TOWARDS SYSTEMS ENGINEERING APPROACH APPLICABLE FOR SMALL DEVELOPING ORGANISATION

BY

S.R. JAROSLAWSKI

Thesis prepared in partial fulfillment of the requirements for the degree of MSc in Operations Management

I, the undersigned, hereby declare that the work contained in this thesis, is my own original work, and has not previously in its entirety, or part, been submitted at any university for a degree.

..... Date

ACKNOWLEDGEMENTS

The author wishes to thank Professor T. Ryan for his invaluable advice in the supervision of this thesis.

SYNOPSIS

The purpose of this thesis was to create the model which could be used to optimise the operational effectiveness of the general type of small developing business/system

The small developing organisation is the system which has the same type of components as a big organisation but the structure is much simpler. These components develop a system in the Technical, Economic and Environmental areas. Each of these components has its own purpose and together these purposes optimise the entire system. The identification of the purpose of the components of the small developing organisation should be analysed by:

- investigating the component's performance;
- investigating the component's behavioural pattern; and
- investigating the component's support requirement.

Further, the analysis of the general characteristics of the small developing organisation is such that:

- analysis of the performance considers only the main point of the component's operation but not the entire operational function;
- the behavioural pattern (i.e. failure and repair occurrence) is random;
- the evaluation of the support requirements is subjective where the statistical tools may not be applicable;
- the relationship between the elements of the aforementioned three activities is often unpredictable;
- the components are usually analysed according to the hierarchy of importance to the development of the entire system.

The systems engineering approach is suggested as the appropriate way to analyse a small developing organisation /system. Therefore, the author regards the small developing organisation as the small developing system. This system consists of subsystems

(components) which characterise the main activity of the system. The internal structure of subsystems consists of subsystem's attributes.

The author could not find the description of the systems engineering analysis applicable to the small developing organisation in the reviewed literature. Therefore, the research had to be based on literature used to analyse large organisations and the author's work experience.

The author uses the structure of the Technical System, proposed by M'Pherson[20,21], to describe the structure of the Technical Subsystem in the small developing system. According to M'Pherson[20] the effective Technical System, thus the Technical Subsystem in a small developing organisation, must have three attributes to accomplish its mission:

- Capability which is a function of a performance;
- Dependability which is a function of reliability and characterises behavioural pattern;
- Availability which is a function of support design.

The outputs of the attributes combine into a value of the subsystem's effectiveness. The value of effectiveness is further optimised against the resource requirement which is used to make the entire system operational. Therefore, the output of the subsystem is given in the form of a ratio of effectiveness to cost.

However, M'Pherson does not describe the attributes of the Economic and the Environmental Subsystem. These types of subsystem are, according to literature, evaluated by the Capability attribute. However, this may not apply to the optimisation of the small developing organisations because the lack of the Dependability and Availability attributes denies the developing character of the Economic and Environmental Subsystem. Thus all subsystems in the small developing system should have all three attributes.

The author uses the following tools to optimise the Capability, Dependability and Availability attributes:

- Dynamic Programming may be an appropriate tool to optimise performance which is a main element of the Capability attribute;
- Monte Carlo Simulation may be used to optimise the reliability which characterises the Dependability attribute;
- Fuzzy Logic may be used to optimise the support design which is the main element of the Availability attribute.

The examples of the subsystem are further analysed to investigate the hypothesis that any type of subsystem does, indeed, consist of the three attributes. The analysis leads not only to the fact that the subsystem has these three attributes, but also shows that the attributes are sufficient to evaluate the effectiveness of any subsystem. Since the effectiveness of different types of subsystems (e.g. Technical, Economic) is different in nature, therefore, the resource requirements of these types of subsystems will differ. The resource requirement is standardised as the Control Input Value. In order to optimise the output (i.e. the ratio of Effectiveness to Control Input Value Factor) of any subsystem, the Control Input Value is transformed into the Control Input Value Factor.

The next task is to find the relationships between the subsystems and the subsystem's attributes in order to complete the theoretical development of the model.

In the small developing organisations the main objective characterises the mission of the system and the secondary objective supports the system's mission. Therefore, the system consists of the one subsystem which is main and the other subsystems which are secondary.

The elements of the subsystem are also classified according to the hierarchy of importance. Performance is usually considered to be the most important measure of the subsystem's activity. Thus, Capability is the main attribute of the subsystem's structure. The elements of the Dependability and Availability attributes are usually selected by the optimised

performances values, therefore, Dependability and Availability are classified as the secondary attributes.

There may be two types of connections between the Main and the Secondary Subsystem:

- The performance (element of Capability) of the Main Subsystem may select the performance values of the Secondary Subsystem. Then, in the Secondary Subsystem, the optimised performances select the appropriate input elements of Dependability and Availability attributes.

or

- The optimised performances (element of Capability) of the Main Subsystem select elements of Dependability and Availability of the Main and Secondary Subsystems, and elements of Capability of the Secondary Subsystem.

The internal structure of the small detergent company, which is a practical example in this project, shows the second type of the above-mentioned connections. Therefore, this kind of connection between the Main and the Secondary Subsystem is applied to the model.

The theoretical model is programmed on the Delphi language supported by the Excel spreadsheet. The computer language code optimises the subsystem's attributes while the Excel spreadsheet selects input data and graphically displays the results of the optimisations.

The computer programme cannot optimise the system without the designer's knowledge of the structure of the analysed small developing organisation. The author suggests that the system should be analysed using a recursive structural model. Then the computer optimises a system in the form of the outputs of this system's subsystems. The key element in the process of the system optimisation is a Control Input Value.

The developed theoretical and the computer model is applied to a small detergent-manufacturing system. Three approaches are analysed in relation to the optimisation of the

manufacturing system. These are: Manufacturer's approach, M'Pherson's approach, and the author's approach, which is based on M'Pherson's theory.

The results of the optimisation of the application of the author's approach brought the author to the following conclusions:

I. The proposed systems engineering model may be used in a small sized developing system such as the detergent-manufacturing small business corporation. The model showed the ability to resolve the conflicting objectives of the Economic and Technical Subsystems through the ratios of Effectiveness to Control Input Value Factor.

II. The Capabilities, Dependabilities and Availabilities may evaluate not only the effectiveness of the Technical Subsystem (M'Pherson[20,21]), but these attributes can also be used to evaluate the Economic Subsystem.

III. The model may not be used in applications where one would exclusively use an objective analysis. Subjectivity is, nevertheless, a part of the management of the small developing system and therefore, the subjective approach can not be avoided. Thus, the small optimal developing system will always partially depend on the designer's intuition.

IV. Finally, it is difficult to validate that the final value of the presented model is the most optimal because:

- there are no objective mathematical models which one would apply for this particular example; and
- the author's model contains elements which are not considered in the other approaches (Manufacturer's approach does not consider Dependability and Availability for the Technical and Economic Subsystem; M'Pherson's approach does not consider Dependability and Availability for the Economic Subsystem).

THESIS OUTLINE

Chapter 1 firstly identifies the small developing business/system as a complex structure within its characteristic boundary; secondly, general characteristics of the nature of the

small developing business are described; and thirdly, systems engineering is suggested as an approach to optimise the small developing organisation/system.

In Chapter 2, the author investigates the systems engineering theory (B.S. Blanchard & W. Fabrycky[2] , M'Pherson[20,21]) which applies to small developing systems. From the theory he identifies the system's/subsystem's attributes which can play a crucial role in systems engineering. Therefore, methods used to optimise these attributes are analysed in Chapter 3.

In Chapter 4, it is suggested that any type of subsystems can be analysed using the same type of attributes. The author builds the general systems engineering model which may apply for any small developing organisation. The developed theoretical model is used as the basis of development for the computer software which is described in Chapter 5. The integration of the theory with the computer programme into the whole modeling structure is then given in Chapter 6.

In Chapter 7, the developed theoretical and computer models are applied to optimise a detergent-manufacturing system and conclusions are drawn with regard to the usefulness of the developed systems engineering model.

Chapter 8 ends the thesis with the author's reflections about the presented work.

TABLE OF CONTENTS

ACKNOWLEDGMENTS i

SYNOPSIS ii

TABLE OF CONTENTS viii

LIST OF ILLUSTRATIONS xiii

1 INTRODUCTION 1

1.1 Background 1
1.2 Small developing business corporation 2
1.2.1 Complexity of the small business organisation 3
1.2.2 Recognising the purpose of modeling the small business and its components 4
1.2.3 General characteristics of the small developing business 5
1.3. Dealing with complex structure 7
1.3.1 General systems analysis 8
1.3.2 Purpose of systems engineering applicable to small business 10
1.3.3 Nature of a system's models 10
1.4 Identification of the problem statement 11

2. SYSTEMS ENGINEERING THEORY IN THE SMALL BUSINESS CORPORATION 13

2.1 General description of system's structure 13
2.2 Inputs of System 14
2.2.1 Input elements of the Technical Subsystem applicable to the small business corporation 15
(a) Definition and measure of technical performance 16
(b) Definition and measure of technical reliability 17
(c) Definition and measure of technical maintainability 19
(d) Definition and measure of technical supportability design 21
2.2.2 Economic Subsystem's input element (economic performance) 23
(a) Interest and Interest formulas used to evaluate economic performance 24
(b) Break-even economic evaluation 25
2.2.3 Environmental Subsystem's input element (environmental performance) 26
(a) Impacts of the environment on the system 26
(b) Impacts of the system's environment on the external environment 27
2.3 Characteristics of the system's transformation function (subsystem) 28
2.3.1 Technical Subsystem's transformation functions (Technical

Subsystem's attributes) 30
(a) Technical Capability attribute 30
(b) Technical Dependability attribute 31
(c) Technical Availability attribute 33
2.3.2 Transfer function of the Economic Subsystem (economic Capability) 36
2.3.3 Transfer function of the Environmental Subsystem (environmental Capability) 36
2.3.4 Analysis of Life-Cycle Subsystem 37
2.4 Output of a system (evaluation of the worth of the whole system) 37
2.4.1 Output of the Technical Subsystem (Cost-effectiveness analysis) 37
2.4.2 Output of the Economic Subsystem (resource to the Capability analysis 39
2.4.3 Output of the Environmental Subsystem (resource to Capability analysis) 40
2.4.4 The combined technical, economic and environmental output of the system 40
2.5 Reflections about the presented theory 43
2.5.1 Criticism of the subsystem's models 43
2.5.2 Search for the optimization tools 43

3. SELECTION OF TOOLS FOR OPTIMISATION OF THE INPUT ELEMENTS CHARACTERISTIC FOR THE SUBSYSTEM'S ATTRIBUTES 45

3.1 Initial consideration of the analysis 45
3.2 Use of Dynamic Programming for optimising Capability 48
3.2.1 Reason for choosing dynamic programming 48
3.2.2 Characteristics of the dynamic programming optimisation 49
3.3 Use of Fuzzy Logic for Availability Optimisation 53
3.3.1 Reason for choosing Fuzzy Logic 53
3.3.2 Characteristics of Fuzzy Logic optimisation 54
3.4 Use of Monte Carlo Simulation for Dependability Optimisation 57
3.4.1 Reason for choosing Monte Carlo simulation 57
3.4.2 Characteristics of Monte Carlo simulation 58
3.5 Reflections about the presented tools 60

4. THE DEVELOPMENT OF THE SYSTEMS ENGINEERING APPROACH FOR A SMALL BUSINESS APPLICATION 62

4.1 Use of subsystem's three attributes (Availability, Dependability Capability) to evaluate effectiveness of any type of subsystem 62
4.1.1 Reason for use of subsystem's three attributes in any type of subsystem evaluation 63
4.1.2 Control Input Value as the limiting factor of optimisation of a subsystem's attributes 66

4.1.3 Proposal of the general structure of systems engineering evaluation in relation to small business development 67
4.2 General approach to systems analysis applicable to small developing business 69
4.2.1 Hierarchy of subsystems 69
4.2.2 Boundaries between subsystems - the method of finding the most optimal effectiveness value 70
4.2.3 Interrelation and connection between subsystems using the Capability attribute 72
4.3 Summary of the structure proposed for modeling a small developing business 74
4.3.1 General model of the small developing system 74
4.3.2 Considerations towards the computer model 77

5. GENERAL CHARACTERISTICS OF THE DEVELOPED SOFTWARE 79

5.1 The Excel part of the software 80
5.1.1 Main Subsystem (Main.xls file) 81
(a) Capabilities Section 81
(i) Controlling variables of Capability optimisation (Control Panel) 82
(ii) Inserting input values for Capabilities evaluation (Table of Main Subsystem's Performances) 83
(iii) Results of the dynamic programming (Critical Path) 84
(iv) Outputs of the Capability evaluation (Table of Capabilities of The Main Subsystem) 84
(b) Dependabilities Section 85
(i) Inserting input values (Table of Proposed Transition Rates) 85
(ii) Selection of the input values (Table of Selected Transition Rates) 86
(iii) Results of Monte Carlo Simulation (Average Values of Transition Rates) 87
(iv) Output and controlling variable of the Dependability evaluation (Table of Dependabilities) 88
(c) Availability Section 88
(i) Inserting the input values (Table of Proposed Inputs) 89
(ii) Selection of the inputs (Table of Proposed Inputs) 89
(iii) Outputs of the Fuzzy Logic (Table of Outputs and Availabilities) 90
(iv) The variables controlling the Availability optimisation (Table of Fuzzy rule & Table of the Magnitude of Inputs and Output) 90
(d) Final results 92
5.1.2. Secondary Subsystem 92
5.1.3. Final display of results 94
5.2 Delphi programme 94
5.2.1. Dynamic programming (Capability module) 95
5.2.2. Monte Carlo simulation (Dependability module) 96

5.2.3. Fuzzy logic (Availability module) 98
5.2.4. Co-ordination module 98

6. GENERAL CHARACTERISTICS OF INTEGRATION OF THE THEORY AND COMPUTER SOFTWARE INTO THE MODELING STRUCTURE 100

6.1. Recursive structure in a systems engineering model 100
6.1.1. Recursion level 1 101
6.1.2. Recursion level 2 101
6.1.3. Recursion level 3 103
6.2. Characteristic elements of the optimisation process 104
6.2.1. Control Input Value of Capability 104
6.2.2. Control Input Value of Dependability 106
6.2.3. Control Input Value of Availability 106
6.3. Evaluation of the subsystem's value (Sensitivity analysis) 107

7. APPLICATION OF THE THEORY AND THE MODEL IN A DETERGENT-MANUFACTURING COMPANY 110

7.1. Identification of the system 110
7.2 Relationship between the subsystems 111
7.3. Characteristics of the subsystem's elements 111
7.3.1. Technical Subsystem 112
(a) Capabilities 112
(b) Dependabilities 113
(c) Availabilities 115
(d) Remarks about the Technical Subsystem 117
7.3.2. Economic Subsystem 118
(a) Capabilities 118
(b) Dependabilities 119
(c) Availabilities 121
(d) Remarks about the Economic Subsystem 122
7.4. Results 123
7.4.1. Manufacturer's approach 123
(a) Results of the Technical Subsystem evaluation 123
(b) Results of the Economic Subsystem optimisation 124
7.4.2. M'Pherson's approach 126
(a) Results of the Technical Subsystem optimisation 126
(b) Results of the Economic Subsystem optimisation 128
7.4.3. The Author's Approach 129
7.5 Conclusions 133

8. REFLECTIONS **135**

8.1. Reflections on the computer model 135
8.1.1 The limitations of the size of dynamic programming 135
8.1.2 Limitations of Monte Carlo Simulation 136
8.1.3 Limitations of Fuzzy Logic 136
8.2. Reflections on the systems engineering model 136
8.3. Reflections on Application 137

9. REFERENCES **138**

APPENDICES

Appendix A Details of the computer software development I
Appendix B Manual for the Systems Optimiser software XX
Appendix C Listing of the computer programme XXXII
Appendix D Results of Present Value calculations CXXX
Appendix E Data used for the optimisation of the detergent-manufacturing system CXXXIII

LIST OF ILLUSTRATIONS

Figures

1.1 General comparison of complexity between the large and small developing organisations 4
1.2 Comparison of the analysis between big and small developing organisation 7

2.1 Perspective of the systems engineering analysis 14
2.2 Examples of performance evaluation 16
2.3 Relation of the reliability to state-space model 18
2.4 The illustration of the maintenance 20
2.5 Relationship between the reliability and supportability 22
2.6 Comparison of the scenarios between planned and unplanned repairs 22
2.7 Break-Even Analysis 25
2.8 General tree structure for technological system objective 29
2.9 Three state-space model 32
2.10 Graphical representation of Availability formula 34
2.11 Interrelation between subsystems attributes through subsystems elements 35
2.12 System Whole Life (Ownership) Costs 38
2.13 The balancing of cost and effectiveness 39
2.14 Quality space of effectiveness for two subsystems 41

3.1 General structure of the research approach 46
3.2 General explanation of the dynamic programming procedure 50
3.3 Single-dimensional model of performance 52
3.4 General structure of the single-dimensional dynamic programming process 53
3.5 Comparison between crisp and fuzzy sets 55
3.6 The Fuzzy Logic principle 56
3.7 General principle of Monte Carlo simulation 59
3.8 Allocated numbers to probability sector 59

4.1 The general structure of the small developing business evaluation 67
4.2 Selection of subsystem's boundaries 71
4.3 Interconnections between subsystems 72
4.4 General optimisation model for Main Subsystem in small developing business 75
4.5 General characteristics of the entire system's evaluation 77

5.1 General structure of the programme 79

5.2 Selection of transition rates through the selected performances values 87
5.3 Interpretation of the possible value of input or output 91
5.4 Relationship between the Main and the Secondary Subsystems' performance values (Capability section) 93

6.1 The general recursive level structure 101
6.2 Suggested connections between subsystems 102
6.3 Selection of the input values for the subsystem's elements 103
6.4 Influence of the magnitude of the Control Input Value on the size of dynamic programming 105
6.5 General structure of the Fuzzy Rule (Firing Rule) 107
6.6 Optimisation of the ratio of Effectiveness to Control Input Value Factor 108

7.1 State-space model for the Technical Subsystem 114
7.2 The relationships between the elements of the Technical Subsystem 117
7.3 State-space model of Economic Subsystem 119
7.4 Table of the Fuzzy Rule for economic Availabilities 122
7.5 Economic Subsystem 122
7.6 Relationship between the Technical and the Economic ratios of Effectiveness to Control Input Value Factor 130

Tables

2.1 Summary of Interest Formulas 24
2.2 Some useful signal to noise ratios 27

7.1 Table of proposed performances for the Technical Subsystem 112
7.2 Fuzzy rule for Availability of the Technical Subsystem 116
7.3 General characteristics of economic performances 118
7.4 Results of screening technical performances 124
7.5 Selected sets of economic performances related to selected optimal technical performances 125
7.6 Selection of the most optimal economic performances 125
7.7 Ratio of Effectiveness to Control Input Value Factor for the Technical Subsystem 127
7.8 Proposed economic performances for the R40000 investment from the Technical Subsystem 128
7.9 Selection of the most optimal economic performances 128
7.10 Ratio of the Effectiveness to Control Input Value Factor for Technical and Economic Subsystems 129
7.11 Percentage distance of the proposed options from the ideal option 132

In order to survive any impacts from the outside environment, small organisations must identify the areas/components which are crucial for their further development. Therefore, development of the small business expands in all possible directions of the organisation's activity (all components of the small developing business grow and develop nearly simultaneously). The development of the internal components of the developing business can be diversified in their nature (i.e. technical, economic, environmental) as well as in their structure (the mechanical subsystem, the process control subsystem, etc.). The same components would apply to the big business corporation but with greater complexity. Thus, it makes intuitive sense that the general problem-solving approach to the big and the small business corporation should be similar. This observation the author makes from his own experience gained during his work both in a leading petroleum company (big business corporation) and a detergent-manufacturing company (small business corporation).

