i += v->dataSize();
            v = v->prev();
            j=j+1;
            } while( j <= 56);
          }
}

void CaseDialog::setData(void *rec)
{
   ushort i = 0;
   ushort j =0;
   if( last!= 0 )
          {
          TView* v = last;
          do {
             v->setData( (char *)rec + i );
             i += v->dataSize();
             v = v->prev();
             j=j+1;
             } while (j<= 56);
          }
}

class TTListBox: public TListBox{
public:
TTListBox(const TRect& bounds, ushort aNumCols, TScrollBar *aScrollBar):
TListBox(bounds, aNumCols, aScrollBar){};
void TTListBox::focusItem(short item);
};

class TListBox1: public TSortedListBox{
public:
TListBox1(const TRect& bounds, ushort aNumCols, TScrollBar *aScrollBar):
TSortedListBox(bounds, aNumCols, aScrollBar){};
void TListBox1::focusItem(short item);
};

class TTListBox1: public TListBox{
public:
TTListBox1(const TRect& bounds, ushort aNumCols, TScrollBar *aScrollBar):
TListBox(bounds, aNumCols, aScrollBar){};
virtual void handleEvent(TEvent& event);
};

class DisplayDialog:public TDialog
{
  public:
  DisplayDialog(const TRect& bounds, const char *aTitle);
  virtual void handleEvent(TEvent& event);
};

void TCaseCollection::writeItem(void *obj, opstream&os)
{os.writeBytes(obj,sizeof(struct CaseData));
}
void *TCaseCollection::readItem(ipstream&is)
{
```

```
 void *obj;
 obj=new char[sizeof(struct CaseData)];
 is.readBytes(obj, sizeof(struct CaseData));
 return obj;
 }

 ccIndex TCaseCollection::insert( CaseData *item )
{
   ccIndex  i;
   if( search( keyOf(item), i ) == 0 || duplicates )
          atInsert( i, item );
   return i;
}

ccIndex TCaseCollection::indexOf(void *item)
{
   ccIndex  i;

   if( search( keyOf1(item), i ) == 0 )
      return ccNotFound;
   else
     {
     if( duplicates )
       {
            while( i < count && item != items[i])
          i++;
       }
     if( i < count )
       return i;
          else
       return ccNotFound;
     }
}

void *TCaseCollection::keyOf(void *item)
{
 return(void*)((CaseData*)item)->Index;
 }