The above statement is met with opposition in the environment of small developed businesses where the approach to design has a rather centralised and mechanistic character (i.e. first, develop a Main Subsystem and only then plan the growth of other subsystems in the business structure). This type of approach, even towards the small design, may cause negative results. For example, the efficient design of the simple mixing tank requires a balance between both the technical as well as the economical factors. In this case the manager of the small developing business would probably consider the economical factors alone. This is understood because a small business has often limited financial resources. However, the economical factors on their own will not secure the long term profitability.

Therefore all subsystems, irrespective of their importance, require a balanced consideration.

1.2 Small developing business corporation

It is difficult to visualise the general structure of the small developing business. The word *small* can be subjective and, often, this word does not fully describe the economic and the

scientific potential of the organisation. Contrary to *small*, the term *big developed business* does not imply that the business is completely developed and that its structure is not changing. The big business also has to develop due to constant environmental changes. However, the change in the big developed business is not so noticeable as in the small developing one (i.e. in the big business usually the structure or substructure of departments/components is changed, while in the small developing business the structure of the entire system may alter to adjust to environmental requirements).

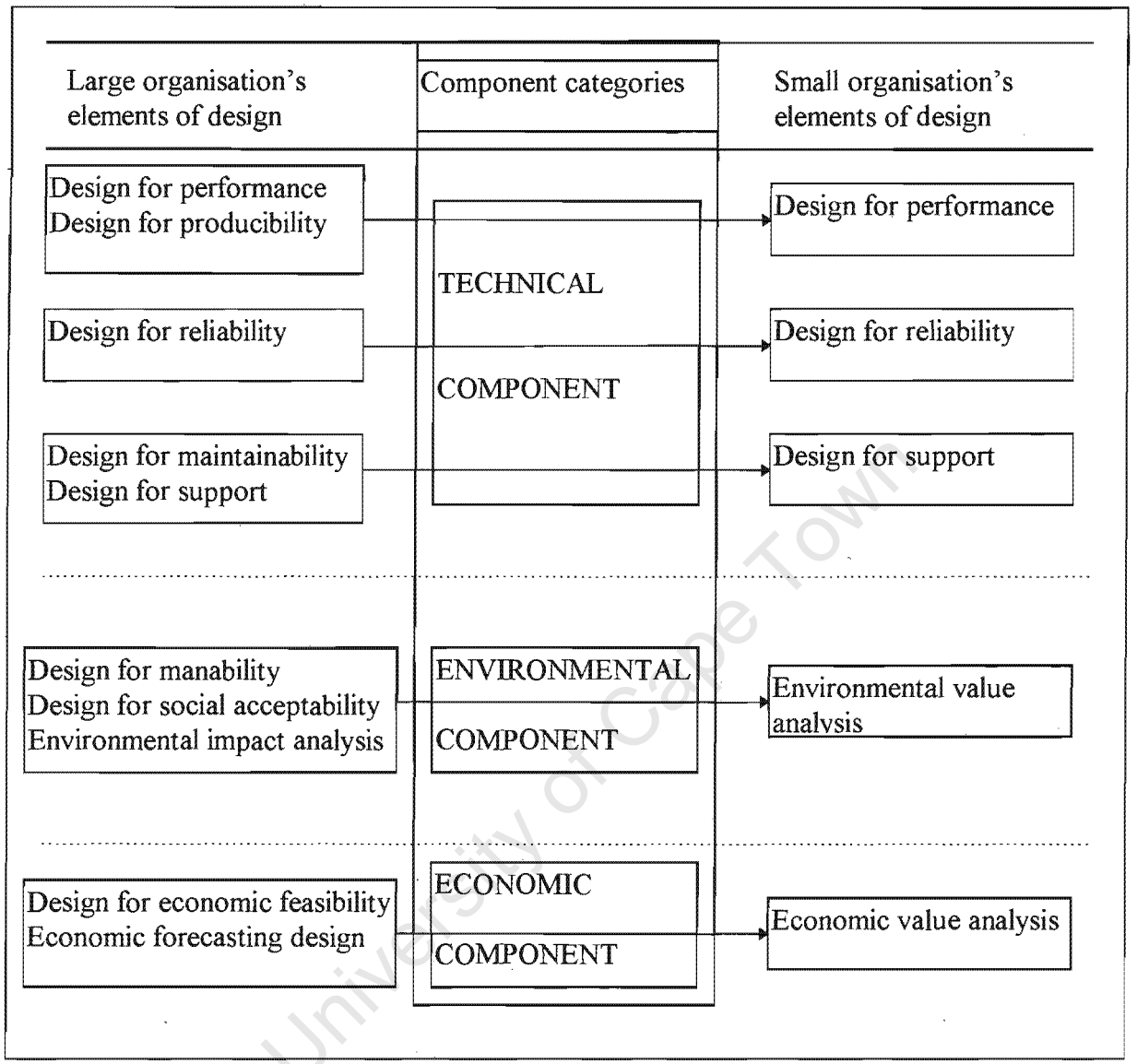
1.2.1 Complexity of the small business organisation

The author (during his work experience) has discovered that the small developing business may consist of just as many different types of components as the big business. The components of the small business are certainly less complex than those of the big corporation. For example, the economic department of the leading petroleum company (big organisation), in which the author was employed, consisted of accounting, planning and forecasting subdepartments, whereas in the detergent-manufacturing company (small developing organisation) all of these functions were the responsibilities of one person. Nevertheless, both types of the business organisations have three main categories of components (subsystems). These components (subsystems) are:

- technical component
- economic component
- environmental component.

The structure of the above components may differ between the large and the small organisation. Figure 1.1 below shows these differences.

Figure 1.1 General comparison of complexity between the large and small developing organisations



1.2.2 Recognising the purpose of modeling the small business and its components

The purpose of the small business (small system) is to be as profitable as possible which is achievable if the system will be effective in all areas of its activity. It means that all components of the system have to be effective. In order to achieve that effectiveness , the attributes of all components must be analysed.

In the case of the small developing business, the performance is regarded as the main element which alters the value of effectiveness. It may be true that performance is the most important but not the only element. It was said that during development of the business, the whole structure is continually reshaped, therefore the behaviour of the organisation and its components, as well as support requirements, must be altered. Therefore during development of the system's structure there must be some additional elements (characteristic for small developing organisation/system) which, along with the performance, would develop the components' effectiveness function. Still, when one component is developing, the other components may be altered. Thus all these components together can change the effectiveness of the entire system. For example, the introduction of the new furnace which processes chemicals more efficiently, may positively affect the budget and the environmental conditions of the chemical plant. Therefore the operational effectiveness of the plant can be increased

Thus the purpose of the organisation's modeling would be to optimise the elements of the organisation's components to such an extent that the effectiveness of the components and, in turn, the combined effectiveness (system's effectiveness is the result of combination of subsystems' effectivenesses) of the entire organisation would have the most optimal value.

1.2.3 General characteristics of the small developing business

The small developing business has a constantly changing structure. The pattern of change is often irregular and is thus difficult to predict. The reasons are as follows:

- Small businesses are not well established on the market (i.e. they compete with the bigger opponents for the market share, they enter the new market etc.)
- The managers, often the owners of the company, choose the short term operational survival (e.g. these managers will concentrate on the sales activity and ignore the production needs). This leads to frequent oscillations in the system's performance.
- Limited resources, in the small corporation, will often prevent the establishment of the reliable support design for the organisation's activities.

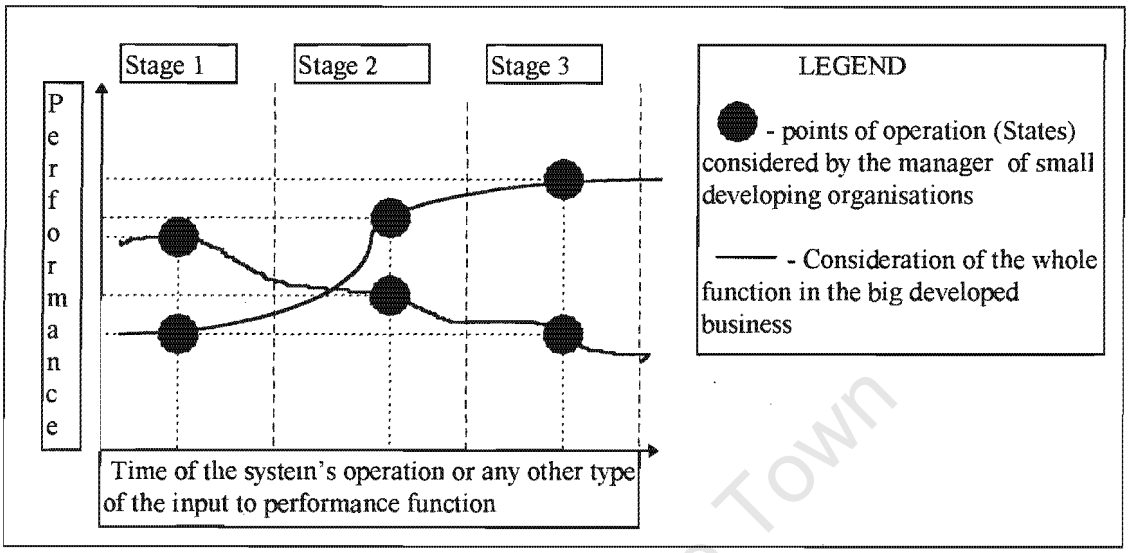
The above limitations in the small business corporation may lead to the following observations:

- The small business corporation may not always have a single mission to perform. Thus this corporation's main approach to problem-solving may vary. For example, during one particular month in the small detergent factory, the primary activity will be transport, while production may be a secondary issue. The next month the same company may primarily concentrate on production rather than transport. Although the one activity , such as transport, may be more important than the other (e.g. sales, production) all these activities have to be analysed. It is just the primary activity that governs the way by which the elements of the secondary activities are selected.

- The interrelationship between different elements in any single activity is often unpredictable. This means that the analytical methods (e.g. differential equations, statistics), used in the management of the small business corporation, may be unrealistic.

The management of the small business corporation will concentrate on the specific points of a system performance function. This function is constantly changing in small systems and, therefore, the analysis of the entire performance function is rather impossible. In a large business corporation the performance function is well established. Therefore, the manager of the large system is able to analyse all values of this performance function. Figure 1.2 shows the graph of the performance function and various elements which describe it.

Figure 1.2 Comparison of the analysis between big and small developing organisation



- The occurrence of events (e.g. failures, repairs) in the small system is random (i.e. demands for products are not established; system structure is not yet verified for potential environmental requirements, etc.). This means that the system's reliability (behavioural pattern) requires a probabilistic approach.

- The evaluation of support requirements of the small business structure (i.e. system's supply, labour, transportation, equipment requirements) is primarily based on the experience of its members who very often worked in organisations of a similar structure. Therefore, the small business will, most likely, subjectively evaluate its support requirements. This means that, in this case, the statistical tools would not be a proper choice.

1.3 Dealing with the complex structure

There are two main approaches which can be used to solve complex problems. The traditional analysis (functional analysis) breaks the problem into separate functions.

However, this approach would reduce the system to separate non-related elements leaving the meaning of the specific groups of elements (subsystems) unrecognised. On the other hand, systems engineering would only reduce the system to the group of elements which together serve a specific purpose (subsystem's purpose). It means that the entire system is divided into subsystems and subsystem's elements whose relationships are known within their boundaries of activity.

The small developing business/system is the structure of components where understanding of components' purposes is crucial for the development of the entire organisation/system. Therefore, systems engineering is, most likely, a suitable approach to model a small developing business.

1.3.1 General systems analysis

The systems approach defines the system as the part of the environment with which that system interacts. Every system is an assemblage or combination of elements which, in spite of their different meaning, have one common objective - system's objective.

B.S. Blanchard & W. Fabrycky[2] defines a system in the following way:
" *Systems are composed of components, attributes and relationships. These are described as follows*:

1. *Components are the operating parts of a system consisting of input, process and output. Each system component may assume a variety of values to describe a system state as set by control action and one or more restrictions*
2. *Attributes are the properties or discernible manifestations of the components of a system. These attributes characterise the system*
3. *Relationships are the links between components and attributes"*

(Note that *attributes* used in the above definition are the functions of a system's input elements).

Since some components of the system may have properties of the system itself, the components are often called *subsystems*. For example, the technical, economic and environmental components are all subsystems of the business structure. The subsystems may themselves contain components, which are called subsystem's elements. The subsystem's elements, the same as the system's subsystems may have different meaning or serve different purposes but they are part of the same business structure. Again B.S. Blanchard & W. Fabrycky[2] defines the system's components :

" *A system is a set of interrelated components working together towards some common objective. The set of components has the following properties:*

1. *The properties and behaviour of the set has an effect on the properties and behaviour of the set as a whole.*
2. *The properties and behaviour of each component of the set depends upon the properties and behaviour of at least one other component in the set*
3. *Each possible subset of components has two properties listed above; the components cannot be divided into independent subsets"*

A system or subsystem can have a distinguishable meaning if it has a boundary. The boundary of a system/subsystem may be physically seen (e.g. mechanical and electrical system of a car) or may have to be drawn arbitrarily by the designer (i.e. distinguishing between the production and quality systems in a factory).

As it was said earlier, systems consist of components. The interaction of components with one another makes a system an operational structure. The system's components (subsystems, subsystems' elements) interact with each other by means of inputs and outputs. The input represents the influence of the environment on the system/subsystem/component. The output is the influence of system/subsystem/component on the environment. By means of inputs and outputs the systems/subsystems/components can be classified according to the hierarchy of importance to the entire system.

1.3.2. Purpose of systems engineering applicable to small business

As it was mentioned earlier, every element of a system/subsystem would have a specific purpose. M'Pherson[20] implies that the effective system would have to

- meet its operational requirements,
- be supported by the available resources,
- have good behavioural pattern.

Therefore, three major types of elements should be sufficient to find the general value of the system's/subsystem's effectiveness. However, the system's/subsystem's elements as well as the systems/subsystems themselves may have different or even opposite objectives.

Therefore the approach will not only have to identify various unrelated subsystems (technical, economic, environmental) and their internal structures, but it would also have to integrate and balance the conflicting objectives of these subsystems in the one criterion requirement.

1.3.3 Nature of a system's models

Depending on how the system's elements are interrelated or dependent on each other so various types of a system's models may be identified.

Traditionally, mechanistic and organismic systems were developed. However, due to accelerating changes, increasing uncertainty and growing complexity a third system, the social system model, was introduced (R.L. Ackoff[22]).

The mechanistic system conceptualises the system as the machine that works with regularity dictated by its internal structure and the casual laws of nature. This model has two fundamental assumptions:

- system is completely understood

- understanding of the system is obtained by analysis

The elements of this type of a system are directly dependent on the main element in the hierarchy and they have no purpose other than to serve the main element.

The organismic system conceptualises the system as an organism. It has a purpose of its own, namely the survival for which growth is taken as the essential element. However, parts of the system have no purpose of their own. The environment of this system is regarded as a purposeless and passive provider of necessary inputs and outputs.

However the above models may not be applicable for the small developing business because they deny the development of all elements of a system.

The social system model relies on development and tries to serve the purposes of the system and its components. There can be conflict between the subsystems or system's elements. Therefore, the major aim of this system will be to balance these conflicts. In this system, a conflict would not be regarded as a negative symptom but rather as the stimulus to the development in every activity of the system. Therefore, the social model is probably the proper choice for the design of the small developing system.

1.4 Identification of the problem statement

The small developing business has been identified as the complex structure which, due to its complexity, should be analysed using the systems engineering approach. However, the developing character of the small business requires an insight into non-analytical methods of optimisation.

No specific references/publications were found about the structure of the developing business. Therefore, the findings are based primarily on the author's work experience.

The above discussion led the author to the formulation of two objectives:

1. Attempt to build the general structure for the Systems Engineering model applicable to the small developing business
2. Check the model applicability towards the small developing system.

2. SYSTEMS ENGINEERING THEORY IN THE SMALL BUSINESS CORPORATION

The analysis of any system (big or small) is usually not so simple because a system is a structure of many combinations of interconnected components. In the case of the small business, the lack of financial resources and the proper expertise, makes the process of identifying the system structure often impossible. The theory of systems engineering presented in literature is complex and very often this theory is impossible to apply without professional expertise.

This section refers to those aspects of the systems engineering theory that could be suitable to the analysis of a general or small-size developing system.

2.1 General description of a system's structure

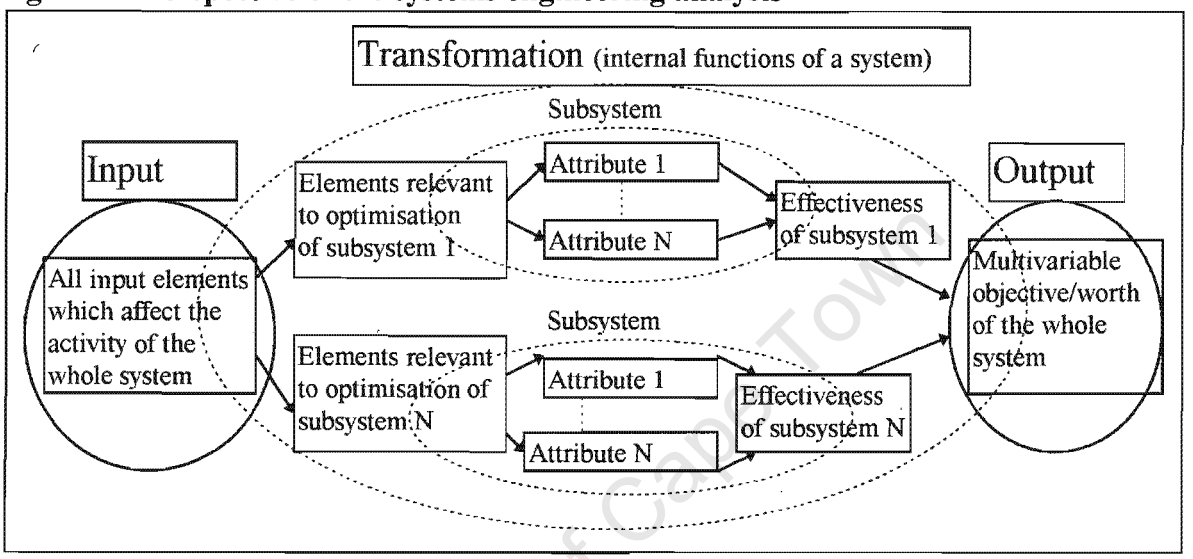
It has already been mentioned that a small developing business is regarded as a system with a single primary objective and many secondary activities (see section 1.2.3). The analysis of all activities (both the primary and the secondary ones) is necessary because the primary activity (main objective) is often interrelated with secondary objectives. For example, the stainless steel reactor will produce a very wide range of chemical products. The plastic reactor, though less expensive than the stainless steel one, can process a very limited number of chemicals. Thus, the primary objective in this example is the production which is interrelated with the secondary activity - the economical analysis.

These activities, otherwise known as components or subsystems (see section 1.3.1), contain one or more characteristic attributes (see section 1.3.1). These attributes convert the quantitative meaning of subsystem's elements into a qualitative representation of these elements. Then the values of these attributes combine into a single value of subsystem's effectiveness. All the values of the subsystem's effectiveness (otherwise known as

objectives) combine into a single multivariable objective. The multivariable objective is also known as the total value of the entire system.

The general perspective of the above systems engineering approach is shown in the figure 2.1 below:

Figure 2.1 Perspective of the systems engineering analysis



2.2 Inputs of the System

The developing business needs information and support from diversified branches of science because it grows simultaneously in different, often not related, areas of knowledge. Therefore, the inputs of the system are the results of the technical design, the economical evaluation and the environmental observations.

The meaning of input elements, used in the systems engineering analysis, may differ from the meaning of input elements which exist in general management or in engineering. For example, performance is identified as the output in the managerial evaluations. However, in the systems engineering optimisation, the value of performance is defined as an input element to the subsystem's effectiveness. The reason for this discrepancy in the meaning

of input elements is that the systems engineering optimises (or selects) rather than designs subsystem's/system's structures.

2.2.1. Input elements of the Technical Subsystem applicable to the small business corporation

Most of the small organisations evaluate Technical Subsystem effectiveness using performance as the only input element. The performance is easy to measure and to compare quantitatively. However, the performance alone is an inadequate measure of the effectiveness of the entire subsystem. Other elements (such as the operational readiness, etc.) must also contribute to the subsystem evaluation. B.S. Blanchard & W. Fabrycky[2] name these elements as reliability, supportability, maintainability, manability, producibility, economic feasibility, social acceptability. M'Pherson[20] reduces the number of B.S. Blanchard & W. Fabrycky[2] elements to the first four or even three.

In general, a small developing business having a simple structure or lacking in financial resources, classifies manability as the subelement of supportability design, producibility as the subelement of performance design, economic feasibility as a part of Economic Subsystem evaluation, and social acceptability as a characteristic element of the Environmental Subsystem. It is very often seen that even maintenance is classified as part of supportability design. Therefore, the Technical Subsystem's optimisation of the small-sized business corporation will, most likely, contain three input elements. These are:

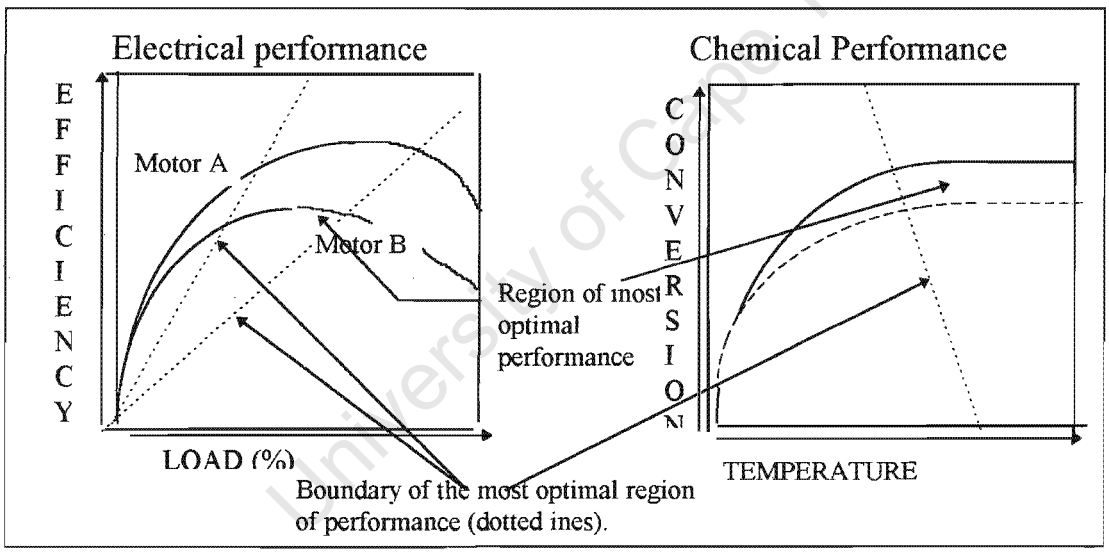
a) performance, which characterises the physical Capability of achieving a system's goals (*able to meet performance requirements at any operational moment)*
b) reliability, which evaluates a structure of a system's coordination to complete the subsystem's mission .
c) support and maintenance design, which supports and maintains the system's mission at an acceptable level during all stages of the system's operation (*operationally ready at any time)*

(a) Definition and measure of technical performance

Performance can be defined as the physical characteristics which the operating system must exhibit to accomplish its planned mission. The design of performance is dependent on the technical performance evaluation. This evaluation is the result of the engineering design rather than the result of the systems engineering analysis.

A variety of models is considered where performance is evaluated against technical specifications. However, no general evaluation can be given because every type of the Technical Subsystem (e.g. chemical, mechanical, biological etc.) has its own methods of optimisation. The following examples of performance evaluation are shown below:

Figure 2.2 Examples of performance evaluation



The above two examples show dependent variables (efficiency and conversion) in the form of ratios or non-dimensional figures of merit. This way of expressing performance can help to analyse possible trends in input and output variables.

The other aspect of the performance evaluation is its complexity. The electrical performance example (figure 2.2) clearly shows the region of most optimal performance. In the chemical performance example (figure 2.2), one can see the region where performance becomes optimal, however the end point of this performance is not known.

Therefore, the systems engineer must also check if one performance value is not interrelated with others. In the chemical performance case, the other related value may be material performance (e.g. high values of conversion require higher temperatures and therefore more thermally resistant materials).

As it was said earlier, only the best and most promising technical/operational data are collected and compared with the ideal scenario. This comparison builds the functional model of a system and therefore evaluates its Capabilities at different states.

The values of the performance of the small business corporation are discretely interrelated with each other. This is because the function of the performance in a small-sized system is constantly changing. Therefore, values of performances are evaluated and collected only at major points of the system's operation (operational stages). The only independent variable which links them is the magnitude of the resource allocated to every discrete performance.

The required performances of any stage may deteriorate as the time of operation increases. Therefore, it is important to know the reliability of the desired performances.

(b) Definition and measure of technical reliability

Reliability is a probability of successful performance (of product or system) for a given period of time under specified operational conditions. Therefore, its input value is the time, and output is a probability of successful completion of a mission.

The successful completion of the mission will be dependent on the transition rate (i.e. rate of failure and repairs) from one state of operation to another. The occurrence of the transitions is not well defined in the small developing organisations and, therefore, it is regarded as random. Since the values of the performances are only known at the main discrete points of operation, the respective values of transitions will have to be collected at those particular points (stages of system's operation). The transition rate (e.g. failure rate)

along with the time in which the transition occurs is usually evaluated in the form of a probability function.

There are various functions of probabilities which can be used to define reliability. These functions may have the form of a binomial, exponential, normal, Poisson, gamma, Weibull or other distributions. Reliability is usually expressed in the form of an exponential function (B.S. Blanchard & W. Fabrycky[2]).

$$R(t) = \exp(\lambda * t)$$

λ - failure rate λ = number of failures/total operating hours

t - time period of interest

One should notice that the failure rate may be also defined in terms of time needed for the replacement or reloading. On the other hand, it could be used to measure the possibility of a defect. Therefore, the reliability evaluates the transference of the subsystem from one operational stage to another. The graphical representation of reliability is shown in figure 2.3 below:

Figure 2.3 Relation of the reliability to state space model

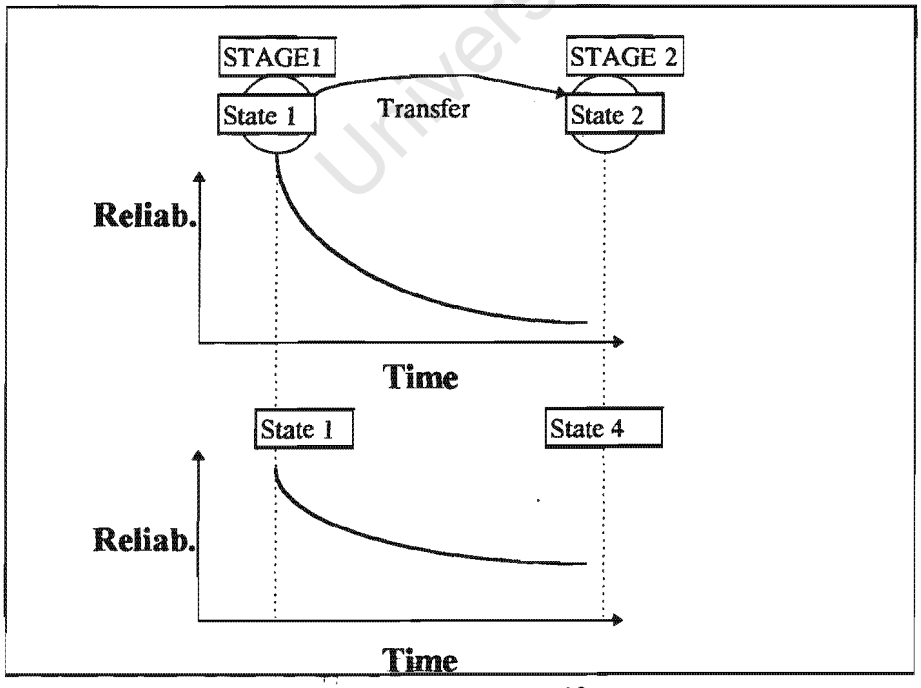
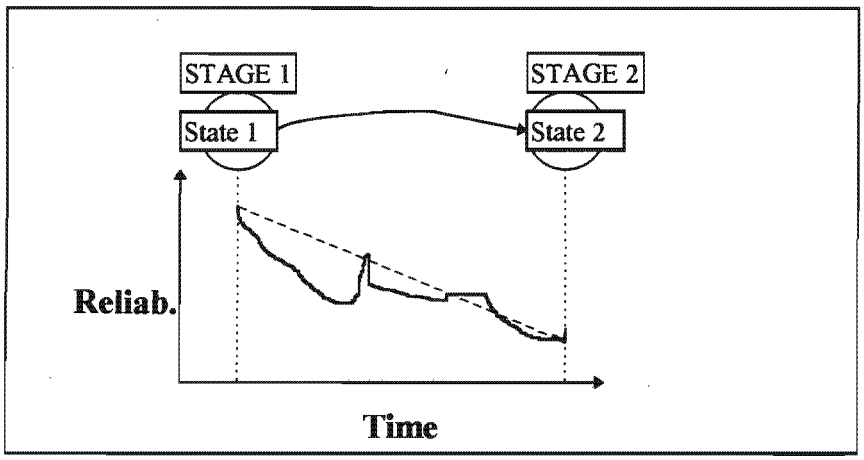


Figure 2.3 Relation of the reliability to state space model (continued)



The first two graphs of figure 2.3 show the exponential nature of reliability. Here, the subsystem is transferred from stage 1 to stage 2. The transfer from stage 1 to stage 4 is more reliable than the transfer from stage 1 to stage 2 over the same period of time. The third graph of figure 2.3 represents a more realistic trend in reliability characteristic of small-sized systems. Here, the pattern of reliability is not linear due to the random occurrence of transition rates (failure or repair rates).

The reliability measures the probability of the subsystem falling from a higher level of performance to a lower level or vice versa. In order to keep a system in operation, one must have support which can 'push' the system to the previous or new level of performance. In this case, the support design and maintenance must be prepared.

(c) Definition and measure of technical maintainability

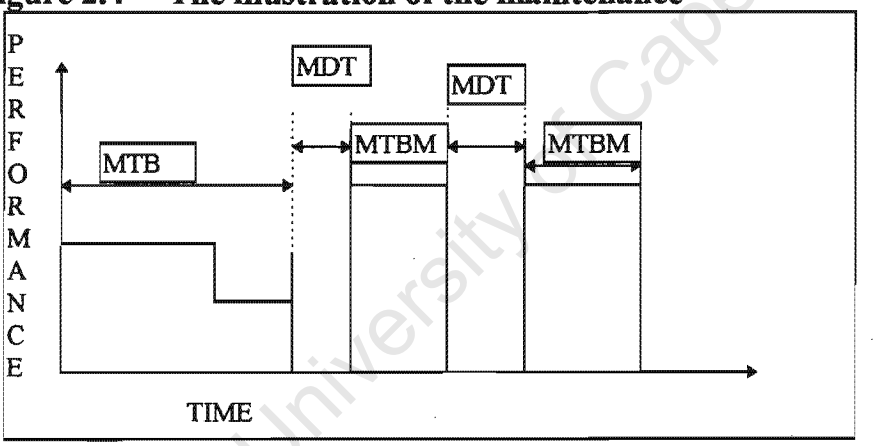
Maintainability can be defined as the ability to keep the system in constant operation. B.S. Blanchard & W. Fabrycky[2] present a very detailed analysis on the subject of maintainability. These researchers concentrate only on the operational maintainability. This is characterised by time intervals during which a system is active (this includes all levels of performance) and the time when the system is idle. Therefore, one may consider *mean time between maintenance* and *maintenance downtime*.

Maintenance downtime (MDT) is the total elapsed time required to repair and restore a system to its full operating status. It may contain such elements as time delay needed for loading or reloading batch process equipment, time delay concerning administrative work or waiting for the spare parts. Maintenance downtime may be modeled by finding the mean of the downtime data. The downtime data can be used to find its distribution. The most common distributions are normal, exponential and log normal (B.S. Blanchard & W. Fabrycky[2]).

Mean time between maintenance (MTBM) is the time when the system is at work. Therefore, time of performance may relate to this parameter.

The illustration of the above terms of maintenance is given in figure 2.4 below.

Figure 2.4 The illustration of the maintenance



As the life cycle increases so the maintainability of specific states will become less predictable. In this case, it is important to find a measure which can support the maintenance plan for the longest period of time. In other words, the resource requirements for the specific behaviour of the system must be supported by the adequate plan of support design (e.g. supply of needed spare parts, quick delivery).

(d) Definition and measure of technical supportability design

This is defined as the effective and economic support of a system throughout its programmed life cycle. It may contain such aspects as maintenance plan, supply and support plan, test and support equipment plan, personnel and training plan, facilities plan, data plan, computer resources plan, retirement plan. Theory of the supportability is explained in detail by B.S. Blanchard & W. Fabrycky[2] . One may only say that it is difficult to find the general measure of supportability.

In the situation of the small business environment, supportability is usually evaluated in the subjective way especially in those cases where its structure can not be modeled mathematically. Generally, the supportability may have some input values (e.g. need of highly specialised labour, spare parts) which, in some subjective way of evaluation, would result in the qualitative output (probability of smooth operation of the working line, probability of long undisturbed functioning of equipment). Often, the relation between the inputs and output will not be scientifically understood. This is the case in the small business environment which almost exclusively uses a subjective approach.

M'Pherson[20] simplifies the function of the support design by showing an exponential nature of repairs (the same as failure). Also a typical example of supportability is given by B.S. Blanchard & W. Fabrycky[2] for inventory modeling. Usually, considerations of inventory may describe a plan where a specific amount of critical stock (spare parts, resources) are required for the operational demands. The aim of supportability is to find such an amount of stock that should prevent future operational disruptions but at the same time be the most economical.

There is a close link between the reliability and supportability in the way that support design would be influenced by the reliability requirements. The relation is shown in figure 2.5 below:

Figure 2.5 Relationship between the reliability and supportability

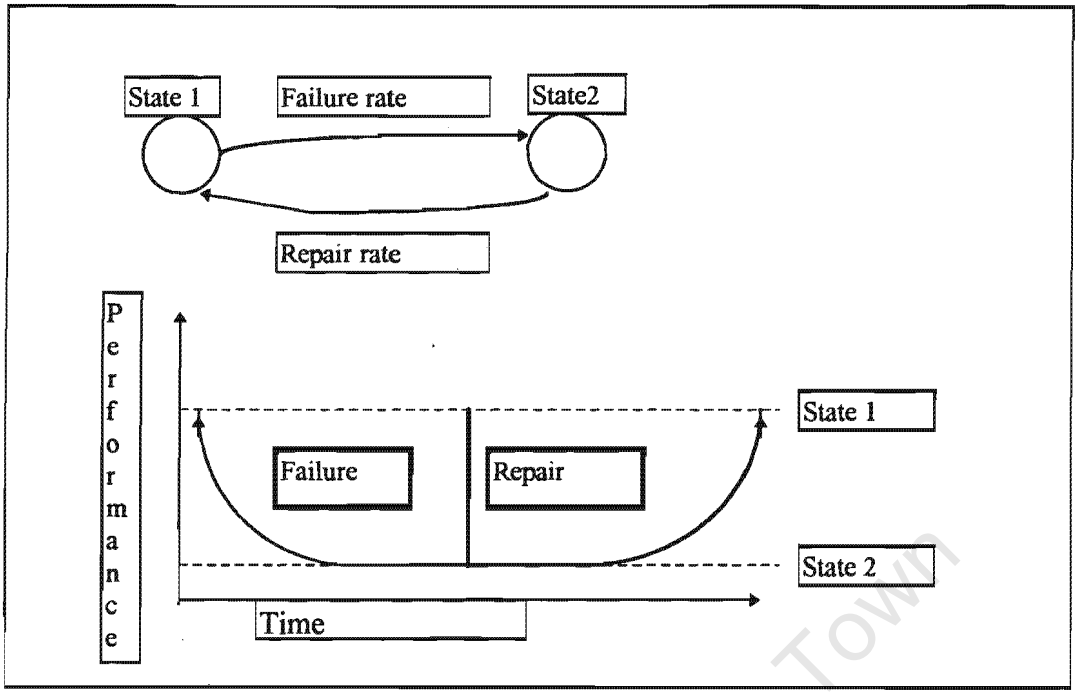
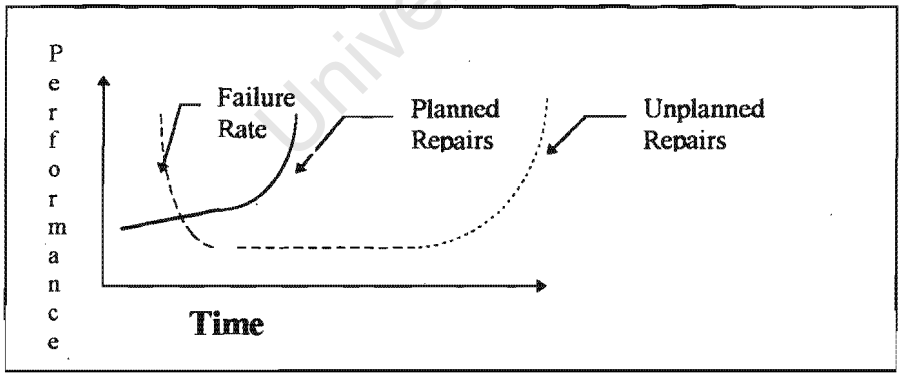


Figure 2.5 shows the situation where the repair rate is greater than the failure rate. At the same time, repair should overlap with failure in order to decrease the Availability of state 2. This is analogous to the amount of stock in the example presented by B.S. Blanchard & W. Fabrycky[2] . Figure 2.6 shows an optimal situation.