void *TCaseCollection::keyOf1(void *item)
{
 return item;
 }
```

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

                Selection of Grinding Conditions
                        Version 1.0

            Copyright ©1995 AMT Research Laboratory

*********************************************************/

/*********************************************************

        Source file 2:      Intelligent Agents

*********************************************************/

#include <stdlib.h>
#include <iostream.h>
#include <strstream.h>
#include <fstream.h>
#include <stdio.h>
#include <string.h>
#include <mem.h>
#include <conio.h>
#include <math.h>

#define Uses_opstream
#define Uses_ipstream
#define Uses_TEventQueue
#define Uses_TEvent
#define Uses_TProgram
#define Uses_TApplication
#define Uses_TKeys
#define Uses_TRect
#define Uses_TMenuBar
#define Uses_TSubMenu
#define Uses_TMenuItem
#define Uses_TStatusLine
#define Uses_TStatusItem
#define Uses_TStatusDef
#define Uses_TDeskTop
#define Uses_TView
#define Uses_TWindow
#define Uses_TFrame
#define Uses_TScrollBar
#define Uses_TDialog
#define Uses_TButton
#define Uses_TSItem
#define Uses_TCheckBoxes
#define Uses_TRadioButtons
#define Uses_TLabel
#define Uses_TInputLine
#define Uses_TListBox
#define Uses_TSortedListBox
#define Uses_TCollection
#define Uses_TNSCollection
#define Uses_TStringCollection
#define Uses_ofpstream
#define Uses_ifpstream
#define Uses_TStreamableClass
#define Uses_iopstream

#include "intsel.h"
extern int password();

__link(RStringCollection);

class TMyApp : public TApplication
{

public:
   TMyApp();
   ~TMyApp();
    static TStatusLine *initStatusLine( TRect r );
    static TMenuBar *initMenuBar( TRect r );
    virtual void handleEvent( TEvent& event);
    void Process();
    void Workpiece();
    void about();
    void Parameter();
```

```
    void newListBox();
    void add(int);
    void add1(int);
    void Case();
    void match();
     void Recommendation(int);
    void oldcase();
    void WheelDisplay();
   void WheelT();
   void More();

};

TMyApp::TMyApp() :
    TProgInit( &TMyApp::initStatusLine,
               &TMyApp::initMenuBar,
               &TMyApp::initDeskTop
             )
{
about();

//initial workpiece data
WorkPieceData1 = new WorkPieceData;
strcpy( WorkPieceData1 ->MaterialTypeLine, "" );
strcpy( WorkPieceData1 ->MaterialLine, "" );
WorkPieceData1->SurConCheckData = 0;
WorkPieceData1->HardnessRadioData = 0;
WorkPieceData1->WheelRadioData = 0;
strcpy( WorkPieceData1->DiameterLine, "30" );
strcpy( WorkPieceData1->Dw2, "" );
strcpy( WorkPieceData1->Width, "" );
strcpy( WorkPieceData1->RoughnessLine, "0.4" );
strcpy( WorkPieceData1->ToleranceLine, "" );
strcpy( WorkPieceData1->Roundness, "" );
strcpy( WorkPieceData1->Partnumber, "" );

//initial process data
ProData=new ProcessData;
ProData->radioButtonData1=0;
ProData->radioButtonData2=0;
strcpy( ProData->PowerLine, "5" );

//initial materials add function data
addData=new adddata;
strcpy( addData->MaterialLine, "0" );

CaseData1=new CaseData;
CaseData2=new CaseData;
CaseData3=new CaseData;
recvalue=new CaseData;

wheelData=new WheelData;

Wts=new WTS;
Wts->wts=0;

//initial user selection of wheel data
userwheel= new UserWheel;
userwheel->wheeltype=0;
userwheel->bondtype=0;
strcpy( userwheel->grits, "60" );
strcpy( userwheel->grade, "M" );
strcpy( userwheel->diameter, "360" );
strcpy( userwheel->speed, "33" );

//arrange memory for material database
Materials1=new Materials;
Materials1->select=0;
Materials2=new TMaterials(29);

Index= new INDEX;

  CaseCollect=new TCaseCollection(10,10);

IndexData1=new IndexData;
IndexData1->data=new TStringCollection(20,10);
IndexData1->select=0;
CaseCollect->duplicates=True;
IndexData1->data->duplicates=True;
```

```
    //built case base reading data from files
     ifpstream iCC("CaseCol.txt");
    iCC>>CaseCollect;
    iCC.close();
     ifpstream iCI("IndexCol.txt");
    iCI>>IndexData1->data;
    iCI.close();

   TCommandSet command;
   command+=cmMatchInd;
   command+=cmWheelS;
   disableCommands(command);

}
TMyApp *app;
TMyApp::~TMyApp()
{

  delete WorkPieceData1;
  delete ProData;
  //delete MatBase;
  delete CaseData1;
  delete CaseData2;
}

void TMyApp::handleEvent(TEvent& event)
{
    TApplication::handleEvent(event);
   if( event.what == evCommand )
          {
          switch( event.message.command )
            {
            case cmProType:
                    break;
            case cmGrinding:
                   Workpiece();
                   break;
            case cmProcess:
                    Process();
                   break;
            case cmWorkpiece:
                   Workpiece();
                   break;
            case cmMyabout:
                    about();
                   break;
            case cmCase:
                   Case();
                   break;
            case cmMatchInd:
                   match();
                   break;
            case cmWheelS:
                   WheelDisplay();
                   break;
            default:
                   return;
            }
          clearEvent( event );          // clear event after handling
          }
}

/*-------------------------------------------------------------------------------------------
                                    Main Menu

-------------------------------------------------------------------------------------------*/

TMenuBar *TMyApp::initMenuBar( TRect r )
{
    char i[]= {126,240,126,0};
   r.b.y = r.a.y + 1;  // set bottom line 1 line below top line
   return new TMenuBar( r,
          *new TSubMenu( i, kbAltA )+
             *new TMenuItem( "~A~bout",cmMyabout,kbAltA,hcNoContext,"Alt-A")+
          *new TSubMenu( "~I~nput", kbAltI )+
             *new TMenuItem( "~P~rocess", cmProcess,kbAltP, hcNoContext,"Alt-P" )+
             *new TMenuItem( "~W~orkpiece", cmWorkpiece,hcNoContext)+
          *new TSubMenu( "~S~elect", kbAltS )+
            *new TMenuItem( "~W~heel Select", cmWheelS, kbAltW, hcNoContext,
          "Alt-W" )+
```

```
            *new TMenuItem( "~C~onditions Select", cmMatchInd, kbAltC,
          hcNoContext,  "Alt-C")+
          *new TSubMenu( "~C~ases", kbAltC )+
             *new TMenuItem( "~C~ase base", cmCase,kbAltB,hcNoContext,"Alt-B")+
          *new TSubMenu( "~E~xit", kbAltF )+
             *new TMenuItem( "E~x~it", cmQuit, cmQuit, hcNoContext, "Alt-X" )
          );
}


TStatusLine *TMyApp::initStatusLine( TRect r )
{
   r.a.y = r.b.y - 1;     // move top to 1 line above bottom
   return new TStatusLine( r,
          *new TStatusDef( 0, 0xFFFF ) +
             *new TStatusItem( "~F1~ Help" , kbF1, cmHelp ) +
             *new TStatusItem( "~F10~ Menu" , kbF10, cmMenu ) +
             *new TStatusItem( "~Alt-X~ Exit", kbAltX, cmQuit ) +
             *new TStatusItem( "~Alt-F3~ Close", kbAltF3, cmClose )
          );
}

void TMyApp::about()
{
   TDialog *pd = new TDialog( TRect( 18, 5, 62, 17), "About" );
    if (!pd)
     return;

    pd->insert(new TStaticText(TRect(3, 1, 41, 8),
                          "\n\003Intelligent System for Selection of Grinding Conditions"
                          "\n\n\n\003Version 1.0"
                          "\n\n\n\n\003Copyright(c) 1995 By AMT Research Lab") );
     pd->insert(new TButton(TRect(15, 9, 27, 11),
                         " OK", cmOK, bfDefault) );

   deskTop->execView(pd);
   destroy(pd);
}
```

```
/*-------------------------------------------------------------------------------------------
                                    Process Input

-------------------------------------------------------------------------------------------*/

void TMyApp::Process()
{
   TDialog *pd = new TDialog( TRect(14, 5, 64, 17 ), "Process" );
   if( pd )
          {
          TView *B = new TRadioButtons( TRect( 3, 3, 19, 7),
             new TSItem( "~E~xternal",
             new TSItem( "~I~nternal",
             new TSItem(      "Ce~n~treless",
             new TSItem( "~S~urface", 0)
             ))));
          pd->insert( B );
          pd->insert( new TLabel( TRect( 2, 2, 19, 3 ),
                    "~P~rocess", B ));

TView *b = new TRadioButtons( TRect( 23, 3, 46, 5),
   new TSItem( "P~l~unge",
   new TSItem( "~T~raverse", 0)
   ));
pd->insert( b );
pd->insert( new TLabel( TRect( 22, 2, 46, 3 ),
          "~M~achining Type ", b ));

b = new TInputLine( TRect( 38, 6, 44, 7 ), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 22, 6, 38, 7 ),
          "Power ~A~vailable ", b ));
pd->insert( new TLabel( TRect( 43, 6, 46, 7 ),
          "kw", b ));

pd->insert( new TButton( TRect( 15, 9, 25, 11 ), "~O~K", cmOK,
             bfDefault ));
pd->insert( new TButton( TRect( 27, 9, 37, 11 ), "~C~ancel", cmCancel,
             bfNormal ));
```

```
            pd->setData(ProData);
            B->select();
            ushort control=execView(pd);
            if(control!=cmCancel)
            pd->getData(ProData);
            }
    destroy(pd);
}

/*-------------------------------------------------------------------------------------------
                              Workpiece Information Input
-------------------------------------------------------------------------------------------*/

void TMyApp::Workpiece()
{   Index->vipi[2]='\0';
    TDialog *pd = new TDialog( TRect(6, 1, 73, 21), "Workpiece" );
   if( pd )
   {
TView *b;
        d = new TTInputLine( TRect( 3, 3, 29, 4 ), 30 );
        d1= new TTInputLine( TRect( 3, 4, 29, 5 ), 30 );
        pd->insert( d );
        pd->insert( d1);
        pd->insert( new TTLabel( TRect( 3, 2, 27, 3), "~M~aterials(F2)", d ));

        b = new TCheckBoxes( TRect( 3, 7, 29, 9),
           new TSItem( "Rough Cast or Forged",
           new TSItem( "Interrupted Cut",0)
           ));
        pd->insert( b );
        pd->insert( new TLabel( TRect( 3, 6, 27, 7), "Sur~f~ace Condition", b ));


        b = new TRadioButtons( TRect( 3, 11, 29, 14),
           new TSItem( "< 50Rc",
           new TSItem( "50-58Rc",
           new TSItem( "> 58Rc",0)
                    )));
        pd->insert( b );
        pd->insert( new TLabel( TRect( 3, 10, 15, 11), "~H~ardness", b ));


        b = new TRadioButtons( TRect( 3, 16, 29, 18),
           new TSItem( "By System",
           new TSItem( "By User", 0 )
           ));
        pd->insert( b );
        pd->insert( new TLabel( TRect( 3, 15, 29, 16), "Whee~l~ Selection", b ));

        b = new TInputLine( TRect(52, 3, 60, 4 ), 30 );
        pd->insert( b );
        pd->insert( new TLabel( TRect(34, 3, 52, 4 ),
                "~S~tart diameter", b ));
        pd->insert( new TLabel( TRect( 60, 3, 63, 4 ),
                "mm", b ));

        b = new TInputLine( TRect(52, 5, 60, 6 ), 30 );
        pd->insert( b );
        pd->insert( new TLabel( TRect(34, 5, 52, 6 ),
                "~F~inish diameter", b ));
        pd->insert( new TLabel( TRect( 60, 5, 63, 6 ),