Figure 2.6 Comparison of the scenarios between planned and unplanned repairs



B.S. Blanchard & W. Fabrycky[2] noticed that mathematical models used to study systems and their elements are different from those used in the physical sciences. The different nature of the subsystem's input elements may create difficulties in evaluating the effectiveness of the Technical Subsystem. The subjective values of support design,

probabilistic outcomes of reliability evaluation and functional quantitative values of performance are difficult to combine.

Therefore, B.S. Blanchard & W. Fabrycky[2] as well as M'Pherson[20 21] mentioned three attributes which combine the above subsystem's input elements to form the single output value (effectiveness).

M'Pherson[20 21] describes these attributes and the effectiveness value more precisely and, therefore, the analysis will be based on his papers.

2.2.2 Economic Subsystem's input element (economic performance)

B.S. Blanchard & W. Fabrycky[2] and Dandy & R.F. Warner[9] give an impression that the economic analysis is based on the evaluation of the monetary values of benefits (e.g. profit increments, cost decrements). The values of the economic benefits may be defined to be the economic performances because they quantitatively indicate the accomplishment of the subsystem's mission (as in the case of the Technical Subsystem).

B.S. Blanchard & W. Fabrycky[2] and M'Pherson[20,21] do not mention the elements which could describe the behaviour and the support of the Economic Subsystem. Therefore, the economic performance is most likely the only input element which optimises the Economic Subsystem. This means that the value of the economic performance would be directly transferred into a qualitative value of the subsystem's effectiveness.

In contrast to the technical performance, the general methods for the evaluation of the economic performance value, are already developed. It is difficult to say which of these methods will apply to the small developing business. Often the nature of the economic scenario will select the best method. From the author's experience, the methods most frequently used are:

- Interest and interest formulations

- Break - even economic evaluation

(a) Interest and Interest formulas used to evaluate economic performance

The term interest (i.e. interest, dividends, rent, commercial profits) denotes the money earned by the original sum of money in a specified period of time. The rate, at which the original sum of money earns the interest, is known as the interest rate. The interest rate is expressed as the ratio of the interest earned during a specified period of time to the original sum of money from which this interest is earned.

According to B.S. Blanchard & W. Fabrycky[2] interest formulas have five common parameters:

i - nominal annual rate of interest

n - number of interest periods

P - principal amount at the time assumed to be the present

A - single payment in series of n equal payments, made at the end of each interest period

F - amount, n interest periods hence, equal to the compound amount of a principal sum, p, or the compound amounts of the payments, a, at the interest rate i.

These formulas are summarised in the table below:

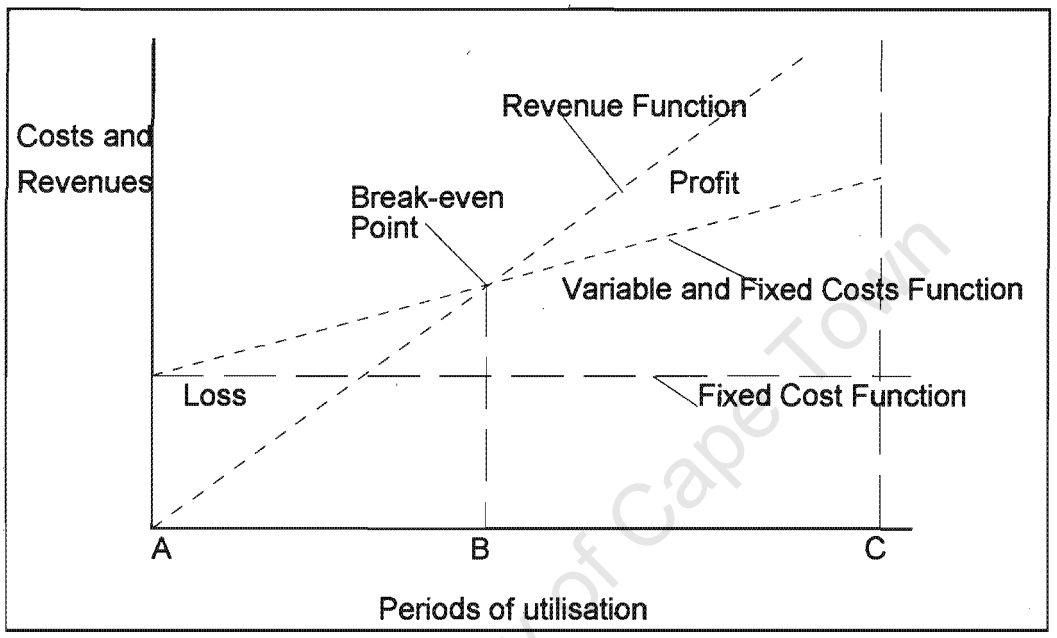
Table 2.1 Summary of interest formulas

FORMULA NAME	FUNCTION	FORMULA	DESIGNATION
Single-payment compound amount	Given P Find F	$F = P(1+i)^n$	$F = P(^{F/P,i,n})$
Single-payment present worth	Given F Find P	$P = F\left[\frac{1}{(1+i)^n}\right]$	$P = F(^{P/F,i,n})$
Equal-payment series compound amount	Given A Find F	$F = A\left[\frac{(1+i)^n - 1}{i}\right]$	$F = A(^{F/A,i,n})$
Equal-payment series sinking fund	Given F Find A	$A = F\left[\frac{i}{(1+i)^n - 1}\right]$	$A = F(^{A/F,i,n})$
Equal-payment series present worth	Given A Find P	$P = A\left[\frac{(1+i)^n - 1}{i(1+i)^n}\right]$	$P = A(^{P/A,i,n})$
Equal-payment series capital recovery	Given P Find A	$A = P\left[\frac{i(1+i)^n}{(1+i)^n - 1}\right]$	$A = P(^{A/P,i,n})$

(b) Break - even economic evaluation

Break - even analysis identifies the range of economic variables (e.g. costs, revenues etc.) within which the most desirable economic outcomes may occur. The General break-even evaluation is shown in figure 2.7.

Figure 2.7 Break - even analysis



In the figure 2.7 one can see that the break - even point (i.e. point B) divides the periods of utilisation where the combined values of Revenues and Costs (i.e. economic performances) result in either the Profit or the Loss. Therefore, the designer has a clear indication which sets of economic values can be used for further optimisation of the subsystem.

The economic performance, observed in the small developing business, will have the same discrete nature as the technical performance (see subsection 2.2.1.a).

2.2.3. Environmental Subsystem's input element (environmental performance)

Input elements to the Environmental Subsystem optimisation are the measures of environmental changes. These environmental changes may be defined as environmental performance (e.g. pollution, climatic changes etc.). Since the reviewed literature (B.S. Blanchard & W. Fabrycky[2], G.C. Dandy & R.F. Warner[9], M'Pherson[20,21]) does not indicate input elements other than the environmental performance, the environmental performance is going to be the only input element in the Environmental Subsystem evaluation.

The environmental performance may be divided into two categories:

a) Impacts of the environment on the system
b) Impacts of the system on the external environment

(a) Impacts of the environment on the system

The surrounding environment may have a positive or negative impact on the system development. For example, access to raw materials will improve the system's development. On the other hand, bad soil for agricultural cultivation, lack of qualified labour in industry may inhibit any of the system's activity.

The Taguchi method (A.Bendel[13]) may be an appropriate approach to the environmental analysis. This method defines controllable and uncontrollable variables of the environment and shows how they interact with one another.

The controllable variables are those variables which can be changed by the designer (e.g. thickness of material). On the other hand, the uncontrollable variables may only be altered by the changes of outside environment. The objective of the method is to predict the interaction between controllable and uncontrollable variables, so the uncontrollable

variables become the controllable ones. In this case, Taguchi introduced a signal to noise ratio which measures the environmental performance.

The formulas of S/N ratios are given in the table below:

Table 2.2 Some useful signal to noise ratios

Type N: nominal is best (dimensions, output voltage, etc.)

$S/N_N = 10\,Log_{10}\dfrac{(S_m - V_e)/n}{V_e}$ $\quad S_m$ - mean $\quad V_e$ - variance

where: y_i is an observation and n is the number of observations

$S_m = \dfrac{(\Sigma y_i)^2}{n}$ $\quad V_e = \dfrac{\Sigma y_i^2 - (\Sigma y_i)^2/n}{n-1}$

Type S: smaller is better (noise, harmful material, contamination, etc.)

$S/N_S = -10\log_{10}\left(\frac{1}{n}(\Sigma y_i^2)\right)$

Type B: bigger is better (strength, power, etc.)

$S/N_B = -10\log_{10}\left(\frac{1}{n}(\Sigma 1/y_i^2)\right)$

In the S/N analysis, controllable values represent signals while the uncontrollable values represent noise. Of course, Taguchi's method will be quite a costly method to use. A large amount of information is required to generate values of signal to noise ratios (i.e. many experiments are necessary). Therefore, the small developing business will probably subjectively evaluate the Environmental Subsystem. Here, the workers' experience will play a crucial role.

(b) Impacts of the system's environment on the external environment

The system may influence the surrounding environment. The operational requirements of this system may have a positive or negative effect on the environment. The cultivation of deserts is a result of a positive impact of the system on the environment, while pollution is an example of negative effects.

The Taguchi method described above may also evaluate this environmental performance.

The environmental performance, observed in the small developing business, will have the same discrete nature as the technical performance (see subsection 2.2.1.a).

2.3 Characteristics of the system's transformation function (subsystem)

The elements of the system's inputs come from different areas. Thus, the functions characteristic of these inputs will optimise these inputs elements. These optimising functions are the subsystems of the entire system. According to M'Pherson[20], every system's structure can contain four types of basic assessments (the author of this thesis regards these assessments as the subsystems optimality) which are necessary for the system to survive. M'Pherson[20] identifies these assessments as follows:

- Life-cycle
- Technical
- Economic
- Environmental

The Life-cycle optimality refers to the evaluation of the system over its entire life. It starts with the identification of need and impacts exerted on outside environments. The life-cycle goes through such stages as planning, research, design, production or construction, evaluation, consumer use, field support, ultimate product phaseout.

Technical optimality considers technological effectiveness/performance (physical design) and non-commercial cost (i.e. realisation, operation, support). Its focus is directed on the techniques which build an 'immune' subsystem (system whose performance is minimally affected by external factors). It means that the subsystem produces a high quality product with the lowest cost possible. The technical optimality deals with cost-effectiveness analysis where the effectiveness is a qualitative evaluation of the entire operational activity of the subsystem.

Economic optimality evaluates the possibility of obtaining the highest profit. The highest profit is estimated after evaluation of several alternatives. Therefore, the economic

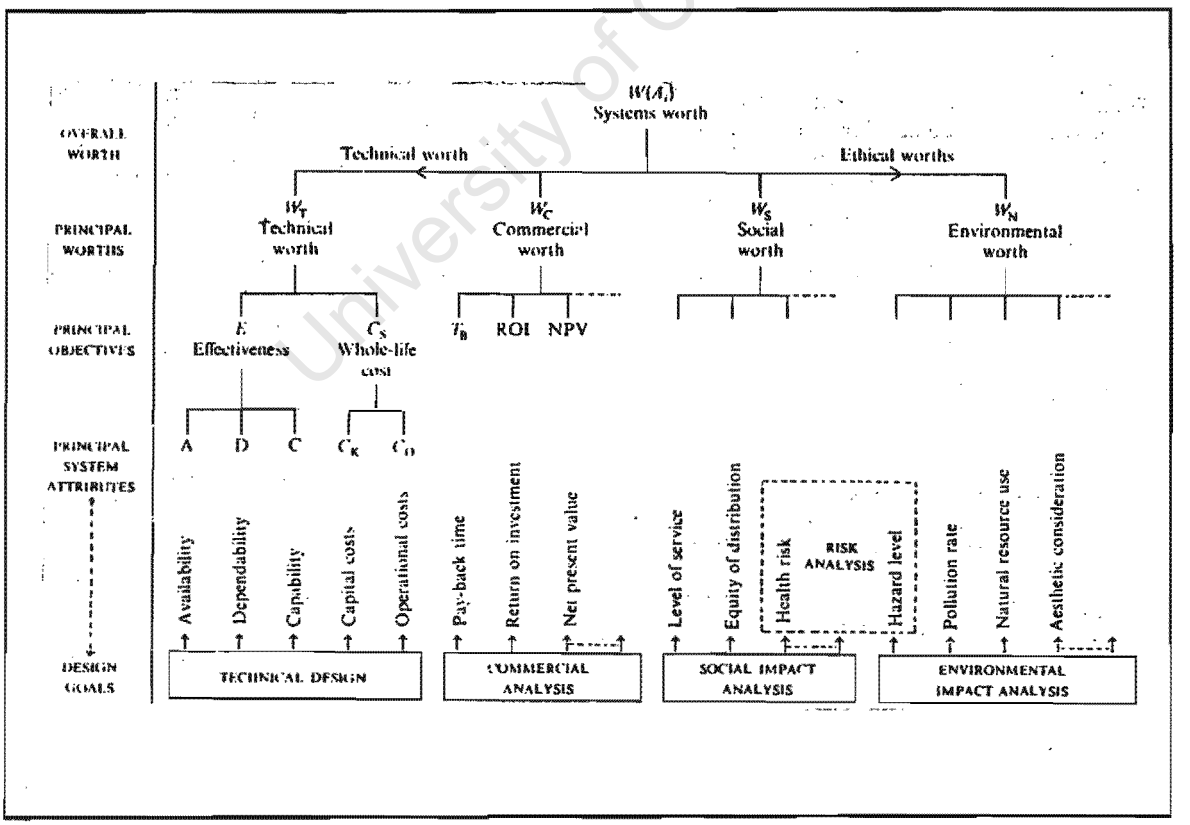
optimality considers a return-on-investment, a break-even point and other time dependent analyses.

Environmental optimality is an assessment of the system's impacts on the social and natural environment. It may also evaluate environmental conditions to which the system is subjected. The optimality can be regarded as evaluation of the most critical values that characterise the extreme points of the system's environment.

The above optimalities presented the diversified nature of subsystems which complicate the understanding of the system's functions. Therefore, the subsystem's attributes and subsystems' input elements depend on their subsystem's nature.

M'Pherson[20] shows the different natures of the principal subsystem's attributes (see figure 2.8).

Figure 2.8 General tree structure for technological system objective



2.3.1 Technical Subsystem's transformation functions (Technical Subsystems attributes)

The role of the Technical Subsystem is to convert the technical data into qualitative values characteristic of the attributes of the subsystem. The combined qualitative values yield an effectiveness value which must be balanced with the resource requirement (the cost in figure 2.8).

According to M'Pherson[21], the technical effectiveness combines three attributes

- Capability
- Availability
- Dependability

These attributes characterise the groups of the subsystem's elements which have unique tasks to accomplish. The general characteristics of these attributes are given below:

(a) Technical Capability attribute

High Capability describes the qualitative value of the subsystem's ability to meet operational requirements at any environmental conditions. The Capability is calculated from the subsystem's operational/functional outputs. This Capability optimises things such as the accuracy, flow rate, power output etc. Therefore, Capability is a function of performance. The performance is assessed for each stage of a system's operation and is then compared to the ideal value. Capabilities can be evaluated in the form of ratios showing how far the selected value deviates from the ideal one. This assessment is expressed in the following way :

$$C = [c_1, c_2, \ldots . c_r \ldots c_n] \qquad 0 \leq c_r \leq 1$$

Capabilities (or probabilities of states performances) are usually compared with other systems' Capabilities in order to make a preliminary choice of the system.

M'Pherson[20] also claims that reliability design influences the Capability's value. It is obvious that in a real life situation, the Capability function will be sensitive to performance degradation (unreliability) at different stages of subsystem's operation. In other words, as the reliability decreases so the chance of performance decreasing becomes greater. For example the more technologically advanced machine will not perform well unless the parts of this machine are reliable.

Since the reliability influences the value of the performance, the different levels of the subsystem's performance may be present at different stages of the subsystem's operation. However, as the system will always seek the ideal performance, many transitions from lower to higher levels/stages of performance may occur. The transitions, which are the effects of the reliability design, are described by the Dependability attribute.

(b) Technical Dependability attribute

High Dependability represents the probability that a system will be progressing successfully from one operational stage to another. It is also called a transition state function which shows an entire subsystem's progress in transferring a system's input values into its respective outputs. It is quantified in terms of failure and repairs and it is interrelated with the Capability and Availability. It is primarily a function of reliability, which is influenced by the support design (the same as the performance is influenced by the reliability).

Dependability may be represented in the form of a matrix :

$$\begin{array}{l} \\ D = \end{array} \begin{array}{c} \downarrow \text{Stages (n-1)} \\ \begin{bmatrix} d_{11} & d_{12} & d_{1R} \\ d_{12} & d_{22} & d_{2R} \\ d_{R1} & d_{2R} & d_{RR} \end{bmatrix} \\ \text{Stages(n)} \rightarrow \end{array} \qquad \sum_j d_{ij} = 1$$

where d_{11} is the probability of being in stage 1 without making a transition to any other stage

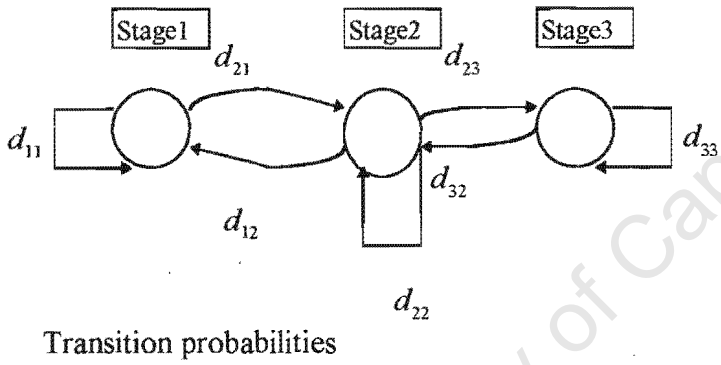
d_{12} is the probability of the transition from stage 1 to stage 2, etc.

R represents the numbers of subsequent stages

According to M'Pherson[20], R. Howard[17], the transition probabilities of the above matrix are expressed in terms of exponential rates of repairs and failures.

For example, a three state system could be expressed in the following way:

Figure 2.9 Three state space model



Transition probabilities

$d_{11} = 1\text{-}\exp(\lambda_a \cdot \Delta t)$

$d_{21} = \exp(\lambda_a \cdot \Delta t)$

$d_{12} = \exp(\mu \cdot \Delta t)$

Where

λ_1 – failure rate of stage 1

μ_2 - repair rate of stage 2

Δt - time interval

The time interval represents the time of the entire operation where all stages are likely to occur. The theory behind the transition probabilities is well explained by the Markov process (R.A Howard[17], D.J Sherwin[11]). These transitions (failure and repair rates) can be sorted and modeled in terms of the probability density function.

As it was mentioned earlier, the support design facilitates the level of reliability. If these requirements can not be fulfilled then the subsystem will not be ready for the operational use. This leads us to the next attribute - the Availability.

(c) Technical Availability attribute

High Availability represents the subsystem's readiness for an operational use. This attribute is expressed as the vector of probabilities which shows the subsystem's needs at the different stages of operation.

$$A = [\, a_1 , a_2 , \ldots a_r \,] \quad \Sigma a_r = 1$$

where a_r represents probabilities or quality values for each operational stage of the system

This attribute reflects the needs for the overall support design in which the maintenance can be one of the support design elements (B.S. Blanchard & W. Fabrycky[2]). The requirements of the reliability will influence the level of the support design. The support design is the core element of the Availability. Thus the Dependability will indirectly, through the reliability, influence the Availability attribute.

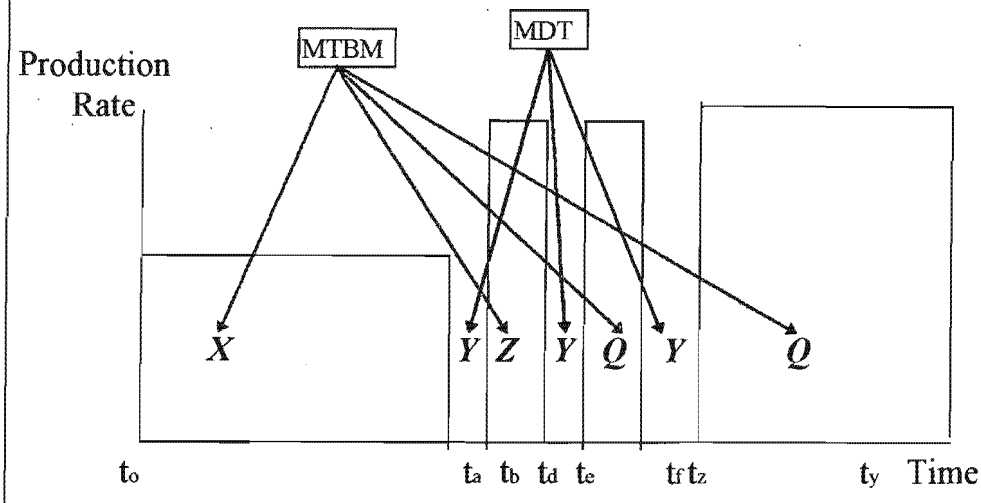
The function of the subsystem's Availability ($A(\infty)$) may be expressed in the following form (M'Pherson[20]):

$$A(\infty) = MTBM/(MTBM + MDT)$$

MTBM mean time between maintenance

MDT maintenance downtime

Figure 2.10 Graphical Representation of Availability Formula



where

X, Z, Q, - represent time intervals for different levels of the system's operations

Y - represents time intervals when the system is at rest (i.e. repairs or failure stage)

Figure 2.10 shows that the Availability is the survival function of the stage because it indicates whether the subsystem is in the operational stage. The same reasoning exists in "Dynamic Programming with Management Applications" by N.A.J. Hastings[6] where the author uses the survival function to explain the Availability. This function gives the probability that a system remains in a state without transition.

Unfortunately, the above reasoning may not always apply to the small developing business. The Availability may not always be a function of time, but a function of the subjective evaluation of the support elements. For example, a certain amount of the spare parts may be subjectively defined as sufficient, small or high for a given subsystem's requirements.

The graphical representation of the relationship between all three attributes (Capability, Dependability, Availability) is shown in figure 2.11.

Figure 2.11 Interrelation between subsystem's attributes through subsystem's elements

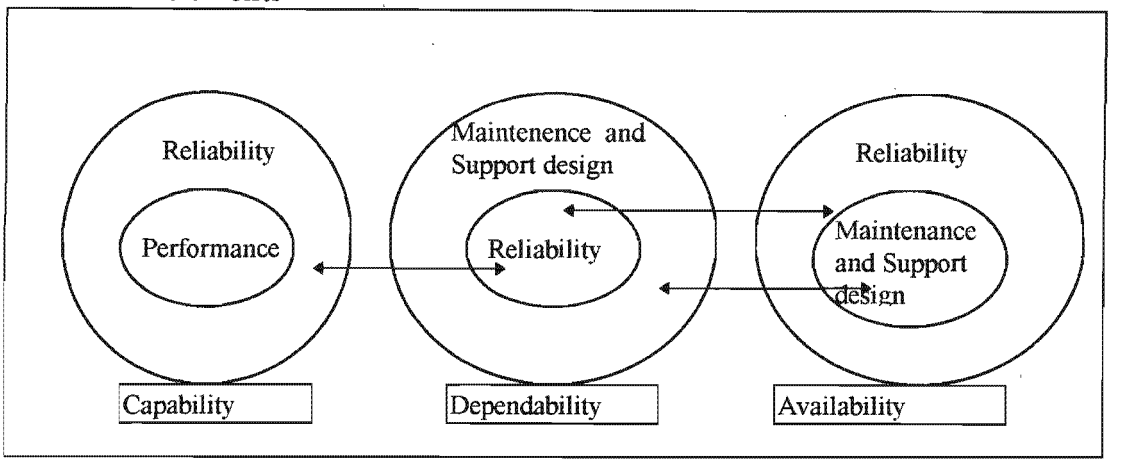


Figure 2.11 shows that the core function of Capability (i.e. performance) is not connected directly with any other attribute. This is because the performance is the measure of the subsystem's final quantitative output. This output (e.g. product) can not influence any of the other two attributes directly but it can only influence alterations of the subsystem's policy.

The above three attributes are directly proportional to the effectiveness of the subsystem (M'Pherson[20]). The effectiveness is a probability which is represented by the following equation:

$$E = A[D]C \qquad 0 <= E <= 1$$

where

A - Availability vector
[D] - Dependability matrix
C - Capability vector

The values of the vectors and the matrices in the above equation are not all probabilities as is suggested by M'Pherson[20]. In a small developing business the Availability and Capability are usually expressed in the form of quality ratios (i.e. there is a limited amount of data to generate the probability values).

It is evident that all three attributes have to be satisfied to produce the optimal effectiveness of the subsystem. For example, if the subsystem is both dependable and capable, it will still be useless if it is never available and vice versa (M'Pherson[20]).

Cost will very often limit all three attributes (Availability, Dependability, Capability). Therefore the cost fluctuations will affect the value of the subsystem's effectiveness. This is further discussed in section 2.4.

2.3.2. Transfer function of the Economic Subsystem (economic Capability)

The aim of the Economic Subsystem is to convert financial input elements (repayments, profit increments, monetary values of interests) into comparable (qualitative) values of economic effectiveness.

The function of the Economic Subsystem will only have one input element (economic performance). Since technical Capability is the function of technical performance, the economic Capability will be the function of economic performance. Values of economic performances (e.g. payments), at different stages of the system's operations, will be compared to the best value of economic performance, resulting in economic Capability values.

2.3.3. Transfer function of the Environmental Subsystem (environmental Capability)

The objective of the Environmental Subsystem is to optimise the output (effectiveness) which results from input elements such as the level of pollution, the impact of noise on the environment etc.

The function of the Environmental Subsystem (the same as the function of the Economic Subsystem) will only have one nature because only one input element is present (environmental performance). Therefore, Capability will be the only attribute of the Environmental Subsystem. The evaluation of environmental Capability will be similar to

the evaluation of the technical and the economic Capability (for a more detailed description see Subsection 2.3.1.(a).

2.3.4 Analysis of Life-Cycle Subsystem

The Life Cycle is an analysis of the system's activities over this system's life-span. The aim of the Life Cycle is to distinguish between those periods of the system's existence when the system is being developed, fully operational and disposed of. During that time various technical, economic and environmental changes are taking place. The Life Cycle analysis forecasts these changes and thus it is a part of the Technical, Environmental and Economic Subsystem's optimisation.

2.4 Output of a system (evaluation of the worth of the whole system)

Finally, all of the subsystems will have to be combined into the total worth of the system. Each output of the subsystem is presented as the ratio of effectiveness to the resource factor.

2.4.1. Output of the Technical Subsystem (Cost-effectiveness analysis)

In the systems engineering analysis, the balance between the cost and the effectiveness measures the optimal subsystem's design.

Cost is defined as discounted whole-life ownership cost of a system. It includes capital cost, operational and support costs, hazard and environmental impact costs (M'Pherson[20]). The general exposure to the cost is given in figure 2.12 below.

Figure 2.12 System Whole-Life (Ownership) Costs

System Whole-Life (ownership) costs $C_S = C_K + C_O$

CAPITAL COSTS

$C_K = C_{RL} + C_{AQ} + C_{KC}$

Realization cost ..... C_{RL}
- Survey
- Preliminary design

} Phase

Acquisition costs ..... C_{AQ}
- Development
- Manufacturing
- Assembly
- Test
- Deployment
- Training

} Phase

Capital charges ..... C_{KC}

as required for:
- R + D
- Operational System
- Support System
- Phase M + I†
- Phase Personnel
- Phase Resources + Support

OPERATING COSTS

$C_O = C_{OS} + C_{SS} + C_{Eth}$

Operating system costs ..... C_{OS}

Equipment:	Resource use, (Support), Modifications, Renewal, Phase-out
Personnel:	Wages, Benefits, Training, (Support), Retirement, M + I
Operations:	Dues, Licences, Facilities, M + I
Marketing:	Research, Advertising, Sales, Personnel, M + I
Intelligence:	Equipment, Personnel, M + I

Support system costs ..... C_{SS}

Personnel:	Victualling, Transport, Health, Recreation, Lodgings, M + I
Maintenance:	Personnel, Monitoring + Diagnosis, Scheduled Maintenance, Equipment, Facilities, Downtime Costs, M + I
Repair:	Personnel, Diagnosis, Unscheduled Maintenance, Equipment, Facilities, Downtime Costs, M + I
Logistics:	Personnel, Spares + Stores Inventories, Transport, Facilities, Procurement, M + I

Ethical costs ..... C_{Eth}

Safety:	Monitoring + Alarms, Equipment, Personnel, Damage, Liability, Insurance, M + I
Environment:	Amenity, Public Relations, Pollution Abatement, M + I

† M + I = Management and Information.

In the case of the small developing organisation the cost may be simplified to include only cost of Capability, Dependability and Availability.

The simple way of combining effectiveness with cost is to evaluate the ratio of effectiveness to cost factor. In this case cost of a certain proposal will be changed into a quality value. This quality value would be the ratio of the current cost to the smallest planned cost of a project. Thus, if the cost factor increases then the project becomes expensive.

The increasing cost often decreases effectiveness and vice versa. Therefore, managers would primarily try to optimise these two values. Optimised cost is not equivalent to the minimum cost but corresponds to the most optimal effectiveness. In other words, cost-effectiveness must be a structure which is able to achieve the best effectiveness at the available level of resources. This idea is schematically shown in figure 2.13.

objective functions are added together to form the total worth of the system. The general formula is given by:

$$f(x) = \sum_{i=1}^{k} w_i \, f_i(x)$$

w_i – *weighted · coefficient*

f_i – *objective · function · (ratio · of · effectiveness · to · resource · factor)*

i – *number · of · subsystem*

Vector Regret Criteria is based on the principle of finding the value which has the shortest distance from the optimum solution. The principle is explained for a system which consists of two subsystems (see figure 2.14).

Figure 2.14 Quality space of effectiveness for two subsystems

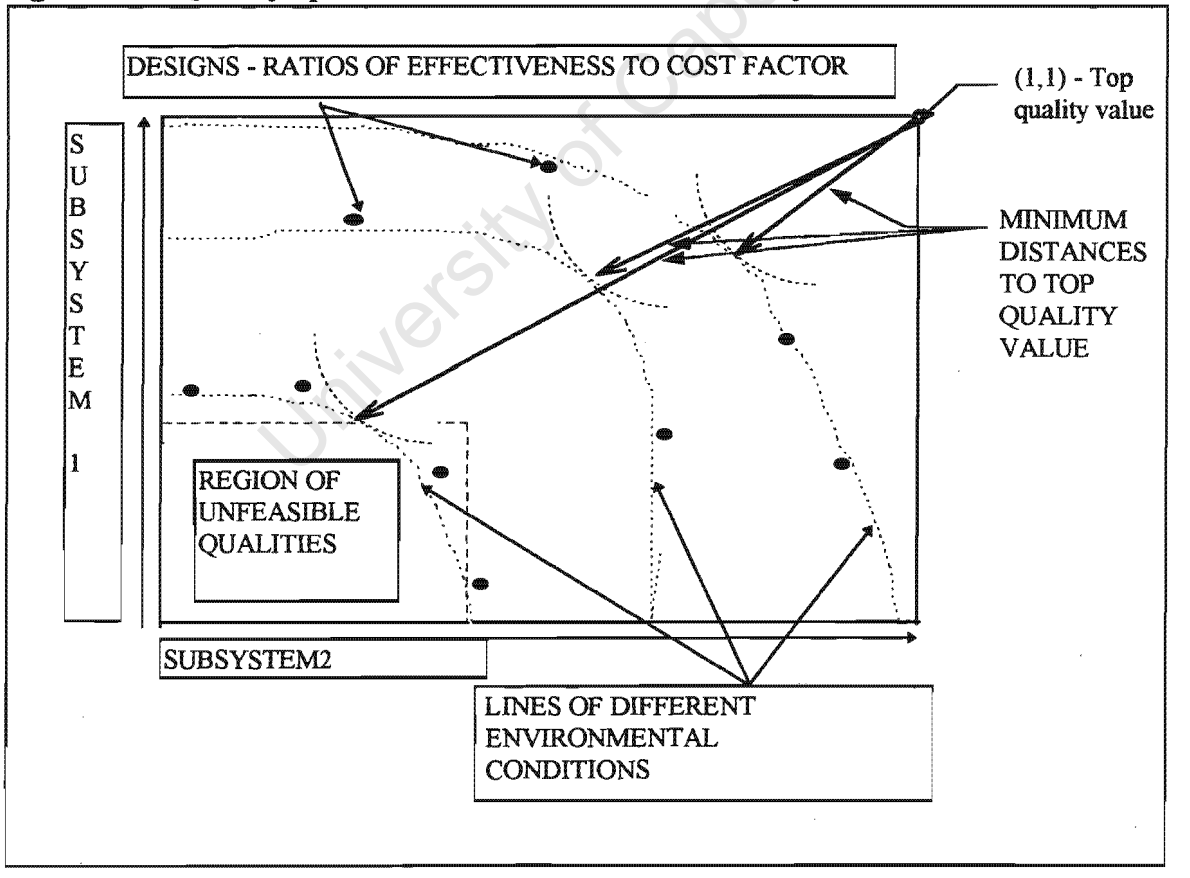


Figure 2.14 shows optimal values of the ratio of effectiveness to the cost factor. These values can be combined into the formula which is known as the total worth of the system (M'Pherson[20]):

$$w_T = 1 - \{w_b^2 + w_a^2\}^{-0.5} \{w_b^2 (1 - q_b)^2 + w_a^2 (1 - q_a)^2\}^{0.5}$$

where

w_T – total worth
wa – weight coefficient of ratio effectiveness to cost factor of subsystem A
w_a – weight coefficient of ratio effectiveness to cost factor of subsystem B
q_a - ratio of effectiveness to cost factor of subsystem A
q_b - ratio of effectivenessto cost factor of subsystem B

Note w_a and w_b are usually assigned equal weights. Otherwise one may use Laplace Criteria (see de Neufville & Stafford[4]).

In the case of two subsystems present (see figure 2.14), one would have to find a two dimensional quality function. This function would generate points (quality values) which would be a certain distance from the target value. The best design with the most optimal value, would be the one which had the smallest distance to the target value.

Min-Max Criteria is used to solve problems where uncertainty is involved. This optimisation is characterised by the maximin rule in the view of expecting the worst outcome and then the maximax rule, which is opposite to the maximin one. In this instance, the maximum or minimum ratio of effectiveness to the resource factor would be selected.

Laplace Criteria is based on the assumption that nature is indifferent (i.e. the outcomes are assumed to be equally likely). Therefore, the total worth would be evaluated as the total sum of all outcomes of all the subsystems' worth divided by the number of these subsystems. In other words, all the ratios of effectiveness to the resource factor are equally important.

2.5 Reflections about the presented theory

2.5.1. Criticism of the subsystems' models

The above theoretical information shows that the Technical Subsystem has a developing nature. The subsystem has more than one attribute, each of which has a unique purpose for the subsystem's growth and development. For example, Capability indicates how the system meets operational requirements, reliability shows the system's ability to meet operational stresses and Availability indicates the system's readiness to operate. When the attributes are combined together, they evaluate effectiveness of the whole subsystem (measure of the subsystem's adaptability to the environmental requirements).