                "mm", b ));

        b = new TInputLine( TRect( 52, 7, 60, 8 ), 30 );
        pd->insert( b );
        pd->insert( new TLabel( TRect( 34, 7, 52, 8 ),
                "~W~idth ", b ));
        pd->insert( new TLabel( TRect( 60, 7, 63, 8 ),
                "mm", b ));

        b = new TInputLine( TRect( 52, 9, 60, 10 ), 30 );
        pd->insert( b );
        pd->insert( new TLabel( TRect( 34, 9, 52, 10 ),
                "~M~ax roughness Ra", b ));
        pd->insert( new TLabel( TRect( 60, 9, 63, 10 ),
                "um", b ));

        b = new TInputLine( TRect( 52, 11, 60, 12 ), 30 );
        pd->insert( b );
```

```
pd->insert( new TLabel( TRect( 34, 11, 52, 12 ),
        "~S~ize tolerance", b ));
pd->insert( new TLabel( TRect( 60, 11, 63, 12 ),
        "um", b ));

b = new TInputLine( TRect( 52, 13, 60, 14 ), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 34, 13, 52, 14 ),
        "R~o~undness tol", b ));
pd->insert( new TLabel( TRect( 60, 13, 63, 14 ),
        "um", b ));

b = new TInputLine( TRect( 52, 15, 60, 16 ), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 34, 15, 52, 16 ),
        "~P~art number", b ));


pd->insert( new TButton( TRect( 36, 17, 46, 19 ), "~O~K", cmOK,
        bfDefault ));
pd->insert( new TButton( TRect( 49, 17, 59, 19 ), "~C~ancel", cmCancel,
        bfNormal ));

pd->setData( WorkPieceData1 );
d->select();
ushort control = deskTop->execView( pd );

if( control != cmCancel ){
   pd->getData( WorkPieceData1 );
   TCommandSet command;
   command+=cmMatchInd;
   command+=cmWheelS;
   enableCommands(command);
   char *roug;
   if(WorkPieceData1->HardnessRadioData==0)hard=newStr("L");
   if(WorkPieceData1->HardnessRadioData==1)hard=newStr("M");
   if(WorkPieceData1->HardnessRadioData==2)hard=newStr("H");
   strcat(Index->vipi,hard );
   Index->roughi=atof(WorkPieceData1->RoughnessLine);
   if(Index->roughi>=0.9)
                roug=newStr("1");
                 else if(Index->roughi<0.9&&Index->roughi>=0.7)
                roug=newStr("2");
                 else if(Index->roughi<0.7&&Index->roughi>=0.4)
                roug=newStr("3");
                 else if(Index->roughi<0.4&&Index->roughi>=0.20)
                roug=newStr("4");
                 else if(Index->roughi<=0.20)
                roug=newStr("5");
                 strcat(Index->vipi,roug);
                 Index->diaWorki=atof(WorkPieceData1->DiameterLine);
                 strcpy(Index->mater,WorkPieceData1->MaterialLine);
                pd->getData( WorkPieceData1 );
                }
            }
  destroy( pd );
}

/*-------------------------------------------------------------------------------------------------
                        Material Database Management
-------------------------------------------------------------------------------------------------*/

void TTInputLine::handleEvent(TEvent& event)
{
   if( event.what == evKeyDown){
     if(event.keyDown.keyCode==kbF2){
    app->newListBox();
   clearEvent(event);
   return;
   }
   }
    if((event.what==evMouseDown)&&(event.mouse.doubleClick)){
    app->newListBox();

   clearEvent(event);
   return;
  }
 }
```

```
 void TTLabel::handleEvent(TEvent& event)
{
   if( event.what == evKeyDown){
    if(event.keyDown.keyCode==kbF2){
    app->newListBox();
    clearEvent(event);
   return;
   }
   }
    if((event.what==evMouseDown)&&(event.mouse.doubleClick)){
   app->newListBox();
   clearEvent(event);
  return;
 }
   TLabel::handleEvent(event);
}

void TTDialog::handleEvent(TEvent& event)
{   TDialog::handleEvent(event);

  if( event.what == evCommand )
   {
        switch( event.message.command )
        {
        case cmAdd:
        Box->getData(Materials1);
        app->add(0);

        break;
        case cmEdit:
        Box->getData(Materials1);
        app->add(1);
        break;
        case cmDelete:
            Box->getData(Materials1);
            int number1=Materials1->data->getCount()-Box->focused-1;
            Materials1->data->atRemove(Materials1->select);
            Box->setRange(Materials1->data->getCount());
           Box->drawView();
            memmove(&Materials_data[Box->focused],
             &Materials_data[Box->focused+1], number1*sizeof(void*));
           memmove(&MatBase[Box->focused],
            &MatBase[Box->focused+1], number1*sizeof(void*));

           List->Data->freeAll();
            int range1=Materials_data[Box->focused]->Data->getCount();
            for(int i=0;i<=range1-1;i++)
           {
            List->Data->insert(newStr((char*)Materials_data[Box->focused]
           ->Data->at(i)));
          }
           List->select=0;
           Box1->setRange(range1);
          Box1->drawView();
        break;
        case cmAdd1:
        app->add1(0);
        break;
        case cmEdit1:
        app->add1(1);
        break;
        case cmDelete1:
           Box1->getData(List);
           List->Data->atRemove(Box1->focused);
           Materials_data[Box->focused]->Data->atRemove(Box1->focused);
           Box1->setRange(List->Data->getCount());
          Box1->drawView();
        break;
        case cmSaveMat:
          Box->getData(Materials1);
          ofpstream of("Mtype.txt");
          of<<Materials1->data;
          of.close();

          ofpstream of1("Matdata1.txt");
          of1<<Materials2;
          of1.close();
```

```
            if((fp=fopen("MatBase2.txt","wb"))==NULL)
            return;
            for(i=0;i<=29;i++){
            fwrite(MatBase[i], sizeof(struct MATBase),1, fp); }
            fclose(fp);
            break;
           default:
              return;
            }
          clearEvent( event );          // clear event after handling
          }
  }

void TTListBox::focusItem(short item)
{  TListBox::focusItem(item);
  message(owner,evBroadcast,cmChangItem1,0);
 }

void TTListBox1::handleEvent(TEvent& event)
{
  if((event.what==evMouseDown)&&(event.mouse.doubleClick)){
 owner->endModal(cmItemSelect);
 clearEvent(event);
return;
}
 TListBox::handleEvent(event);

 if( event.what == evBroadcast&&
 event.message.command==cmChangItem1){
List->Data->freeAll();
 int range1=Materials_data[Box->focused]->Data->getCount();
 for(int i=0;i<=range1-1;i++)
{
  List->Data->insert(newStr((char*)Materials_data[Box->focused]
 ->Data->at(i)));
 }
  List->select=0;
  Box1->setRange(range1);
 Box1->drawView();
 clearEvent(event);
  }
 }

void TListBox1::focusItem(short item)
{  TListBox::focusItem(item);
  message(owner,evBroadcast,cmChangItem2,0);
}

void TMyApp::newListBox()
{
    for(int i=0;i<=15;i++){
   Materials_data[i]=new MM;
    Materials_data[i]->Data=new TwpieceCollection(30,10);
    Materials_data[i]->select=0;
   }
    ifpstream is("Mtype.txt");
          is>>Materials1->data;
          Materials1->select=0;
          is.close();

 ifpstream is2("Matdata1.txt");
 is2>>Materials2;
 is2.close();

 int bb=Materials_data[0]->Data->getCount();
List=new MM;
 List->Data=new TwpieceCollection(30,10);

 if((fp=fopen("MatBase2.txt","rb"))==NULL)
        return;
MatBase1=new MATBase;
 for( i=0; i<=15;i++){
MatBase[i]=new MATBase;
 fread(MatBase[i], sizeof(struct MATBase),1, fp);}
 fclose(fp);

 for(i=0;i<=bb-1;i++)
 List->Data->insert(newStr((char*)Materials_data[0]->Data->at(i)));
List->select=0;
```

```
pd = new TTDialog( TRect(6, 3, 74, 22), "Workpiece Materials" );
if( pd )
{
TRect r = TRect(28, 3, 29, 12);
sr=new TScrollBar(r);
pd->insert(sr);
Box=new TTListBox( TRect( 5, 3, 28, 12), 1, sr);
Box->setData(Materials1);
pd->insert(Box);
pd->insert( new TLabel( TRect( 3, 2, 20, 3 ),
                    "Material ~G~roup", Box ));

TRect r1 = TRect(60, 3, 61, 12);

sr1=new TScrollBar(r1);
pd->insert(sr1);
Box1=new TTListBox1( TRect( 37, 3, 60, 12), 2, sr1);
 Box1->setData(List);//Materials2->Materials_data[0]);
pd->insert(Box1);
 pd->insert( new TLabel( TRect( 36, 2, 45, 3 ),
                    "~M~aterial", Box1 ));

 pd->insert( new TButton( TRect( 2, 13, 12, 15 ), "~D~elete", cmDelete,
bfNormal ));
 pd->insert( new TButton( TRect( 12, 13, 22, 15), "~A~dd", cmAdd,
bfNormal ));
 pd->insert( new TButton( TRect( 22, 13, 32, 15), "~E~dit", cmEdit,
bfNormal ));

 pd->insert( new TButton( TRect( 34, 13, 44, 15 ), "De~l~ete", cmDelete1,
bfNormal ));
 pd->insert( new TButton( TRect( 44, 13, 54, 15), "Ad~d~", cmAdd1,
bfNormal ));
 pd->insert( new TButton( TRect( 54, 13, 64, 15), "Edi~t~", cmEdit1,
bfNormal ));

 pd->insert( new TButton( TRect( 17, 16, 27, 18),"~C~ancel", cmCancel,
bfNormal ));

 pd->insert( new TButton( TRect( 28, 16, 38, 18 ), "~S~ave", cmSaveMat,
bfNormal ));

 pd->insert( new TButton( TRect( 39, 16, 49, 18 ), "~O~K", cmOK,
bfDefault ));

int control=deskTop->execView( pd );
        if( control == cmOK||control==cmItemSelect)
          {
           Box->getData(Materials1);
           Box1->getData(Materials_data[Materials1->select]);
           strcpy(WorkPieceData1->MaterialTypeLine,
          (char *)Materials1->data->at(Materials1->select));
           d->setData(WorkPieceData1->MaterialTypeLine);
           strcpy(WorkPieceData1->MaterialLine,
           (char *)Materials_data[Materials1->select]->Data
           ->at(Materials_data[Materials1->select]->select));
           d1->setData(WorkPieceData1->MaterialLine);
          }
           stpcpy(Index->vipi,MatBase[Materials1->select]->index);
  }
    destroy(pd);
          }
}

void TMyApp::add(int cri)
{
  TDialog *pd = new TDialog( TRect(12, 5, 68, 18), "Material Type" );
  if( !pd )
  return;

        TView *B;
        B = new TInputLine( TRect(3, 3, 25, 4 ), 40 );
        pd->insert( B );
        pd->insert( new TLabel( TRect(2, 2, 9, 3 ),
                "Index", B ));

        TView* b = new TInputLine( TRect( 3, 5, 25, 6 ), 40 );
        pd->insert( b );
        pd->insert( new TLabel( TRect( 2, 4, 25, 5 ),
```

```
            "Material Group", b ));

b = new TInputLine( TRect(3, 7, 25, 8 ), 40 );
pd->insert( b );
pd->insert( new TLabel( TRect(2, 6, 25, 7 ),
            "Thermal Conductivity", b ));

b = new TInputLine( TRect( 3, 9, 25, 10 ), 40 );
pd->insert( b );
pd->insert( new TLabel( TRect( 2, 8, 25, 9 ),
            "Specific Heat ", b ));

b = new TInputLine( TRect(29, 3, 51, 4 ), 40 );
pd->insert( b );
pd->insert( new TLabel( TRect(28, 2, 51, 3 ),
            "Critical Temperature ", b ));

b = new TInputLine( TRect( 29, 5, 51, 6 ), 40 );
pd->insert( b );
        pd->insert( new TLabel( TRect( 28, 4, 51, 5 ),
                    "Melting Temperature ", b ));

        b = new TInputLine( TRect( 29, 7, 51, 8 ), 40 );
        pd->insert( b );
        pd->insert( new TLabel( TRect( 28, 6, 51, 7 ),
                    "Density ", b ));

pd->insert( new TButton( TRect( 28, 9, 38, 11 ), "~O~K", cmOK,
                         bfDefault ));
pd->insert( new TButton( TRect( 39, 9, 49, 11 ), "~C~ancel", cmCancel,
                         bfNormal ));
if(cri==0)
pd->setData(MatBase[Box->focused]);