The remaining type of subsystems (economic and environmental) have a more mechanistic nature. This means that they consist only of the Capability attribute which optimises the subsystem's performance requirements. When using this type of evaluation, it is impossible to analyse the subsystem's adaptability to the environmental changes and requirements. There is a definite lack of Dependability and Availability attributes.

Thus it seems that the Technical Subsystem optimisation will be a main contributor to the development of the whole system. However, this statement needs additional investigation because all subsystems in a small developing business may be at the stage of development. The author will attempt to approach this issue in Chapter 4.

2.5.2. Search for the optimisation tools

The presented systems engineering theory reveals that subsystems are the functions which transform a system's inputs into a single value of a multi-objective system's output. The specific nature of the system's inputs in the small developing business (Section 2.2), requires a specific method of transformation.

The tools which optimise the properties of the input elements (Section 2.2) have the following characteristics:

- performance should be optimised by the method which can evaluate non linear and discrete values in the stage-wise process
- transition rate of reliability should be generated by a simulation method which can optimise random values
- supportability should be evaluated by the methods which can account for the subjective, non linear and discrete values.

The search for the specific methods applicable for the optimisation of the above input elements is described in Chapter 3.

3. SELECTION OF TOOLS FOR OPTIMISATION OF THE INPUT ELEMENTS CHARACTERISTIC FOR THE SUBSYSTEM'S ATTRIBUTES

In the previous section key attributes of the systems engineering approach were defined. The measures of the attribute evaluation were also presented. These attributes are : Capability (performance), Dependability (reliability) and Availability (maintenance and support design). These attributes play the most important role in modeling the operational subsystem's feasibility. Together, they result in the subsystem's effectiveness and as the effectiveness values of all subsystems result in the total worth of the system, so the attributes influence the effectiveness of the whole system.

Therefore, the objective of this section is to find the appropriate tools to evaluate the most optimal configuration of the attributes and their elements. Before any analysis is started it will be important for the designer to realise how the elements of the system depend on one another. We need a method for investigating these dependencies before we can consider optimisation. The next section deals with such a method.

3.1. Initial considerations of the analysis

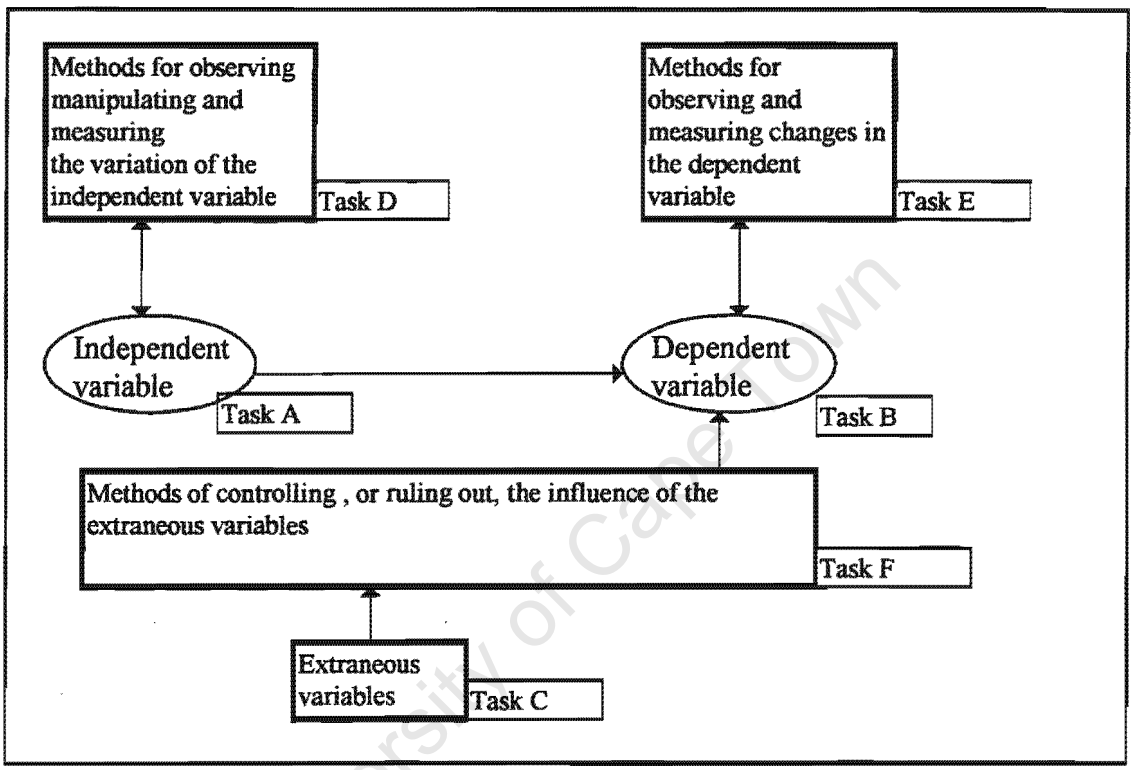
Most researchers follow a common process. The general research methodology described by John Gill and Phil Johnson[1] in "Research Methods" is as follows:

1. Find theoretically dependent variables (i.e. phenomena or factors whose variation is understood)
2. Find theoretically independent variables (i.e. factors whose variations are tested and whose variations change the dependent variables)
3. Create operational scenario in which independent variables are changed. In this case variations are monitored and compared to the system's limitations.

4. Find variations in the dependent variables whose outcomes are not necessarily the result of the action of independent variables. In this case one deals with so called extraneous variables.

The schematic structuring of the process is shown in the figure below:

Figure 3.1 General structure of the research approach



The authors of the above process identify tasks A,B,C as theoretical issues where the designer describes definitions. Tasks D, E, F are methodological issues where observations, measurements and controlling are taking place.

Usually, the input values (allocated resources) to the system/subsystem are regarded as independent. The pattern of variation of these values is tested against their outcomes (dependent variables such as production output, system's reliability). Therefore, dependent variables represent those elements of the subsystem which are in some way related to independent variables. For example, high performance equipment is related to the money

allocated to buy this equipment; reliability depends on the time in which reliability maintains its desired value, etc.

The unexplained change of the system's outside environment may create new inputs to the subsystem. These new inputs (i.e. noises or disturbances) may not be possible to explain in relation to the variations of the system's input elements. Those inputs could be classified as extraneous variables. As is the case of the inputs, the elements of the subsystem may be:

- - independent
- - discrete
- - non-linear.

This is the distinguishing point of the analysis where the systems engineering analysis for big developed business is continued, but in the small developing business, the analysis is stopped at this point. The result of the continued analysis (investigation of the extraneous variables) is the creation of equations of state of the system. In this case, all of the input elements to the system can be interrelated by the mathematical equations.

In the case of the small developing business (no further investigation of the extraneous variables is undertaken), the search continues for the methods which could accommodate unexplained variations and still give a reasonable result of the elements' optimisation. In this case, the methods should be able to optimise non-linear and discrete variables.

After realising which sets of values are dependent/independent and controllable/non-controllable, the next step is to optimise them. As it was mentioned earlier, performance is a very significant element of any type of the system/subsystem. It describes the magnitude of the physical output and very often, in the small business environment, is regarded as the measure of the whole system's effectiveness. It was also described in the previous section that performance is the only element of certain types of subsystems (i.e. economic, environmental).

3.2. Use of Dynamic Programming for Optimising Capability

3.2.1 Reason for choosing dynamic programming

There are various classical mathematical techniques which can be used to optimise performance. Most of them use differential equations, and others, calculus techniques. However these techniques can be applied after finding a direct interrelationship between collected elements. This interrelationship (in most cases) can be identified by the experienced mathematician. In the case of the small business this can be a rather costly procedure because every optimisation procedure would need a supervision. Therefore, the author decided to look for the most universal technique which could be applied in most cases. Linear and dynamic programming are two of the possibilities.

The linear programming allows the designer to optimise the multi-objective problems (e.g. system/subsystem with more than one performance objective) however, it can not solve problems where the relationship between the elements is not linear. Since the interrelation between the performance elements is not linear, the linear programming will be an unattractive choice.

Dynamic programming can be applied to both the linear and non-linear problems, although, in the case of linear problems, linear programming works better. Dynamic programming is particularly useful if:

- systems are non-linear
- regions are feasible and non convex
- variables are discontinuous

Dynamic programming has some limitations. It can be applied to a system/subsystem with one objective function. Dynamic programming would also be limited to the systems where the approach is stage-wise. It means that every move to the next stage is influenced by the decision which has been made in the previous stage.

The same description of dynamic programming was also found by de Neufville & Stafford[4], Richard de Neufville[3] and D.J. White[5].

The limitation of dynamic programming will not affect optimisation of the performance value of the small developing system. In the small organisations usually one objective is considered and analysed through operational stages of the considered system (multi-objective analysis is complex and requires advanced skills from the designer).

In the small business every project is often divided into stages. This procedure helps to identify the states/stages where the system's performance is changing its value. Therefore the resource requirement for the respective value of performance is monitored more closely. The sequential interrelationship between the stages (choice of the state of the following stage depends on the choice of the state made in the previous stage) is advantageous because the interrelation and dependence between the performances of the system can be identified. For example, one may analyse the three stage manufacturing system which is represented by three different types of work stations (i.e. washing, cutting, packing). For each work station there is more than one choice to make in order to choose the right washing machine, cutting machine, packaging machine. The choices (in the feasible resource allowance) made in relation to the machine (state) for every stage will result in a certain total performance value. There may be more than one 'path' (combination of states/machines) leading to achieve the same required level of performance. These paths identify the various makes of machines (and their performances) which, when put together, achieve the required total performance. Therefore different states/performances are related to each other and their importance is identified in the whole system.

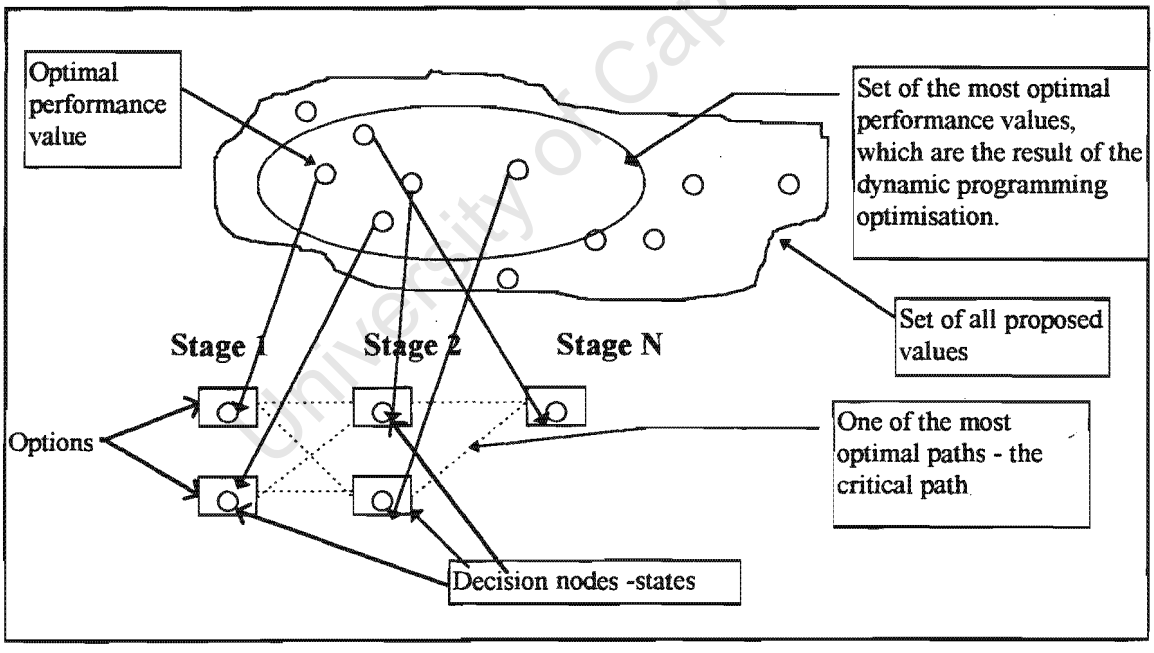
3.2.2 Characteristics of the dynamic programming optimisation

Dynamic programming is a mathematical strategy which can pick 'disorganised numbers' (values which do not have a mathematical pattern) and group them into organised sets.

These sets are then connected with each other by the most 'optimal' links (i.e. critical paths that connect most optimal states). The optimal solution (i.e. critical path) is found by a sequential decision process in which a complex problem is broken into simpler subproblems called stages (e.g. intervals of time or work stations of the manufacturing system). At each stage (i.e. subproblem), there is more than one state or position (i.e. values of performance) where the considered system can be present. Therefore, in each stage, a decision is taken (a state is chosen) after which a transfer to a next stage takes place. The chosen optimal state (or a optimal set of states) of the previous stage is used as the initial value for optimisation of the next stage. The process continues until all stages are solved.

A general graphical explanation of the dynamic programming procedure is shown in figure 3.2 below:

Figure 3.2 Graphical explanation of the dynamic programming procedure



The Dynamic Programming technique can show us which group of data is the most optimum for the particular system and for every intermediate stage of the system's operation. Also, it categorises (according to the hierarchy of importance) which group of

values should be considered first and which should be followed next according to resource allocation.

However, modeling of the dynamic programming may be awkward, because it has two structural limitations (MEC 532Z- Prof. T.Ryan lecture copies 1994):

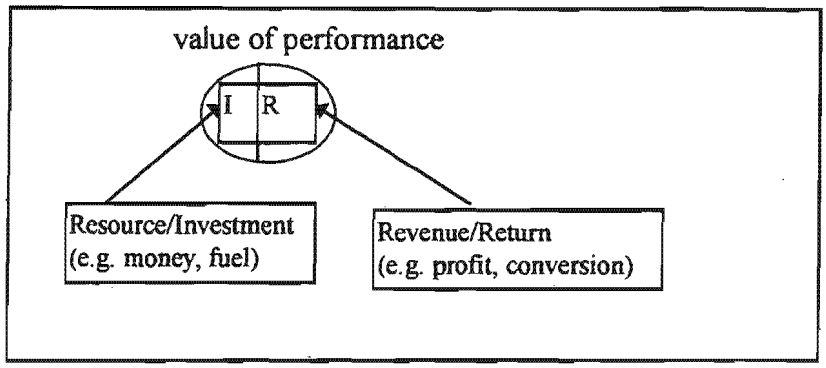
- dynamic programming has no general algorithm (i.e. every time a problem differs slightly, a new function is designed)
- dynamic programming is exponentially affected by the amount of computation (i.e. if dimensionality of the problem is double in size then the amount of computation quadruples)

In the small developing organisations (see Chapter 1) objectives are considered in the hierarchy of importance (i.e. primary, secondary objectives). Therefore, their optimisation is often considered separately. Thus each type of set of performance values (e.g. technical, economic, environmental) may be optimised by the separate, single dimensional dynamic programming procedure.

In this case, optimised performance consists of two components -Revenue (component which is optimised due to Investment's allocation to the system's stages) and Investment (component which is allocated to the system's stages).

The Revenue will represent the value of performance where Investment will represent the value of resource required to maintain this performance. The performance is called single-dimensional because it contains only one type of Revenue (see figure 3.3).

Figure 3.3 Single-dimensional model of performance



In the small developing business environment the Return will be represented by a numerical value (not a function) and performance has a single-dimensional structure. Therefore, it is possible to develop the general dynamic programming model. In this model, resources (Investments) are optimised by the Return function and depend on the number of activities, n. The total Return function can be expressed by the following formula:

$R(x1,x2,....xn) = g1(x1)+g2(x2)+.....+gn(x)n$

x1...xn - allocated resource (Investments) to stage from 1 to n

g1(x1) ...gn(x)n - optimal Return values from stages 1 to n

The quantity of the total resource always has a specific limitation. Therefore, the quantity has a constraint:

$Q = x1+x2+.....xn \qquad xi >= 0$

Returns, R, can be maximised or minimised, depending on the quantity of the total resource (Investment) Q.

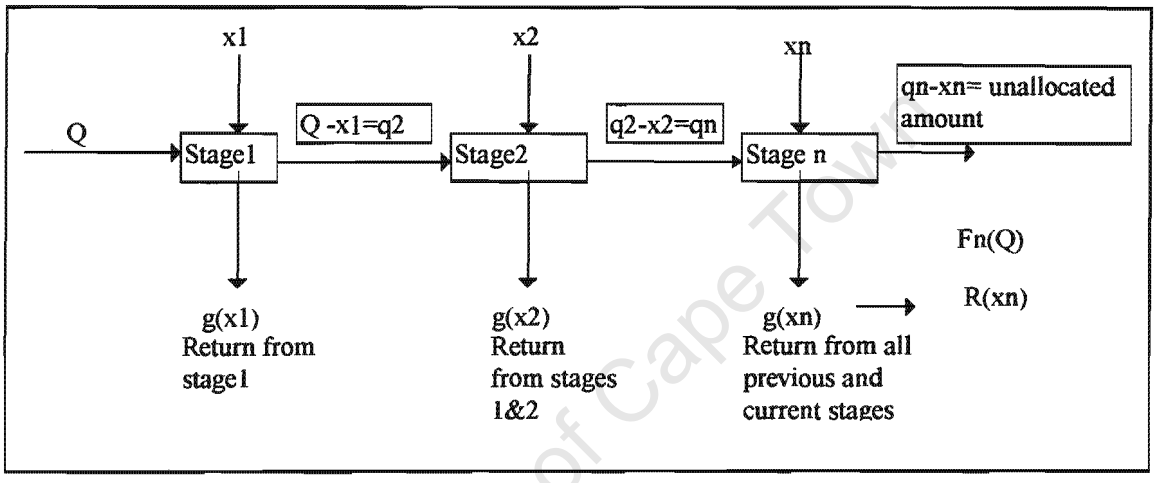
$$Fn(Q) = \max_{\min}\{R(x1,x2,.....xn)\}$$

The resources (Investments) are allocated one at a time. Thus, if allocation is finished for stage 1 then the next resource is allocated to stage 2. This is called a forward recursive procedure. The Return function from every activity may be expressed as:

$$Fn(Q) = \max \text{ or } \min(gn(xn)+gn-1(Q-xn))$$

Graphically, the general dynamic programming process can be shown in figure 3.4:

Figure 3.4 General structure of the single- dimensional dynamic programming process



The above figure shows the process as the sequence of separate allocation processes.

Further reading, related to the topic of dynamic programming, may be found in H.A. Taha[7] " Operations Research An Introduction".

3.3. Use of Fuzzy Logic for Availability Optimisation

3.3.1 Reason for choosing Fuzzy Logic

Availabilities characterise the ability to support the behaviour of a system's stages. As it was pointed out earlier, analytical solutions are difficult to use in the small business corporations. The analysis very often is based on the subjective evaluations drawn from experience. For example, a system, which operates at temperature 60°C for 2 hours, may have a very high Availability for one observer and very low for the other. Still, the values

defined as the Availability may not be related to their respective outputs in linear mathematical patterns. This is the case of the small corporation where there is often a lack of mathematical models which could describe the situation of the support design.