else pd->setData(MatBase[Box->focused]);
B->select();
 ushort control=deskTop->execView(pd);
 if(control!=cmCancel)
{
 pd->getData(MatBase1);
if(cri==0){
 Materials1->data->atInsert(Box->focused,
 newStr(MatBase1->material));
 Box->getData(Materials1);
 int number2=Materials1->data->getCount()-Box->focused-1;
 memmove(&MatBase[Box->focused+1],
 &MatBase[Box->focused], number2*sizeof(void*));
MatBase[Box->focused]=new MATBase;
 memcpy(MatBase[Box->focused],MatBase1,sizeof(struct MATBase));
 memmove(&Materials_data[Box->focused+1],
 &Materials_data[Box->focused], number2*sizeof(void*));
Materials_data[Box->focused]=new MM;
 Materials_data[Box->focused]->Data=new TwpieceCollection(30,10);
 Materials_data[Box->focused]->select=0;
List->Data->removeAll();
 List->select=0;
 Box1->setRange(0);
Box1->drawView();
}
else{
  Materials1->data->atPut(Box->focused, newStr(MatBase1->material));
  memcpy(MatBase[Box->focused],MatBase1,sizeof(struct MATBase));
 }
  Box->getData(Materials1);
  Box->setRange(Materials1->data->getCount());
 Box->drawView();
 }
  destroy( pd);
}

void TMyApp::add1(int cri)
{
  TDialog *pd = new TDialog( TRect(20, 7, 60, 16), "Material" );
  if( !pd )
  return;

   TView* b = new TInputLine( TRect( 8, 3, 32, 4 ), 40 );
         pd->insert( b );
         pd->insert( new TLabel( TRect( 7, 2, 20, 3 ),
                     "Material", b ));
```

```
pd->insert( new TButton( TRect( 9, 5, 19, 7 ), "~O~K", cmOK,
                    bfDefault ));
pd->insert( new TButton( TRect( 20, 5, 30, 7 ), "~C~ancel", cmCancel,
                    bfNormal ));
if(cri==0)
pd->setData(addData);
else
 pd->setData(Materials_data[Box->focused]->Data->at(Box1->focused));
b->select();
 ushort control=deskTop->execView(pd);
 if(control!=cmCancel)
{
pd->getData(addData);
if(cri==0){
 Materials_data[Box->focused]->Data->atInsert(Box1->focused,
newStr(addData->MaterialLine));

 List->Data->atInsert(Box1->focused, newStr(addData->MaterialLine));
  }
  else {
   List->Data->atPut(Box1->focused, newStr(addData->MaterialLine));
   Materials_data[Box->focused]->Data->atPut(Box1->focused,
  newStr(addData->MaterialLine));
  }
   Box1->setRange(List->Data->getCount());
  Box1->drawView();
  }
  destroy( pd);
}

/*-------------------------------------------------------------------------------
                        Case Base

-------------------------------------------------------------------------------*/

 pd = new CaseDialog( TRect(3, 0, 77, 23), "Case Base" );
  if( pd )
  {
```

```
TView *b;
b = new Casedisplay( TRect( 11, 6, 36, 7 ), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 6, 11 , 7), "Index", b ));

b = new Casedisplay( TRect( 17, 7, 36, 8), 20 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 7, 17, 8), "Part Number", b ));

b = new Casedisplay( TRect( 13, 8, 36, 9 ), 20 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 8, 13, 9), "Process", b ));

b = new Casedisplay( TRect( 18, 9, 36, 10), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 9, 18, 10),"Machine tool", b ));

b = new Casedisplay( TRect( 14, 10, 36, 11), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 10, 14, 11),"Power kw", b ));

b = new Casedisplay( TRect( 13, 11, 36, 12), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 11, 13, 12),"Mater g", b ));

b = new Casedisplay( TRect( 14, 12, 36, 13), 20 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 12, 14, 13),"Material", b ));

b = new Casedisplay( TRect( 17, 13, 36, 14), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 13, 17, 14), "Hardness Rc", b ));

b = new Casedisplay( TRect( 25, 14, 36, 15 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 14, 25, 15), "Max roughness Ra um", b ));

b = new Casedisplay( TRect( 23, 15, 36, 16 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 15, 23, 16), "Size tolerance um", b ));
```

```
b = new Casedisplay( TRect( 22, 16, 36, 17 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 16, 22, 17), "Roundness tol um", b ));

b = new Casedisplay( TRect( 18, 17, 36, 18), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 17, 18, 18), "Start Dia mm", b ));

b = new Casedisplay( TRect( 19, 18, 36, 19), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 18, 19, 19),"Finish Dia mm", b ));

b = new Casedisplay( TRect( 14, 19, 36, 20 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 4, 19, 14, 20), "Width mm", b ));

b = new Casedisplay( TRect( 45, 6, 70, 7 ), 20 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 6, 45, 7), "Wheel", b ));

b = new Casedisplay( TRect( 54, 7, 70, 8 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect(38, 7, 54, 8), "Wheelspeed m/s", b ));

b = new Casedisplay( TRect( 56, 8, 70, 9 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 8, 56, 9), "Wheeldiameter mm", b ));

b = new Casedisplay( TRect( 47, 9, 70, 10), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 9, 47, 10), "Dresser", b ));

b = new Casedisplay( TRect( 47, 10, 70, 11 ), 30 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 10, 47, 11), "Coolant", b ));


b = new Casedisplay( TRect(53, 11, 70, 12 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 11, 53, 12), "Workspeed m/s", b ));

b = new Casedisplay( TRect(53, 12, 70, 13 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 12, 53, 13), "feedrate mm/s", b ));

b = new Casedisplay( TRect(57, 13, 70, 14 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 13, 57, 14), "Dressing depth mm", b ));

b = new Casedisplay( TRect(60, 14, 70, 15), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 14, 60, 15), "Dressing lead mm/rev", b ));

b = new Casedisplay( TRect(56, 15, 70, 16 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 15, 56, 16), "Spark-out time s", b ));

b = new Casedisplay( TRect(62, 16, 70, 17 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 16, 62, 17),"Equivalent diameter mm",b ));

b = new Casedisplay( TRect(61, 17, 70, 18 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 17, 61, 18), "Sp removal rate mm2/s", b ));

b = new Casedisplay( TRect(61, 18, 70, 19 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 18, 61, 19), "Specific energy J/mm3", b ));

b = new Casedisplay( TRect(56, 19, 70, 20 ), 10 );
pd->insert( b );
pd->insert( new TLabel( TRect( 38, 19, 56, 20), "Grinding ratio G", b ));

pd->insert( new TButton( TRect( 8, 21, 18, 23 ), "~S~ave", cmSaveCase,
            bfDefault ));
pd->insert( new TButton( TRect( 20, 21, 30, 23 ), "~C~ancel", cmCancel,
bfNormal ));
pd->insert( new TButton( TRect( 32, 21, 42, 23 ), "~D~elete", cmDeletec,
bfNormal ));
```

```
    pd->insert( new TButton( TRect( 44, 21, 54, 23 ), "~A~dd", cmAddc,
    bfNormal ));
    pd->insert( new TButton( TRect( 56, 21, 66, 23 ), "~E~dit", cmEditc,
    bfNormal ));

    memcpy(CaseData1,(CaseData*)CaseCollect->at(0),sizeof(struct CaseData));

    TRect r(5, 4, 70, 5);
    sr2=new TScrollBar(r);
    pd->insert(sr2);
    Box2=new TListBox1( TRect( 5, 2, 70, 4), 3, sr2);
    Box2->newList(IndexData1->data);
    pd->insert(Box2);
    pd->insert( new TLabel( TRect( 4, 1, 12, 2 ),
            "~I~ndex", Box2 ));
    pd->setData(CaseData1);

}
    deskTop->execView( pd );
      pd->getData(CaseData1);
       destroy(pd);
}

/*-------------------------------------------------------------------------------
                        Case Base Management
-------------------------------------------------------------------------------*/

void CaseDialog::handleEvent(TEvent& event)

{  TDialog::handleEvent(event);

  if( event.what == evCommand )
  {
      switch( event.message.command )
      {
       case cmDeletec:
          int Ps=password();
          if(Ps==0)
          {
           CaseCollect->atFree(Box2->focused);
           IndexData1->data->atFree(Box2->focused);
            Box2->setRange(IndexData1->data->getCount());
           Box2->drawView();
            pd->setData(CaseCollect->at(Box2->focused));
           }
          pd->drawView();
      break;
      case cmSaveCase:
       ofpstream oCC("CaseCol.txt");
       oCC<<CaseCollect;
       oCC.close();
       ofpstream oCI("IndexCol.txt");
       oCI<<IndexData1->data;
       oCI.close();
       break;
      case cmAddc:
          Ps=password();
          if(Ps==0){
          CaseData *A=new CaseData;
          pd->getData(A);
           pd->getData(CaseData1);
           CaseCollect->insert(A);
           IndexData1->data->insert(newStr(A->Index));
            Box2->setRange(IndexData1->data->getCount());
           Box2->drawView();
           pd->drawView();
           }
           else{
            pd->setData(CaseData1);
           pd->drawView();
           }
           break;
      case cmEditc:
          Ps=password();
          if(Ps==0){
          pd->getData(CaseData1);
          CaseData *A=new CaseData;
          pd->getData(A);
           CaseCollect->atRemove(Box2->focused);
```

```
    CaseCollect->insert(A);
    IndexData1->data->atRemove(Box2->focused);
    IndexData1->data->insert(newStr(A->Index));
     Box2->setRange(IndexData1->data->getCount());
   Box2->drawView();
   pd->drawView();
   }
   else{
   pd->setData(CaseData1);
   pd->drawView();
   }
   break;
case cmSaveNew:
   CaseData *B=new CaseData;
   pdr->getData(B);
   pdr->getData(recvalue);
    CaseCollect->insert(B);
    IndexData1->data->insert(newStr(B->Index));
     ofpstream oCC1("CaseCol.txt");
            oCC1<<CaseCollect;
            oCC1.close();
             ofpstream oCI1("IndexCol.txt");
           oCI1<<IndexData1->data;
            oCI1.close();
           break;
        case cmOldCase:
           app->oldcase();
           break;

        default:
           return;
         }clearEvent(event);
        }
if( event.what == evKeyDown){
 switch(event.keyDown.keyCode){

default:return;
  //clearEvent(event);
}
```

```
  }
  if( event.what == evBroadcast&&
   event.message.command==cmChangItem2){
   memcpy(CaseData1, CaseCollect->at(Box2->focused),sizeof(struct CaseData));
  pd->setData(CaseData1);
  }
}

void TMyApp::Case()
{

     ifpstream iCC("CaseCol.txt");
    iCC>>CaseCollect;
    iCC.close();
     ifpstream iCI("IndexCol.txt");
    iCI>>IndexData1->data;
    iCI.close();

/*-------------------------------------------------------------------------------------------------
                        Case Based Reasoning
-------------------------------------------------------------------------------------------------*/

void TMyApp::match()
{
 int rangeInd=IndexData1->data->getCount();
 int materialmatch=1;
 float maxsim=0;
 bestnum=-1;

 //***********match part number*******************

 for(int i=0; i<rangeInd;i++)
 {
 int numbermatch=strcmp(WorkPieceData1->Partnumber,
 (const char*)(((CaseData *)(CaseCollect->at(i)))->Number));
 if(numbermatch==0)
 bestnum=i;
 }
 if(bestnum!=-1)
```

```
 Recommendation(4);

//**************match  Index**************
 else
 {

 if(WorkPieceData1->WheelRadioData == 0)