The support design is very often described in a linguistic manner rather than in the mathematical form. Therefore, usually applied methods such as: statistical measures of maintainability, inventory system's method or spare parts Availability methods, etc. (see B.S Blanchard & W. Fabrycky[2]), can not model the support design. Currently, Fuzzy Logic is the only method which provides an interface between linguistic statements and the implementation of these statements in numerical form (B. Kosko[15], A. Bastiani & C. Speedy[18], D.G. Swartz[24]).

Another advantage of Fuzzy Logic is in its applicability to the world of non-linear functions. Therefore, Fuzzy Logic will most likely be an appropriate choice to the support design.

3.3.2 Characteristics of Fuzzy Logic optimisation

What is Fuzzy Logic?

Fuzzy Logic is a set of subjective statements/rules, imposed by a designer, which are expressed in the following form :

" IF conditions THEN actions"

All of the rules are subjective, imprecise and context dependent. These characteristics may be found in most managerial case studies, especially in modeling a supportability system.

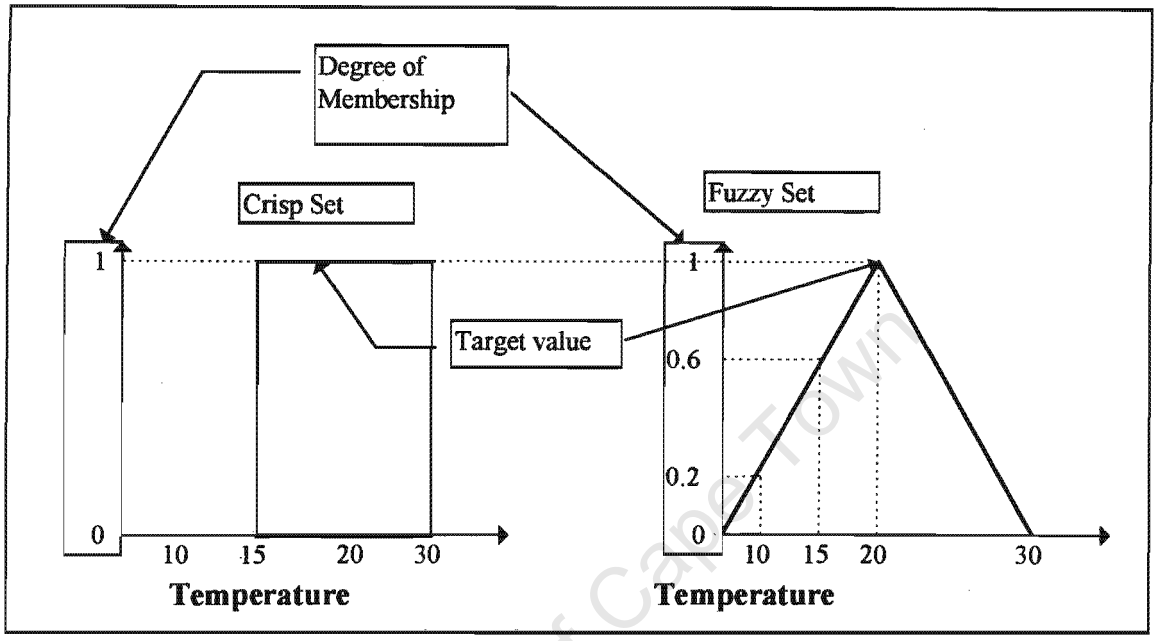
How does fuzzy logic work?

Fuzzy Logic is a combination of fuzzy sets . Unlike crisp sets (traditional sets), fuzzy sets are the group of values which entirely or partially contains a certain degree of

membership. The degree of membership shows how far a given value is from the ideal value (i.e. target).

Figure 3.5 below depicts a comparison between crisp and fuzzy sets.

Figure 3.5 Comparison between crisp and fuzzy sets

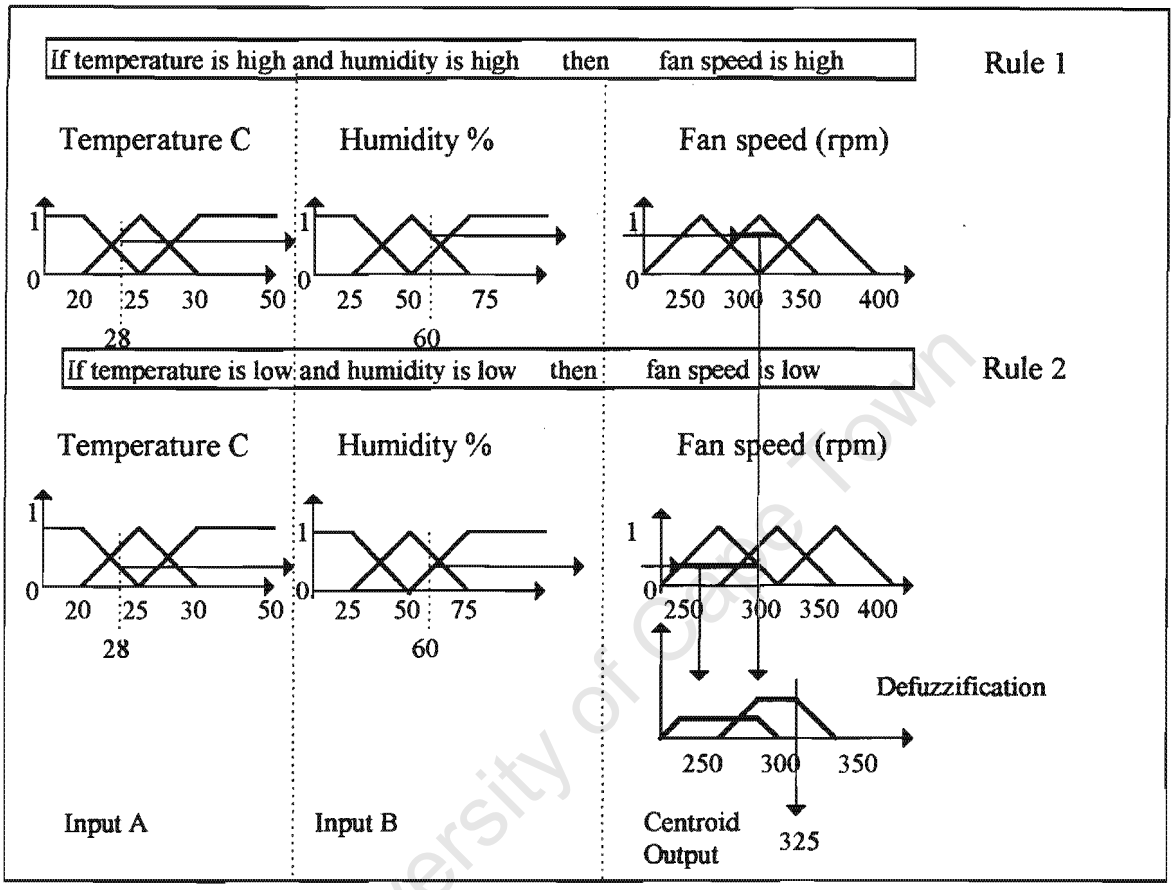


The above comparison of sets of temperatures shows that the Fuzzy Set has a certain degradation related to the magnitude of 'warmness' (i.e. the membership criteria range from 0, 0.1, 0.2 till 1). In the Crisp Set, temperatures are 'high' or not 'high' at all (i.e. the membership criteria has only value 0 or 1). A similar problem one may find in traditional operation research/operational management. Most of the methods used for decision-making (probabilistic methods in decision trees) model the environment using Crisp Sets.

One may notice that more than one condition of the same environmental factor may be applied (usually three, five or seven sets of values are considered -A. Bastiani & B Speedy[18]). These conditions can overlap each other, forming a continuous pattern of non-linear sets of values (see figure below) At the same time, accompanied environmental

factors can also be evaluated together (e.g. temperature and humidity are needed to model air condition).

Figure 3.6 The Fuzzy Logic principle



As it is seen in the above figure, for two environmental factors usually two rules are applied. The output of every rule is added together in the process of defuzzification. The final value (the final output value of the combined single fuzzy outputs) is found as the centroid of combined fan speeds. The most important gain is the creation of numerical value by the subjective and word connected patterns.

3.4. Use of Monte Carlo Simulation for Dependability Optimisation

3.4.1 Reason for choosing Monte Carlo simulation

So far, techniques for sorting values of Capabilities and Availabilities have been considered. The attribute, indicating the operational change of a system, may have a different character in comparison to previously mentioned Availabilities and Capabilities.

Dependabilities depend not only on technical but also on environmental factors. As Dependabilities measure a system's adaptation to environmental changes it is more difficult to predict them than to predict Availabilities and, especially, Capabilities. Therefore, Dependabilities are very often represented as probabilities of a system's resistance to external/internal changes.

The probabilities are usually evaluated from exponential equations of reliability functions. However, each of these equations requires a rate at which the reliability is changing its value (during analysed period of time). This rate is called a transition rate. As it was mentioned earlier, transition rates may represent change (e.g. in the form of repairs or failures of the state) where a system moves from one operational state of a stage to another.

D.J Sherwin[11] suggests two solutions for modeling reliabilities viz. the exact solution method and the approximated solution method. The exact solution method requires understanding of the Laplace transformations. Every time a new situation is created, a new equation, using Laplace transformation, must be formulated.

Unfortunately, in most cases (i.e. for small developing business), the structure of stages (i.e. interconnections between states) and transition rates have changing patterns. The occurrence of transitions rates is often random. Thus, the whole procedure is difficult to model mathematically. In the case when the structure of the stage transition is not known or difficult to model for every separate situation, the approximated solution method is

suggested. Since the change of the transitions may also be very often time-dependent, the Monte Carlo simulation is recommended as a method to evaluate the averaged transition rates (D.J Sherwin[11]).

3.4.2 Characteristics of Monte Carlo simulation

The typical use of the Monte Carlo simulation in solving complex problems would include the following conditions (MEC532Z - Prof. T. Ryan lecture notes 1994).

1. The problem may contain a large number of uncertainties, each of which has many or even an infinite number of possible outcomes.
2. The probabilities of the events in the problem may be influenced by a course of action previously selected, or outcomes of previous uncertainties.
3. The value of the final outcome to the decision-maker may depend on the detailed sequence of events that lead to the final result, not simply the final result itself.
4. There may be a combination of points 1,2, and 3 above, which are typical of most planning problems, particularly those involving evaluation over an extended period of time.

Generally, this method may be divided into a static and dynamic part. The static part of Dependabilities evaluation may be enclosed in gathering data of failure, repair or transition rates. The dynamic part starts with analysis of the random occurrence of values. This method picks the frequencies of occurrences of random values and then transforms these frequencies into their respective probability distributions. The allocation of random frequencies into their probability distributions helps to identify which values are more likely to occur and, at the same time, which values can be the most significant for a considered system. Also, long and fair generation of random numbers, using the above technique, can result in the true probability of occurrence for those values which are not possible to model with conventional mathematical techniques.

The other sectors of numbers fall within the 55% range from 0.35 to 0.89 and 10% range from 0.90 to 0.99. This finally will show ranges (accumulated probabilities) where occurrences are more or less likely to occur.

3.5 Reflections about the presented tools

The above chosen tools for optimisation of a subsystem's elements can not result in such precise values as the analytical methods (mathematical equations). However, they do not require a large amount of financial investment and they do not need highly trained scientific personnel.

Most of the data, in the environment of the small developing business, presented for the optimisation, is non-linear and discrete. Thus, Dynamic Programming, Monte Carlo Simulation and Fuzzy Logic are the tools which are applicable to this environment.

- Dynamic programming optimises the disorganised values of performance and places the most optimal performances in the relevant stages of system operations.
- Monte Carlo simulation generates the random occurrence of every operational stage and therefore is used to optimise unpredictable behaviour (rates of transitions) of the small developing business.
- Fuzzy Logic optimises the subjective information which is characteristic in the support design.

All of the mentioned methodologies generate values which can be further used for evaluation of Capabilities (from optimisation of performance), Dependabilities (from optimisation of transition rates for reliability evaluation) and Availabilities (from optimisation of support design) (see Subsection 2.3).

The above methods of optimisation may still be time consuming if various options of resource allocations would be considered. Therefore, there is a need to develop the

software support which could speed up the optimisation process. However, before the software can be developed another dilemma has to be resolved. That is, one must know how performance (the primary element in the evaluation of the small developing system (Chapter 1)) can be incorporated with the rest of the elements (at the moment only the Technical Subsystem reveals the property of having the performance, reliability and the support design). If the performance will be the primary element then how will dynamic programming be a controlling method? How will dynamic programming be connected to other methods (Simulation, Fuzzy Logic)? These questions will be answered in Chapter 5.

4. THE DEVELOPMENT OF THE SYSTEMS ENGINEERING APPROACH FOR A SMALL BUSINESS APPLICATION

The development of a system is measured by the total worth of this system. In Chapter 2, the author pointed out that the total worth of a system depends on the combined ratios of Effectiveness to Resource Factor. These ratios represent the outputs of a system's subsystems and effectiveness is the combined result of the subsystem's attributes.

There are three attributes (i.e. Capability, Dependability, Availability) for the Technical Subsystem (M'Pherson[20,21] and B.S Blanchard & W. Fabrycky[2]) and only one attribute (i.e. Capability) for Economic and Environmental Subsystems. Therefore, the Technical Subsystem was identified as the only developing subsystem. This, however, contradicts the theory of the small developing business where all subsystems should be regarded as developing. Thus, the author will argue that every type of subsystem can be evaluated using all three types of attributes.

If every subsystem is regarded as developing then usually each of these attributes could be equally important to the system. However, in the small developing business one subsystem is usually regarded as primarily important (i.e. evaluating the main objective) where the rest of the subsystems are of secondary importance (i.e. evaluating the secondary objectives). The author will try to use systems engineering theory to combine a subsystem's developing character with the subsystem's importance to the total evaluation of the system.

4.1 Use of subsystem's three attributes (Availability, Dependability, Capability) to evaluate effectiveness of any type of subsystem.

This section starts with a description of the important role all three attributes play to evaluate any type of subsystem (an example of Economic, Environmental and Social Subsystem is given). Attention is then focused on the factors which could be used to

optimise the three attributes. This is important because attributes evaluate the final value of the subsystem's effectiveness. These factors are called the Control Input Values and they identify the resource needed for each value of the attribute. The value of the factors is used to evaluate the Control Input Factor for the ratio of Effectiveness to Control Input Factor. This ratio will replace the ratio of Effectiveness to Resource Factor of the entire subsystem. Finally, a new general structure for evaluation of the small developing business is identified in accordance with the above findings.

4.1.1 Reason for use of subsystem's three attributes in any type of subsystem evaluation

At this stage one may ask the question: can one evaluate any type of subsystem by use of three attributes (Availabilities, Dependabilities, Capabilities)?

In the author's opinion the pattern of evaluating any subsystem is similar. Any type of subsystem has the effectiveness which can be the function of the above three attributes. In other words, all subsystems should have a dynamic nature unless they are not self-developing in a given period of time or, more generally, during execution of the stages of the operational process. Researchers, B.S. Blanchard & W. Fabrycky [2], claim that *"It is recognized that the system is static only in a limited frame of reference. A bridge is constructed over a period of time, and this is a dynamic process"*. Obviously, this thought may also be referred to the component of a system (subsystem). B.S. Blanchard &W. Fabrycky [2] give the example of the fire department which consists of buildings, the fire engines, the communication equipment and maintenance facilities. It is obvious that all of these components (subsystems of the fire control system) must have certain performance criteria, must be reliable and need support during operations. Therefore, they have to be evaluated by the three characteristic subsystem's attributes (Availability, Dependability, Capability). M'Pherson[20] in his definition of the 'Integrated System Design' says that *"The inner part defines the process where the systems designer is working hard to provide the design framework within which the subsystem and detail design teams should operate. This may be described as:*

Performance

Reliability design

Support design

Integration through mutual trade-offs

Optimisation through cost-effectiveness analysis"

As the above claim characterises a subsystem where performance defines Capability; reliability design defines Dependability and support defines Availability, thus, every subsystem can have effectiveness that is a function of the above mentioned subsystem's attributes.

The examples of a subsystem's evaluation, using three subsystem's elements, other than technical, are as follows:

Economic Subsystem

There can be more than one option to be chosen for Pay-back time or Return on investment for one project. For example, already designed equipment can be repaid at different stages of time (i.e. repayments as economic performance values can be optimised to analyse economic Capabilities of the project repayment). The repayment can be limited by the time (economic resource factor). The repayment/money may be available with higher or lower probabilities (i.e. probabilities can represent support of repayment plan), depending on a state/stage under consideration. These Availabilities may be altered due to external factors (inputs to support design of repayment) such as interest rates, strength of a currency etc.

The jump from one repayment stage to another (Dependability) can be random and depends on a profit. Failure of a state can be seen as the ability to move into another state of repayment. The repair rate may be seen as a drop to a previously considered state of repayment.

Environmental Subsystem

In this subsystem one can consider such aspects like pollution or the resources of raw materials. Again, one may start to consider environmental Capabilities which may be represented by levels of the resources/pollution. The optimisation of those resources can be limited by a magnitude of available capital or system technical performance (input values to dynamic programming, environmental Capabilities optimisation).

Dependabilities may be represented as rates (probabilities) of falling/rising levels of pollution/resources from one operational state of a system to another. Availability may be a time, available resources, etc. of states within the considered stages of operation.

Social Subsystem

A very similar scenario can be seen in a Social Subsystem which may be a part of an Environmental Subsystem (e.g. internal environment). Various options of subjectively evaluated systems' aesthetics can be used to satisfy workers/staff. These options may be limited to financial resources (input value to Capabilities optimisation).

A 'satisfaction' may be considered as social environment Capability. Change of a 'satisfaction', which is due to failure or improvement of working conditions, may be described by Dependabilities. The Availability of a 'satisfaction' may describe available economical resources, a system's technical development etc.

Therefore, one may see that any type of subsystem (the developing subsystems) may have at least three types of attributes, namely Capability, Dependability and Availability. However, any optimisation must have its limiting factors. Therefore, optimisations of the above-mentioned attributes must be limited by their respective limiting factors.

4.1.2 Control Input Value as the limiting factor of optimisation of a subsystem's attributes

Considering a Technical Subsystem, one may say that Capability is a function of performance which may be limited by money; Dependability is a function of reliability which may be limited by time; and Availability is a function of support design which may be limited by the quality level of delivered spare parts. All of the above limiting variables are named here as Control Input Values.