 //******** wheel is selected by the system********

 {
 for(int i=0; i<rangeInd;i++)
 {
 int vipmatch=strnicmp(Index->vipi,(const char*)IndexData1->data->at(i),4);
 if(vipmatch==0)
 {
  if(strcmp(Index->mater,
  (const char*)(((CaseData *)(CaseCollect->at(i)))->material))==0)
  materialmatch=0;
   float caserough=atof(((CaseData*)(CaseCollect->at(i)))->Roughness);
   float roughmatch=(Index->roughi-caserough)/0.3;
   float caseDw=atof(((CaseData*)(CaseCollect->at(i)))->Dw1);
   float Dwmatch=(Index->diaWorki-caseDw)/500.00;
   float simility=(2*(1-materialmatch)+2*(1-roughmatch*roughmatch)+
  (1-Dwmatch*Dwmatch))/5;
   if(maxsim<simility){
   maxsim=simility;
  bestnum=i;
  }}
}

if(bestnum!=-1){
float Dwo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Dw1);
float Dso=atof(((CaseData*)(CaseCollect->at(bestnum)))->Ds);
float Vfo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Vf);
float Vwo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Vw);
float Deo=(Dwo*Dso)/(Dwo+Dso);
float Rao=atof(((CaseData*)(CaseCollect->at(bestnum)))->Roughness);
float fdo=atof(((CaseData*)(CaseCollect->at(bestnum)))->fd);
float Ran=Index->roughi;
float Dwn=Index->diaWorki;
    Den=(Dwn*Dso)/(Dwn+Dso);
 Vfn=Vfo*(Dwo/Dwn);
 Vwn=Vwo*Den/Deo;
 fdn=fdo*Ran*Ran/(Rao*Rao);
  Sr=3.1416*Vfn*Dwn;
Recommendation(0);
}
else{
Wts->wts=0;
if(WorkPieceData1->HardnessRadioData>=1)
WheelT();
Recommendation(1);
}
}

else

//**********wheel is selected by the user*****

{
 TDialog *pdw=new TDialog( TRect(13, 4, 67, 19), "Wheel Input" );
  if( pdw )
 {
  TView* b = new TRadioButtons( TRect( 4, 3, 20, 6),
            new TSItem( "A          ",
            new TSItem( "C          ",
            new TSItem( "B          ",0)
                  )));
        pdw->insert( b );
        pdw->insert( new TLabel( TRect( 3, 2, 18, 3), "~A~brasive type", b ));

   b = new TRadioButtons( TRect( 4, 8, 20, 13),
            new TSItem( "V          ",
            new TSItem( "E          ",
            new TSItem( "B          ",
            new TSItem( "R          ",
            new TSItem( "M          ",0)
                  )));
        pdw->insert( b );
```

```
        pdw->insert( new TLabel( TRect( 3, 7, 18, 8), "~B~ond type", b ));

b = new TInputLine( TRect( 38, 3, 48, 4 ), 10 );
pdw->insert( b );
pdw->insert( new TLabel( TRect( 25, 3, 38, 4 ),
                    "~G~rits", b ));

b = new TInputLine( TRect( 38, 5, 48, 6 ), 10 );
pdw->insert( b );
pdw->insert( new TLabel( TRect( 25, 5, 38, 6 ),
                    "G~r~ade", b ));

b = new TInputLine( TRect( 38, 7, 48, 8 ), 10 );
pdw->insert( b );
pdw->insert( new TLabel( TRect( 25, 7, 38, 8 ),
                    "~D~iameter mm", b ));

  b = new TInputLine( TRect( 38, 9, 48, 10 ), 10 );
  pdw->insert( b );
  pdw->insert( new TLabel( TRect( 25, 9, 38, 10 ),
                      "~S~peed mm/s", b ));

  pdw->insert( new TButton( TRect( 31, 12, 41, 14 ), "~O~K", cmOK,
  bfDefault ));


 pdw->setData(userwheel);

 }
    int control=deskTop->execView( pdw );

    if(control!=cmCancel){
          pdw->getData(userwheel);
          destroy(pdw); }
          else
          { destroy(pdw);
          return;}

if(userwheel->wheeltype==0)
wt=newStr("A");
if(userwheel->wheeltype==1)
wt=newStr("C");
if(userwheel->wheeltype==2)
wt=newStr("B");

if(userwheel->bondtype==0)
bt="V";
if(userwheel->bondtype==1)
bt="E";
if(userwheel->bondtype==2)
bt="B";
if(userwheel->bondtype==3)
bt="R";
if(userwheel->bondtype==4)
bt="M";
float grits=atof(userwheel->grits);
char *cgrit;

if(grits<=40)
cgrit="1";
else if(grits<=53)
cgrit="2";
else if(grits<=70)
cgrit="3";
else if(grits<=90)
cgrit="4";
else if(grits<=110)
cgrit="5";
else if(grits<=130)
cgrit="6";
Index->vipi[4]='\0';
strcat(Index->vipi,wt);
strcat(Index->vipi,bt);
strcat(Index->vipi,cgrit);

for(int i=0; i<rangeInd;i++)
{
 int vipmatch=strnicmp(Index->vipi,(const char*)IndexData1->data->at(i),7);
 if(vipmatch==0)
{
```

```
if(strcmp(Index->mater,
(const char*)(((CaseData *)(CaseCollect->at(i)))->material))==0)
materialmatch=0;
 float caserough=atof(((CaseData*)(CaseCollect->at(i)))->Roughness);
 float roughmatch=(Index->roughi-caserough)/0.3;
 float caseDw=atof(((CaseData*)(CaseCollect->at(i)))->Dw1);
 float Dwmatch=(Index->diaWorki-caseDw)/500.00;
 float inputDs=atof(userwheel->diameter);
 float caseDs=atof(((CaseData*)(CaseCollect->at(i)))->Ds);
 float Dsmatch=(inputDs-caseDs)/100.00;
 float inputVs=atof(userwheel->speed);
 float caseVs=atof(((CaseData*)(CaseCollect->at(i)))->Vs);
 float Vsmatch=(inputVs-caseVs)/15.00;
 float simility=(2*(1-materialmatch)+2*(1-roughmatch*roughmatch)+
(1-Dwmatch*Dwmatch)+(1-Dsmatch*Dsmatch)+(1-Vsmatch*Vsmatch))/7;
 if(maxsim<simility){
 maxsim=simility;
 bestnum=i;
 }

 }
 if(bestnum!=-1){
 float Dwo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Dw1);
 float Dso=atof(((CaseData*)(CaseCollect->at(bestnum)))->Ds);
 float Vfo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Vf);
 float Vwo=atof(((CaseData*)(CaseCollect->at(bestnum)))->Vw);
 float Vso=atof(((CaseData*)(CaseCollect->at(bestnum)))->Vs);
 float Deo=(Dwo*Dso)/(Dwo+Dso);
 float Rao=atof(((CaseData*)(CaseCollect->at(bestnum)))->Roughness);
 float fdo=atof(((CaseData*)(CaseCollect->at(bestnum)))->fd);
 float Ran=Index->roughi;
 float Dwn=Index->diaWorki;
 float Dsn=atof(userwheel->diameter);
 float Vsn=atof(userwheel->speed);
     Den=(Dwn*Dsn)/(Dwn+Dso);
  Vfn=Vfo*(Dwo*Vsn)/(Dwn*Vso);
  Vwn=Vwo*(Den*Vsn)/(Deo*Vso);
  fdn=fdo*Ran*Ran/(Rao*Rao);
  Sr=3.1416*Vfn*Dwn;
 Recommendation(2);
 }
 else
 Recommendation(3);
 }}
 }

 /*-----------------------------------------------------------------------------------------------
                                    Rule Based Reasoning

 -----------------------------------------------------------------------------------------------*/

 //**********Carbon and Alloy Steels**********
     if(Materials1->select==0||Materials1->select==1){
     strcpy(recvalue->Vs,"28-33");
     strcpy(recvalue->Vw,"0.35-0.5");
      Vfn=0.013/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
     if(Index->roughi>=0.9 )
      Vfn=0.05/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
      if(strncmp(hard,"H",1)==0)
      strcpy(recvalue->Coolant,"Oils-Heavy Duty");
     else
      strcpy(recvalue->Coolant,"Emulsifiable-Light");
     }

  //***********Tool  Steels********************
     if(Materials1->select==2){
     if(strncmp(hard,"L",1)==0){
      strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
      strcpy(recvalue->Vs,"28-33");
      strcpy(recvalue->Vw,"0.3-0.5");
       Vfn=0.013/2*0.30*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
     if(Index->roughi>=0.9 )
       Vfn=0.05/2*0.30*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
     }
     else
     {
      strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
      strcpy(recvalue->Vs,"20-30");
      strcpy(recvalue->Vw,"0.3-0.5");
```

```
  Vfn=0.01/2*0.3*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
 if(Index->roughi>=0.9 )
  Vfn=0.05/2*0.30*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
 }
 }

//*************High Speed Steels***************
 if(Materials1->select==3){
 if(strncmp(hard,"L",1)==0){
  strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
  strcpy(recvalue->Vs,"20-30");
  strcpy(recvalue->Vw,"0.3-0.5");
  Vfn=0.01/2*0.3*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
 if(Index->roughi>=0.9 )
  Vfn=0.025/2*0.3*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
 }
 else
 {
  strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
  strcpy(recvalue->Vs,"20-28");
  strcpy(recvalue->Vw,"0.3-0.5");
   Vfn=0.008/2*0.3*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.025/2*0.3*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }
  }

//*******************Superlloys*********************
  if(Materials1->select==4){
  //if(strncmp(hard,"L",1)==0){
  strcpy(recvalue->Coolant,"Oils-Heavy");
  strcpy(recvalue->Vs,"15-18");
  strcpy(recvalue->Vw,"0.25-0.5");
   Vfn=0.005/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.025/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }

//*************Stainless Steels & NiAlloys************
  if(Materials1->select==5||Materials1->select==6||Materials1->select==10){
  strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
  strcpy(recvalue->Vs,"28-33");
  strcpy(recvalue->Vw,"0.25-0.5");
   Vfn=0.013/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.05/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }

//*****************Cast Iron*********************
  if(Materials1->select==7){
  if(strncmp(hard,"L",1)==0){
  strcpy(recvalue->Coolant,"Emulsifiable-Light");
  strcpy(recvalue->Vs,"28-33");
  strcpy(recvalue->Vw,"0.35-0.5");
   Vfn=0.025/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.05/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }
  else
  {
  strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
  strcpy(recvalue->Vs,"28-33");
  strcpy(recvalue->Vw,"0.35-0.5");
   Vfn=0.013/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.05/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }
  }

//*****************Aluminum Alloys***************
  if(Materials1->select==8){
  strcpy(recvalue->Coolant,"Oils-Light Duty");
  strcpy(recvalue->Vs,"28-33");
  strcpy(recvalue->Vw,"0.25-0.77");
   Vfn=0.013/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  if(Index->roughi>=0.9 )
   Vfn=0.05/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }

//*******************Copper Alloys******************
```

```
    if(Materials1->select==9){
     strcpy(recvalue->Coolant,"Emulsifiable-Light");
     strcpy(recvalue->Vs,"28-33");
     strcpy(recvalue->Vw,"0.35-0.5");
      Vfn=0.013/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
    if(Index->roughi>=0.9 )
      Vfn=0.05/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
   }

//*******************Ti  Alloys**********************
    if(Materials1->select==11){
     strcpy(recvalue->Coolant,"Special-Light");
     strcpy(recvalue->Vs,"15-20");
     strcpy(recvalue->Vw,"0.35-0.5");
      Vfn=0.013/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
    if(Index->roughi>=0.9 )
      Vfn=0.05/2*0.35*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
   }

//*****************CBN  Wheel************************
   if(strcmp(wt,"B")==0){
    strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
    strcpy(recvalue->Vs,"25-38");
    strcpy(recvalue->Vw,"0.25-0.5");
     Vfn=0.005/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
   if(Index->roughi>=0.9 )
     Vfn=0.025/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }

  //User selecting wheel
  if(WL==1)
 {
  strcpy(recvalue->Vs,userwheel->speed);
  strcpy(recvalue->Ds,userwheel->diameter);
  strcpy(recvalue->Wheel,wt);
  strcat(recvalue->Wheel,userwheel->grits);
  strcat(recvalue->Wheel,userwheel->grade);
 if(strcmp(wt,"B")==0){          //CBN
  strcat(recvalue->Wheel,"100");
   strcpy(recvalue->Coolant,"Emulsifiable-Heavy");
    strcpy(recvalue->Vw,"0.25-0.5");
     Vfn=0.005/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
   if(Index->roughi>=0.9 )
     Vfn=0.025/2*0.25*1000/(3.1416*atof(WorkPieceData1->DiameterLine));
  }
   strcat(recvalue->Wheel,bt);
  }

//*****************Dressing  parameter*************
    if(Index->roughi>=0.9  ){
     strcpy(recvalue->fd,"0.18");
     strcpy(recvalue->ad,"0.025");
    }
    else {
     strcpy(recvalue->fd,"0.10");
      strcpy(recvalue->ad,".012-.019");
    }
    char buf[10];
     gcvt(Vfn,2,buf);
    strcpy(recvalue->Vf,"< ");
    strcat(recvalue->Vf,buf);
 }

/*-------------------------------------------------------------------------------------------
                              Neural Network for Wheel Selection

-------------------------------------------------------------------------------------------*/

 void encode()
 {
if(Materials1->select==0||Materials1->select==1||Materials1->select==2
||Materials1->select==3) wheelData->radioMaterialData=0;
else if(Materials1->select==4)wheelData->radioMaterialData=2;
else wheelData->radioMaterialData=Materials1->select-2;
wheelData->radioHardData=WorkPieceData1->HardnessRadioData;
roughness=atof(WorkPieceData1->RoughnessLine);
wheelData->checkBoxSeverityData=WorkPieceData1->SurConCheckData;

float a1=0.0;float a2=0.0;float a3=0.0; float a4=0.0; float a6=0.0;
float b1=0.0;float b2=0.0; float b5=0.0; float b6=0.0;
```

```
float ax=0.0;float ai=0.0;float bx=0.0;float bi=0.0;
if(wheelData->radioMaterialData==0||
wheelData->radioMaterialData==1)
inputData[4]=0.0;
if(wheelData->radioMaterialData>=4)
inputData[4]=1.0;

if(wheelData->radioHardData==0)
inputData[0]=0.3;
if(wheelData->radioHardData==1)
inputData[0]=0.5;
if(wheelData->radioHardData==2)
inputData[0]=0.7;
if(wheelData->radioMaterialData!=0&&
wheelData->radioMaterialData!=1&&
wheelData->radioMaterialData!=5)
inputData[0]=0.7;

if(roughness>=0.9)
inputData[1]=0.3;
else if(roughness<0.9&&roughness>=0.7)
inputData[1]=0.4;
else if(roughness<0.7&&roughness>=0.4)
inputData[1]=0.5;
else if(roughness<0.4&&roughness>=0.20)
inputData[1]=0.6;
else if(roughness<0.20)
inputData[1]=0.7;

inputData[2]=0.1;inputData[3]=0.1;

if(wheelData->checkBoxSeverityData&1){a1=1;b1=-1;}
if(wheelData->checkBoxSeverityData&2){a2=2;b2=1;}
if(wheelData->checkBoxSeverityData&4){a3=-1;}
if(wheelData->checkBoxSeverityData&8){a4=1;}
if(wheelData->checkBoxSeverityData&16){a6=-1;b6=-1;}
if(wheelData->checkBoxSeverityData&32){a6=1;b6=1;}
b5=1;

ax=max(a1,a2);ax=max(ax,a3);ax=max(ax,a4);ax=max(ax,a6);
ai=min(a1,a2);ai=min(ai,a3);ai=min(ai,a4);ai=min(ai,a6);
bx=max(b1,b2);bx=max(bx,b5);bx=max(bx,b6);
bi=min(b1,b2);bi=min(bi,b5);bi=min(bi,b6);
inputData[2]=(ax+ai)/10.0+0.1;inputData[3]=(bx+bi)/10.0+0.1;

}

char *getWheel()
{
 wheelData->radioWheelTypeData=Wts->wts;
 char wheeltype[4],gritsize[4],grade[2],bond[2],recom[20];
 if(wheelData->radioMaterialData==2||wheelData->radioMaterialData==3)
  outputData[0]=0.4;
 if(wheelData->radioWheelTypeData==0){
 bt="V";
 if(outputData[0]<0.26)
  strcpy( wheeltype,"11A");
 else if(outputData[0]<0.5)
  strcpy( wheeltype,"48A");
 else if(outputData[0]<0.65)
 strcpy( wheeltype,"51A");
 if(outputData[0]>=0.65&&outputData[0]<0.74)
 strcpy( wheeltype,"WA");
 wt="A";
 if(outputData[0]>0.75)
 {
  strcpy( wheeltype ,"C");
 wt="C";
  }
 }
  else {strcpy( wheeltype ,"B"); wt="B";bt="B";}

 if(wheelData->radioWheelTypeData==1)
 outputData[1]=outputData[1]+0.2;

 if(outputData[1]<1.15 && outputData[1]>0.05)
 {
 if(outputData[1]<0.15 )
 strcpy( gritsize,"24");
 else if(outputData[1]<0.25 )
```

```
 strcpy( gritsize,"30");
 else if(outputData[1]<0.35 )
  strcpy( gritsize,"36");
 else if(outputData[1]<0.45 )
  strcpy( gritsize,"46");
 else if(outputData[1]<0.55 )
 strcpy( gritsize,"60");
 else if(outputData[1]<0.65 )
 strcpy( gritsize,"80");
 else if(outputData[1]<0.75 )
  strcpy( gritsize,"100");
 else if(outputData[1]<0.85 )
  strcpy( gritsize,"120");
 else
  strcpy( gritsize,"150");
 }
 if(wheelData->radioWheelTypeData==1)
 outputData[1]=outputData[1]-0.2;

 if(wheelData->radioWheelTypeData==0){
 if(outputData[2]<0.15)
  strcpy( grade,"P");
 else if(outputData[2]>0.15 && outputData[2]<0.25)
  strcpy( grade,"O");
 else if(outputData[2]>0.25 && outputData[2]<0.35)
  strcpy( grade,"N");
 else if(outputData[2]>0.35 && outputData[2]<0.45)
  strcpy( grade,"M");
 else if(outputData[2]>0.45 && outputData[2]<0.55)
  strcpy( grade,"L");
 else if(outputData[2]>0.55 && outputData[2]<0.65)
  strcpy( grade,"K");
 else if(outputData[2]>0.65 && outputData[2]<0.75)
  strcpy( grade,"J");
 else if(outputData[2]>0.75 && outputData[2]<0.85)
  strcpy( grade,"I");
 else
  strcpy( grade,"H");
 strcpy(recom,wheeltype);
 strcat(recom,gritsize);
 strcat(recom,grade);
 strcat(recom,"V");
 }

 else {
 if(outputData[2]<0.35 )
  strcpy( grade,"X");
 else if(outputData[2]<0.45)
  strcpy( grade,"V");
 else if(outputData[2]>0.45 && outputData[2]<0.55)
  strcpy( grade,"U");
 else if(outputData[2]>0.55 && outputData[2]<0.65)
  strcpy( grade,"T");
 else if(outputData[2]>0.65 && outputData[2]<0.75)
  strcpy( grade,"S");
 else if(outputData[2]>0.75 && outputData[2]<0.85)
  strcpy( grade,"R");
 else
  strcpy( grade,"Q");
  strcpy(recom,wheeltype);
  strcat(recom,gritsize);
  strcat(recom,grade);
  strcat(recom,"100B");
 }
 char *cgrit;
 if(outputData[1]<=0.35)
 cgrit="1";
 else if(outputData[1]<=0.45)
 cgrit="2";
 else if(outputData[1]<=0.55)
 cgrit="3";
 else if(outputData[1]<=0.65)
 cgrit="4";
 else if(outputData[1]<=0.75)
 cgrit="5";
 else cgrit="6";

 Index->vipi[4]='\0';
 strcat(Index->vipi,wt);
 strcat(Index->vipi,bt);
```

```
 strcat(Index->vipi,cgrit);
 return recom;
}

void DisplayDialog::DisplayDialog(const TRect& bounds, const char *aTitle):
TDialog(bounds, aTitle),TWindowInit(&DisplayDialog::initFrame)
{
}

TMyApp *wa;
void DisplayDialog::handleEvent(TEvent& event)

{

   TDialog::handleEvent(event);

  if( event.what == evCommand )
  {
        switch( event.message.command )
        {
        case cmMore:
        wa->More();
        break;
        default:
           return;
         }
        clearEvent( event );         // clear event after handling
        }
}

void TMyApp::More()
{
 TDialog *pdm = new TDialog( TRect( 8, 4, 70, 17), "More Suggestions" );
   if (!pdm)
    return;

   // add some centered text to the dialog box
   pdm->insert(new TStaticText(TRect(2, 1, 60, 6),
        "\nThe abrasive type can be changed to a basicl"
        " similar type. For example, the type 48A can be"
        " replaced by 51A or 51A can be replaced by 48A."
        " The 11A can be replaced by the 48A. However, Type"
        " WA for grinding hardened steel and Type"

        ) );
   pdm->insert(new TStaticText(TRect(2, 6, 60, 9),
        "C for grinding low tensile strength materials such as"
        " cast iron and non-ferrous materials are not recommended"
        " replacements for ensuring the grinding performance."
        ));
   pdm->insert(new TButton(TRect(25, 10, 35, 12),
                    " OK", cmOK, bfDefault) );

   deskTop->execView(pdm);
   destroy(pdm);
}

void TMyApp::WheelDisplay()
{
Wts->wts=0;
if(WorkPieceData1->HardnessRadioData>=1)
app->WheelT();
encode();
 appl(inputData,outputData,"wn10.w");
char *recom;


DisplayDialog *pdw = new DisplayDialog( TRect( 18, 3, 62, 19), "OUTPUT" );
if (!pdw)
 return;

pdw->insert(new TStaticText(TRect(3, 1, 41, 3),
                    "\n\003The Recommendation is"
                    "\n\n\n\003"
                   ) );

 strcpy(recom ,"\003");
strcat(recom,getWheel());
pdw->insert(new WTStaticText(TRect(15, 4, 29, 5),
                    recom
```

```
                    ) );

pdw->insert(new TStaticText(TRect(3, 5, 41, 7),
                     "\n\003Alternatives are"
                     "\n\n\n\003"
                    ) );
float gd=outputData[2];
outputData[2]=gd-0.1;
 strcpy(recom ,"\003");
strcat(recom,getWheel());
pdw->insert(new WTStaticText(TRect(15, 8, 29, 9),
                    recom
                    ) );

outputData[2]=gd+0.1;
 strcpy(recom ,"\003");
strcat(recom,getWheel());
pdw->insert(new WTStaticText(TRect(15, 9, 29, 10),
                    recom
                    ) );

outputData[2]=gd;

 float gs=outputData[1];
 outputData[1]=gs+0.1;
  strcpy(recom ,"\003");
 strcat(recom,getWheel());
outputData[1]=gs;
 pdw->insert(new WTStaticText(TRect(15, 10, 29, 11),
                    recom
                    ) );

outputData[1]=gs-0.1;
 strcpy(recom ,"\003");
strcat(recom,getWheel());

 pdw->insert(new WTStaticText(TRect(15, 11, 29, 12),
                    recom
                    ) );
```

```
   // add an OK button
    pdw->insert(new TButton(TRect(10, 13, 22, 15),
                         "More",cmMore, bfDefault) );

    pdw->insert(new TButton(TRect(24, 13, 36, 15),
                         "OK", cmOK, bfDefault) );

   deskTop->execView(pdw);
   destroy(pdw);
}

void TMyApp::WheelT()
{
TDialog *pds=new TDialog( TRect(22, 6, 58, 16), "Wheel Selection" );
  if( pds )
 {
   TView* b = new TRadioButtons( TRect( 9, 3, 27, 5),
             new TSItem( "Conventional",
             new TSItem( "CBN ",0)
                       ));
          pds->insert( b );
          pds->insert( new TLabel( TRect( 8, 2, 26, 3), "~A~brasive type", b ));
          pds->insert( new TButton( TRect( 13, 7, 24, 9 ), "~O~K", cmOK,
  bfDefault ));
    pds->setData(0);
  }
     int control=deskTop->execView( pds );

     if(control!=cmCancel)
          pds->getData(Wts);
          destroy(pds);
}

/*-------------------------------------------------------------------------------
                         Recommendation Display

-------------------------------------------------------------------------------*/

void TMyApp::Recommendation(int M)
{
```

```
TCommandSet command;
        command+=cmOldCase;
        if(M==4){
        enableCommands(command);
        memcpy(recvalue,(CaseData*)CaseCollect->at(bestnum),sizeof(struct
        CaseData));
        }
        command+=cmOldCase;
        if(M==0){
        enableCommands(command);
        recValueCase(0);
        }
        if(M==1){
        recValueRule(0);
        disableCommands(command);
        }
        if(M==2){
        enableCommands(command);
        recValueCase(1);
        }
        if(M==3){
        recValueRule(1);
        disableCommands(command);
        }
pdr = new CaseDialog( TRect(3, 1, 77, 21), "Recommendations" );
 if( pdr )
 {
        TView *b;
        b = new Casedisplay( TRect( 11, 2, 36, 3 ), 30 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 2, 11 , 3), "Index", b ));

        b = new Casedisplay( TRect( 17, 3, 36, 4), 20 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 3, 17, 4), "Part Number", b ));

        b = new Casedisplay( TRect( 13, 4, 36, 5 ), 20 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 4, 13, 5), "Process", b ));

        b = new Casedisplay( TRect( 18, 5, 36, 6), 30 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 5, 18, 6),"Machine tool", b ));

        b = new Casedisplay( TRect( 14, 6, 36, 7), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 6, 14, 7),"Power kw", b ));

        b = new Casedisplay( TRect( 13, 7, 36, 8), 30 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 7, 13, 8),"Mater g", b ));

        b = new Casedisplay( TRect( 14, 8, 36, 9), 20 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 8, 14, 9),"Material", b ));

        b = new Casedisplay( TRect( 17, 9, 36, 10), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 9, 17, 10), "Hardness Rc", b ));

        b = new Casedisplay( TRect( 25, 10, 36, 11 ), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 10, 25, 11), "Max roughness Ra um", b ));

        b = new Casedisplay( TRect( 23, 11, 36, 12 ), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 11, 23, 12), "Size tolerance um", b ));

        b = new Casedisplay( TRect( 22, 12, 36, 13 ), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 12, 22, 13), "Roundness tol um", b ));

        b = new Casedisplay( TRect( 18, 13, 36, 14), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 13, 18, 14), "Start Dia mm", b ));

        b = new Casedisplay( TRect( 19, 14 ,36, 15), 10 );
        pdr->insert( b );
        pdr->insert( new TLabel( TRect( 4, 14, 19, 15),"Finish Dia mm", b ));
```

```
b = new Casedisplay( TRect( 14, 15, 36, 16 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 4, 15, 14, 16), "Width mm", b ));

b = new Casedisplay( TRect( 45, 2, 70, 3 ), 20 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 2, 45, 3), "Wheel", b ));

b = new Casedisplay( TRect( 54, 3, 70, 4 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect(38, 3, 54, 4), "Wheelspeed m/s", b ));

b = new Casedisplay( TRect( 56, 4, 70, 5 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 4, 56, 5), "Wheeldiameter mm", b ));

b = new Casedisplay( TRect( 47, 5, 70, 6), 30 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 5, 47, 6), "Dresser", b ));

b = new Casedisplay( TRect( 47, 6, 70, 7 ), 30 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 6, 47, 7), "Coolant", b ));


b = new Casedisplay( TRect(53, 7, 70, 8 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 7, 53, 8), "Workspeed m/s", b ));

b = new Casedisplay( TRect(53, 8, 70, 9 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 8, 53, 9), "feedrate mm/s", b ));

b = new Casedisplay( TRect(57, 9, 70, 10 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 9, 57, 10), "Dressing depth mm", b ));

b = new Casedisplay( TRect(60, 10, 70, 11), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 10, 60, 11), "Dressing lead mm/rev", b ));

b = new Casedisplay( TRect(56, 11, 70, 12 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect( 38, 11, 56, 12), "Spark-out time s", b ));


b = new Casedisplay( TRect(62, 12, 70, 13 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect(38, 12, 62, 13),"Equivalent diameter mm",b ));

b = new Casedisplay( TRect(61, 13, 70, 14 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect(38,13, 61, 14), "Sp removal rate mm2/s", b ));

b = new Casedisplay( TRect(61, 14, 70, 15 ), 10 );
pdr->insert( b );
pdr->insert( new TLabel( TRect(38, 14, 61, 15),"Specific energy J/mm3", b ));

b = new Casedisplay( TRect(56, 15, 70, 16 ), 10 );
pdr->insert( b );
        pdr->insert( new TLabel( TRect( 38, 15, 56, 16), "Grinding ratio G", b ));

        pdr->insert( new TButton( TRect( 8, 17, 28, 19 ), "~S~ave Case", cmSaveNew,
                bfNormal ));
        pdr->insert( new TButton( TRect( 32, 17, 52, 19 ), "Old ~C~ase", cmOldCase,
        bfNormal ));
        pdr->insert( new TButton( TRect( 56, 17, 66, 19 ), "~O~K", cmOK,
        bfDefault));
    pdr->setData(recvalue);
  }
        deskTop->execView( pdr );
         pdr->getData(recvalue);
          destroy(pdr);
}

void recValueCase(int WL)
{
 memcpy(recvalue,(CaseData*)CaseCollect->at(bestnum),sizeof(struct CaseData));
 strcpy(recvalue->Number, WorkPieceData1->Partnumber);
 strcpy(recvalue->Power, ProData->PowerLine);
```

```
 strcpy(recvalue->Machine, "");
 strcpy(recvalue->material, Index->mater);
 strcpy(recvalue->Hardness, hard);
 strcpy(recvalue->Roughness, WorkPieceData1->RoughnessLine);
 strcpy(recvalue->Tolerance, WorkPieceData1->ToleranceLine);
 strcpy(recvalue->Roundness, WorkPieceData1->Roundness);
 strcpy(recvalue->Dw1, WorkPieceData1->DiameterLine);
 strcpy(recvalue->Dw2, WorkPieceData1->Dw2);
 strcpy(recvalue->Width, WorkPieceData1->Width);
 strcpy(recvalue->ec, "");
 strcpy(recvalue->G, "");
 gcvt(Vwn,2,recvalue->Vw);
 gcvt(Vfn,2,recvalue->Vf);
 gcvt(fdn,2,recvalue->fd);
 gcvt(Den,4,recvalue->ED);
 gcvt(Sr,2,recvalue->SR);
if(WL==1)
{
 strcpy(recvalue->Vs,userwheel->speed);
  strcpy(recvalue->Ds,userwheel->diameter);
  strcpy(recvalue->Wheel,wt);
  strcat(recvalue->Wheel,userwheel->grits);
  strcat(recvalue->Wheel,userwheel->grade);
  strcat(recvalue->Wheel,bt);
 }
}

void recValueRule(int WL)
{
  strcpy(recvalue->Index,Index->vipi);
  strcpy(recvalue->Number, WorkPieceData1->Partnumber);
  strcpy(recvalue->Process, "External");
  strcpy(recvalue->Machine,"");
  strcpy(recvalue->Power, ProData->PowerLine);
  strcpy(recvalue->materialtype, WorkPieceData1->MaterialTypeLine);
  strcpy(recvalue->material, Index->mater);
  strcpy(recvalue->Hardness, hard);
  strcpy(recvalue->Roughness, WorkPieceData1->RoughnessLine);
  strcpy(recvalue->Tolerance, WorkPieceData1->ToleranceLine);
  strcpy(recvalue->Roundness, WorkPieceData1->Roundness);
  strcpy(recvalue->Dw1, WorkPieceData1->DiameterLine);
  strcpy(recvalue->Dw2, WorkPieceData1->Dw2);
  strcpy(recvalue->Width,WorkPieceData1->Width);
  strcpy(recvalue->Dresser,"Single Point Diamond");
  strcpy(recvalue->SOT,"");
  strcpy(recvalue->SR, "");
  strcpy(recvalue->ec, "");
  strcpy(recvalue->G, "");
  strcpy(recvalue->ED, "");

  if(WL==0){
  encode();
    appl(inputData,outputData,"wn10.w");
    strcpy(recvalue->Wheel,getWheel());
    strcpy(recvalue->Ds,"");

    strcpy(recvalue->Index,Index->vipi);}

/*-------------------------------------------------------------------------------------------------
                              Display Old Case
-------------------------------------------------------------------------------------------------*/
void TMyApp::oldcase()
{
 TDialog *pd1 = new CaseDialog( TRect(2, 2, 76, 22), "Old Case" );
  if( pd1 )
  {
        TView *b;
        b = new Casedisplay( TRect( 11, 2, 36, 3 ), 30 );
        pd1->insert( b );

        pd1->insert( new TLabel( TRect( 4, 2, 11 , 3), "Index", b ));

        b = new Casedisplay( TRect( 17, 3, 36, 4), 20 );
        pd1->insert( b );
        pd1->insert( new TLabel( TRect( 4, 3, 17, 4), "Part Number", b ));

        b = new Casedisplay( TRect( 13, 4, 36, 5 ), 20 );
        pd1->insert( b );
        pd1->insert( new TLabel( TRect( 4, 4, 13, 5), "Process", b ));
```

```
b = new Casedisplay( TRect( 18, 5, 36, 6), 30 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 5, 18, 6),"Machine tool", b ));

b = new Casedisplay( TRect( 14, 6, 36, 7), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 6, 14, 7),"Power kw", b ));

b = new Casedisplay( TRect( 13, 7, 36, 8), 30 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 7, 13, 8),"Mater g", b ));

b = new Casedisplay( TRect( 14, 8, 36, 9), 20 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 8, 14, 9),"Material", b ));

b = new Casedisplay( TRect( 17, 9, 36, 10), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 9, 17, 10), "Hardness Rc", b ));

b = new Casedisplay( TRect( 25, 10, 36, 11 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 10, 25, 11), "Max roughness Ra um", b ));

b = new Casedisplay( TRect( 23, 11, 36, 12 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 11, 23, 12), "Size tolerance um", b ));

b = new Casedisplay( TRect( 22, 12, 36, 13 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 12, 22, 13), "Roundness tol um", b ));

b = new Casedisplay( TRect( 18, 13, 36, 14), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 13, 18, 14), "Start Dia mm", b ));

b = new Casedisplay( TRect( 19, 14 ,36, 15), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 14, 19, 15),"Finish Dia mm", b ));

b = new Casedisplay( TRect( 14, 15, 36, 16 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 4, 15, 14, 16), "Width mm", b ));

b = new Casedisplay( TRect( 45, 2, 70, 3 ), 20 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 2, 45, 3), "Wheel", b ));

b = new Casedisplay( TRect( 54, 3, 70, 4 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect(38, 3, 54, 4), "Wheelspeed m/s", b ));

b = new Casedisplay( TRect( 56, 4, 70, 5 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 4, 56, 5), "Wheeldiameter mm", b ));

b = new Casedisplay( TRect( 47, 5, 70, 6), 30 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 5, 47, 6), "Dresser", b ));

b = new Casedisplay( TRect( 47, 6, 70, 7 ), 30 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 6, 47, 7), "Coolant", b ));

b = new Casedisplay( TRect(53, 7, 70, 8 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 7, 53, 8), "Workspeed m/s", b ));

b = new Casedisplay( TRect(53, 8, 70, 9 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 8, 53, 9), "feedrate mm/s", b ));

b = new Casedisplay( TRect(57, 9, 70, 10 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 9, 57, 10), "Dressing depth mm", b ));

b = new Casedisplay( TRect(60, 10, 70, 11), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 10, 60, 11), "Dressing lead mm/rev", b ));
```

```
b = new Casedisplay( TRect(56, 11, 70, 12 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect( 38, 11, 56, 12), "Spark-out time s", b ));

b = new Casedisplay( TRect(62, 12, 70, 13 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect(38, 12, 62,13),"Equivalent diameter mm",b ));

b = new Casedisplay( TRect(61, 13, 70, 14 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect(38, 13, 61, 14),"Sp removal rate mm2/s",b ));

b = new Casedisplay( TRect(61, 14, 70, 15 ), 10 );
pd1->insert( b );
pd1->insert( new TLabel( TRect(38, 14, 61, 15),"Specific energy J/mm3",b ));

b = new Casedisplay( TRect(56, 15, 70, 16 ), 10 );
        pd1->insert( b );
        pd1->insert( new TLabel( TRect( 38, 15, 56, 16), "Grinding ratio G", b ));

        pd1->insert( new TButton( TRect( 32, 17, 42, 19 ), "~O~K", cmOK,
        bfNormal ));
    pd1->setData(CaseCollect->at(bestnum));
  }
        deskTop->execView( pd1 );
          destroy(pd1);
}

/*-------------------------------------------------------------------------------
                              Main Programme

-------------------------------------------------------------------------------*/

int main()
{
  TMyApp myApp;
  myApp.run();
  return 0;
}
```

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

                Selection of Grinding Conditions
                        Version 1.0

            Copyright ©1995 AMT Research Laboratory

*********************************************************/

/*********************************************************

        Source file 3:      Password (Non-Turbor Vision Programme)

*********************************************************/

#include <dos.h>
#include <conio.h>
#include <string.h>

void txblock(int x1,int y1,int x2,int y2,int bkcolor,int txcolor,int tx)
{
 int i,j,m;
 for(j=y1;j<=y2;j++)
 {
   for(i=x1;i<=x2;i++)
   {
     m=((j-1)*80+(i-1))*2;
      poke(0xb800,m,(((bkcolor<<4)+txcolor)<<8)+tx);
   }
 }
}

void singlewrite(int x1,int y1,int ch)
{
 gotoxy(x1,y1);
 putch(ch);
}

void curon()
{
 union REGS regs;
 regs.h.ah=1;
 regs.h.ch=10;
 regs.h.cl=10;
 int86(0x10,&regs,&regs);
 regs.h.ah=2;
 regs.h.dh=11;
 regs.h.dl=41;
 regs.h.bh=0;
 int86(0x10,&regs,&regs);
}

void stringwrite(int x1,int y1,int txbackcolor,int txcolor,char *p)
{
   gotoxy(x1,y1);
   textbackground(txbackcolor);
   textcolor(txcolor);
  cprintf(p);
}

int shadow1(int x1,int y1,int x2,int y2)
{
 int i,j,m;
 for(j=y1+1;j<=y2+1;j++)
 {
   for(i=x1+2;i<=x2+2;i++)
   {
     m=((j-1)*80+(i-1))*2;
      pokeb(0xb800,m+1,7);
   }
 }
 return 0;
}

void frame1(int x1,int y1,int x2,int y2,int txbackcolor,int color)
{
 int i,j;
 textbackground(txbackcolor);
 textcolor(color);
```

```
 for(i=1;i<=x2-x1-1;i++)
 {
   singlewrite(x1+i,y1,196);
   singlewrite(x1+i,y2,196);
 }
 for(i=1;i<=y2-y1-1;i++)
 {
   singlewrite(x1,y1+i,179);
   singlewrite(x2,y1+i,179);
 }
   singlewrite(x1,y1,218);
   singlewrite(x2,y1,191);
   singlewrite(x1,y2,192);
   singlewrite(x2,y2,217);
}

int password()
{
 int i;char *pass; char buffer[3];
 pass="amt";
 shadow1(25,11,55,13);
 txblock(25,11,55,13,4,4,255);
 frame1(25,11,55,13,4,15);
 stringwrite(26,12,4,15,"Enter password:          ");
 curon();
 for(i=0;i<=2;i++)
 {
  buffer[i]=getch();
  cprintf("x");
 }
 if((i=strncmp(pass,buffer,3))!=0)
 {
   textcolor(14);
  cprintf(" WRONG");
   delay(800);
 }
 return i;
}
```

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

        Source file 4:      Neural Network Application Module

*********************************************************/

// See Feedforward Neural Network Source file 3.
```

Appendix II List of the Author's Relevant Published Papers

1. Rowe W B, Li Y, Inasaki I, Malkin S, 1994, Applications of artificial intelligence in grinding, Keynote Paper, Annals of CIRP, Vol 43, No 2, pp 521-531

2. Li Y, Mills B, Moruzzi J L, Rowe W B, 1994, Grinding wheel selection using a neural network, Proceedings of 10th National Manufacturing Research Conference, published by Taylor & Francis Ltd, Loughborough, 13-15 September , pp 597-601

3. Rowe W B, Li Y, Mills B, Allanson D R, Application of intelligent CNC in grinding, Accepted for publication in Journal of Computer in Industry, 1996.

4. Rowe W B, Li Y, Chen X, Mills B, Case based reasoning for selection of grinding conditions, Accepted for publication in Journal of Computer Integrated Manufacturing Systems, 1996