It should also be noticed that whole life cost (the factor which limits the technical effectiveness - see Subsection 2.4.1) may also be called Control Input Value. However, one must be aware of the fact that Control Input Value of the effectiveness function may not be the same as Control Input Values of any of the mentioned subsystem's attributes. If one would like to analyse first the trade-off between the effectiveness of a subsystem and its Control Input Value then he/she should find that Control Input Values of all subsystem's attributes have the same nature (e.g. all are representing costs.) If this is the case then the Control Input Value of the entire subsystem can be the sum of Control Input Values of all subsystem's attributes. In the case when Control Input Values of the subsystem's attributes are diversified in nature, the Control Input Value of the most important attribute (usually the Capability function of performance is regarded as the primary attribute) can be used as the limiting factor of the subsystem's effectiveness function.

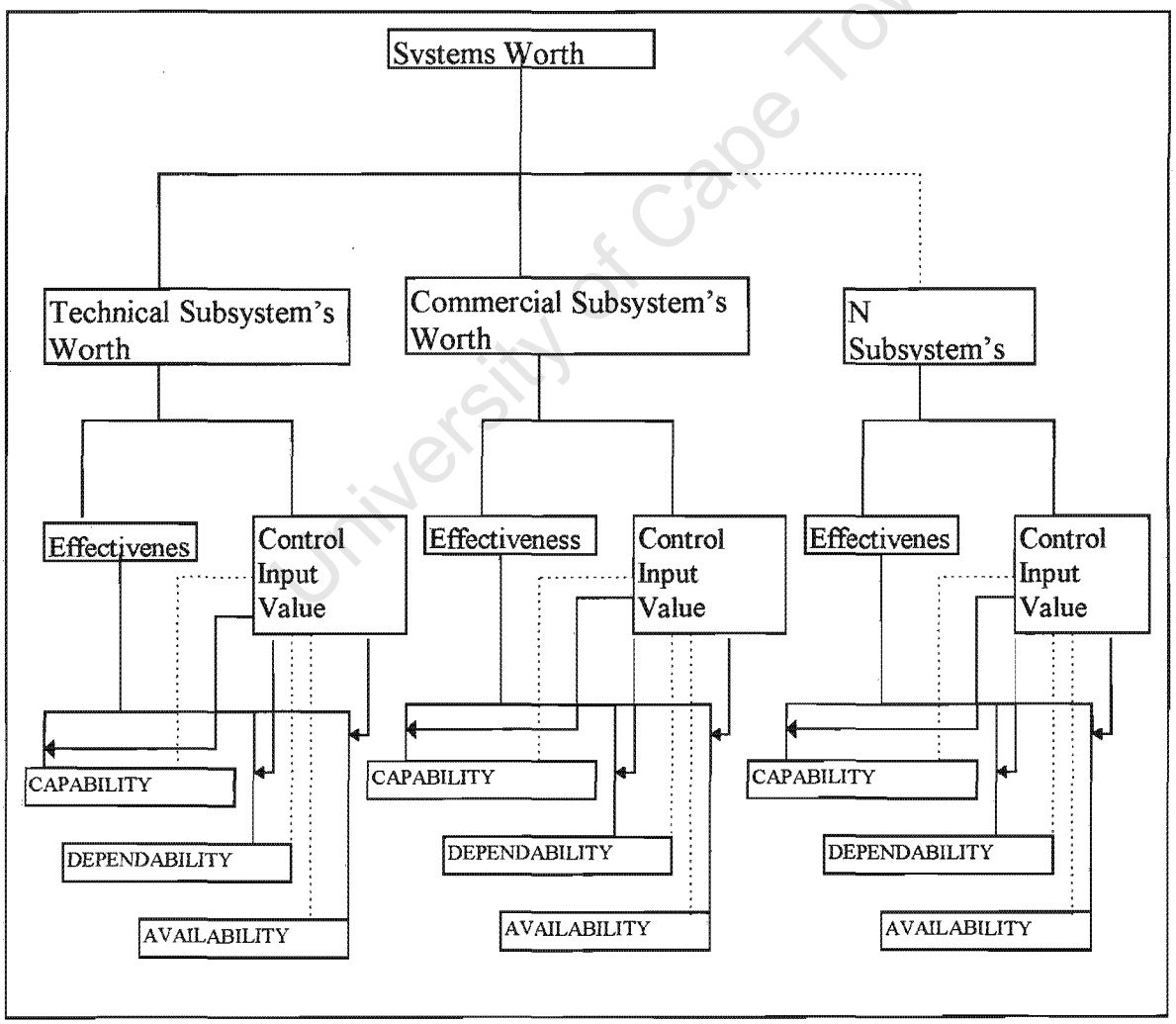
The Control Input Value is combined with the Effectiveness as the ratio of Effectiveness to Control Input Value Factor. The Control Input Value Factor is the ratio of the currently considered Control Input Value to the 'smallest' (or most optimal) Control Input Value.

4.1.3. Proposal of the general structure of systems engineering evaluation in relation to small business development

So far, the appearance of the General Tree Structure shown by M'Pherson[20] (see Figure 2.8 in Chapter 2) becomes progressively modified because every type of subsystem is evaluated in the same way as the Technical Subsystem. Therefore, the general structure of the system's evaluation will have to be revised for the small developing business.

Thus, the M'Pherson theory is not changed but it is rather considered from a more general point of view. This general structure is shown in figure 4.1 below:

Figure 4.1 The general structure of the small developing business evaluation



The above approach may characterise the small or newly developing business. Every subsystem is assumed to be developing because a developing organisation/system tries to improve the value of any component of its activity uniformly . For example, the new chemical company which wants to enter the market to produce and sell chemicals, will have to: develop an efficient technical process to produce more, but with improved quality of products; develop an Economic Subsystem to decrease the expenses and increase the profit; develop the Environmental Subsystem to build a good relationship between the customer and the company (communication between the customers and the company would deliver feedback about the sold product).

In this way all subsystems will have to be developed before they reach a certain level of saturation. After the 'saturation point', the organisation/system becomes developed. Thus, certain subsystems may serve the purpose of the one subsystem (i.e. Main Subsystem) which is a main contributor to further development of the entire system. Perhaps this is one of the differences in the management between developing and developed systems.

Three attributes of the subsystem monitor the development of the system. Capability monitors improvement of the subsystem's performance; Dependability monitors adaptability of the subsystem's structure to its environment; and Availability shows the support requirement necessary for given development.

It has been mentioned that subsystems may have more than one nature (e.g. not all need to be technical in nature). The example of the fire department given by B.S. Blanchard & W. Fabrycky[2] is a typical choice of the system with only a technical nature. Of course, the fire control system can be further analysed to reveal also the existence of economic and Environmental Subsystems. But, as it is known in the systems theory, it depends on the systems analyst where he/she would form the boundary condition in the investigation of a system.

4.2. General approach to systems analysis applicable to small developing business

In the environment of the small developing business one subsystem is treated as primary and the others as secondary. All of these subsystems are assumed to be in the developing stage.

The approach to the systems analysis applicable to small developing business is as follows:

- identify the general hierarchy between subsystems and their meaning in the system's evaluation
- identify the boundaries between subsystems
- identify how to connect subsystems using the primary attribute of each subsystem.

4.2.1 Hierarchy of subsystems

Often, the Main Subsystem evaluates the main objective of the entire system.. The objectives of the Secondary Subsystems are generally influenced by the objective of the primary subsystem (subsystem in focus.) However, the internal environment of the Secondary Subsystem is isolated from the Main Subsystem because all components of all subsystems serve, exclusively, to fulfill the purpose of their subsystem (see figure 4.3).

For example, a chemical company wants to produce certain chemicals where the production process depends on the chemical reaction of specific gases. These gases are supplied in a mixture containing undesired gases (e.g. undesired gases may prevent the progress of chemical reaction). Therefore, the development of the production system will have to start with a selection of the proper equipment to purify the mixture of the supplied gases. In this case, the Technical Subsystem is placed in the main focus. As the selection of the suitable equipment is finished, various options of payment may be considered in the Economic Subsystem (i.e. 2, 5, 10, year plans). The purpose of the Economic Subsystem

is to find the most optimal option for repayment of the selected equipment. Along with the economic, the Environmental Subsystem may be considered. The analysis of the Environmental Subsystem may involve the utilisation of undesired gases.

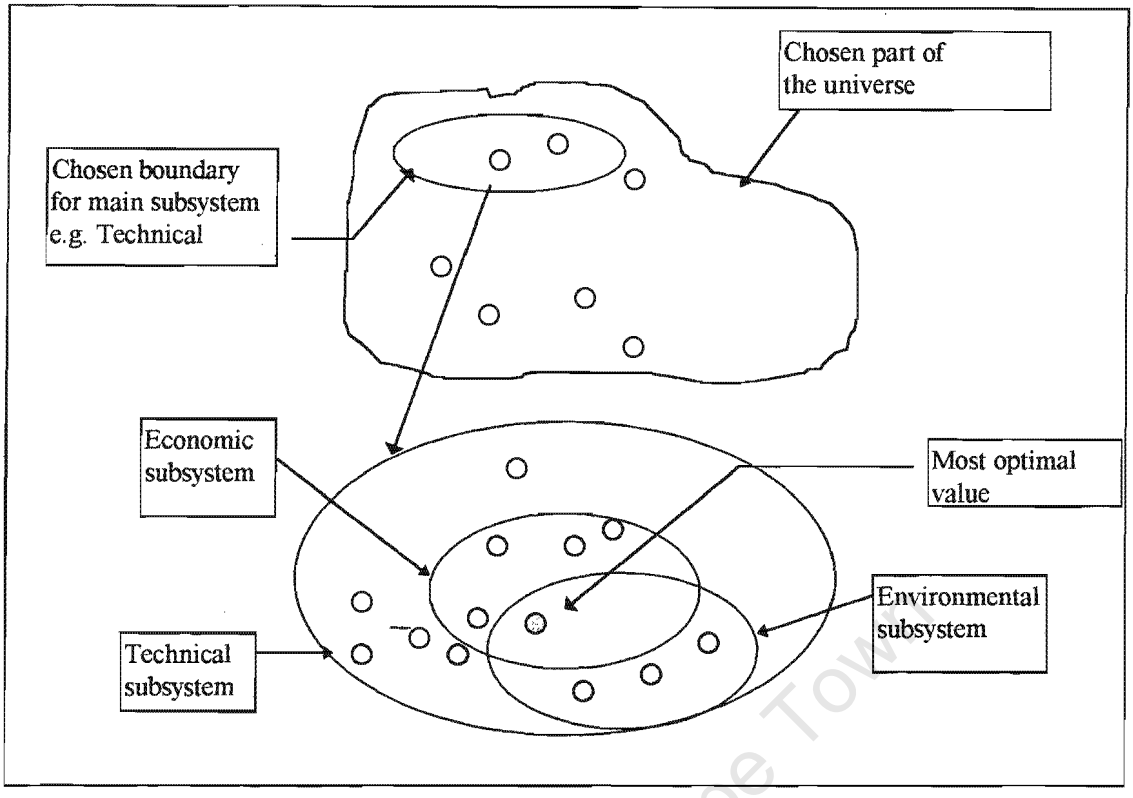
Thus, in the above example, Economic and Environmental Subsystems are classified as secondary. The selection of the sets of Secondary Subsystems specifications depends on the selection of one specification from the Technical Subsystem (Main Subsystem). However, the optimisation of the most optimal specification in each of the Secondary Subsystems is independent from the Main Subsystem.

4.2.2 Boundaries between subsystems - the method of finding the most optimal effectiveness value

The example presented in section 4.2.1 reveals the importance of a subsystem's boundaries. The Technical Subsystem has drawn the boundary in which the system can be technically effective. Within this boundary only certain economic and environmental options are feasible.

Therefore, in order to create a more general structure for systems engineering, one should realise that the creation of a boundary in one subsystem does not encircle the most optimum values of the entire system . It is the set of boundaries within this main boundary which eventually selects the most optimal value. Figure 4.2 explains the above considerations.

Figure 4.2 Selection of subsystem's boundaries



These boundaries overlap each other creating common sets of elements. The common sets embrace those elements which are related to more than one subsystem. Finally, the overlap of all subsystems results in one or more optimal values.

The boundaries of subsystems are usually drawn where there are fewest input-output connections among subsystems' components (G.C. Dandy & R.F. Warner[9]). It means that one unique component of a certain subsystem is connected to many components, each coming from different subsystems.

For example, performance of the technical subcomponent of the control system of a chemical reactor may influence economic and environmental components in a case of a fault occurrence.

The connections between subsystem's components (e.g. attributes) form a web of interrelationships. These interrelationships characterise the mission of the entire system.

4.2.3 Interrelation and connections between subsystems using the Capability attribute

In the small developing organisations, subsystems are usually connected through the Capability attribute. Usually, Capability (function of performance) reflects the objective given to the system/subsystem. Figure 4.3 illustrates the structure of interconnections between subsystems.

Figure 4.3 Interconnections between subsystems

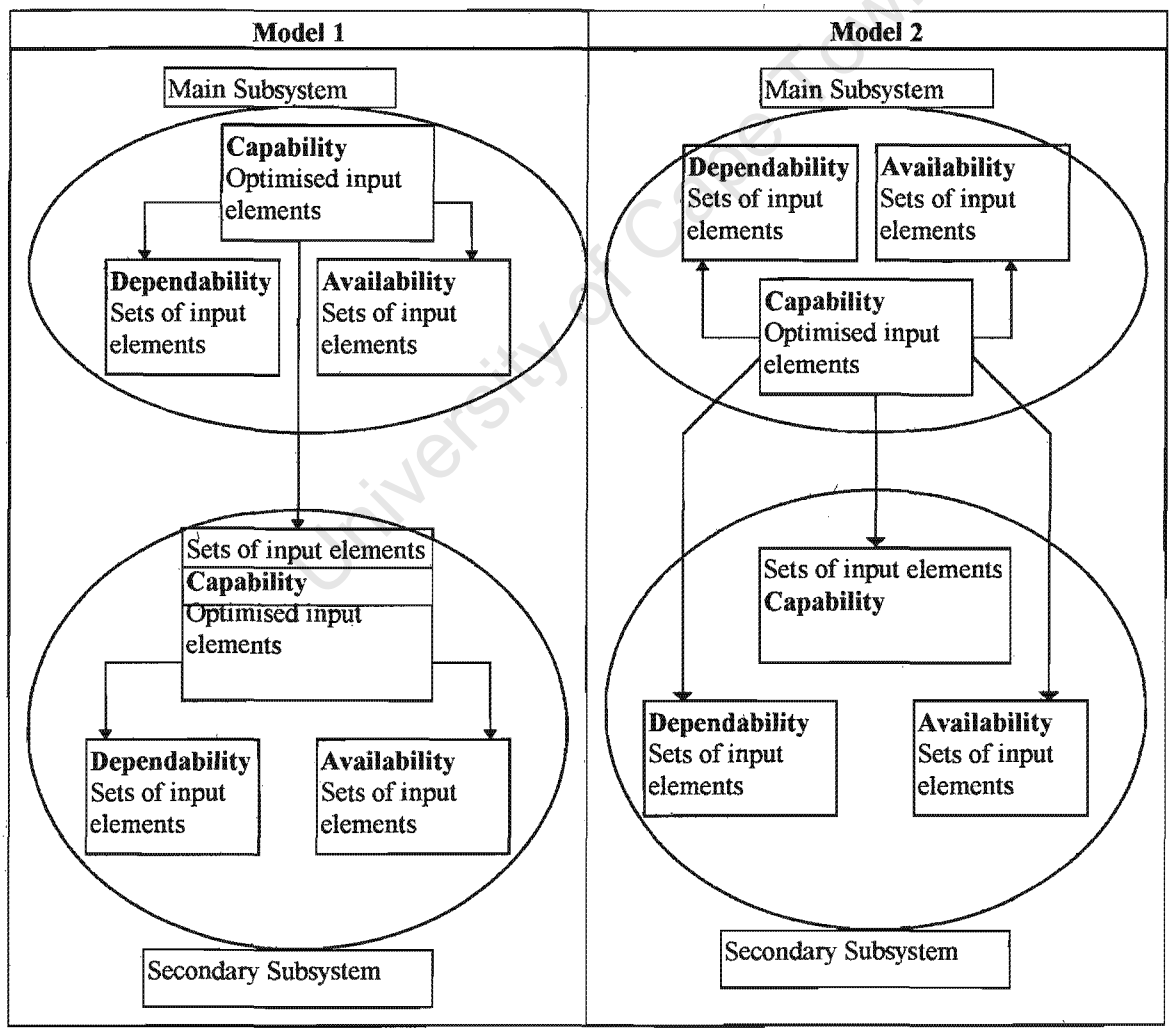


Figure 4.3 shows two types of connections between the Main and the Secondary Subsystems.

Model 1 shows that optimised input elements of Capability of the Main Subsystem select the respective sets of input elements of Capability of the Secondary Subsystem. The optimised input elements of Capability of the Main and the Secondary Subsystems select sets of input elements of Dependability and Availability attributes in the Main and the Secondary Subsystems. For example, performance (i.e. input element of Capability of the Main Subsystem) is usually considered as the element which selects suitable equipment for each operational stage of the manufacturing system. However, the choice of the suitable equipment is related to the purchase price of that equipment. Therefore, various options of repayment of the equipment are considered. The repayments are regarded as the economic performances of the Economic Subsystem (in this case the Secondary Subsystem).

Model 2 shows that input elements of Capability of the Main Subsystem select the sets of input elements of Dependability and Availability of the Main Subsystem as well as the input elements of Capability of the Secondary Subsystem. For example, the selected equipment may have a high rate of productivity, therefore, the amount of sold products as well as the revenues will increase. The increase of revenues will increase the reliability (Dependability) of repayment for the purchased equipment. Also, the higher quality of manufactured products by the selected equipment may increase the number of customers who are the source of the income. Thus, Availability of the Economic Subsystem increases. The selection of the input elements of Capability of the Secondary Subsystem may be the same as the selection presented in the example of Model 1

Model 2 is chosen to develop the computer programme because this model has the same structure as the detergent-manufacturing system which is optimised by the developed programme (see Chapter 7).

The optimisation of the attributes is independent of one another because these attributes have different purposes (See Chapter 2).

In specific situations any of the three attributes may be chosen as primary. For example, laboratory and scientific institutions may be more concerned about Dependabilities of their newly discovered products rather than the amount of outputs (amount of newly discovered products). Dependabilities, in this case, can indicate the reliability of the product/system (e.g. medicine with low possibility of side effects). In other situations, such as food supply to impoverished areas, Availability may be a primary attribute. The Availability of supplied food is then more crucial in comparison to the nutritious value of the food (Capability) or the resistance of the food to deterioration (Dependability).

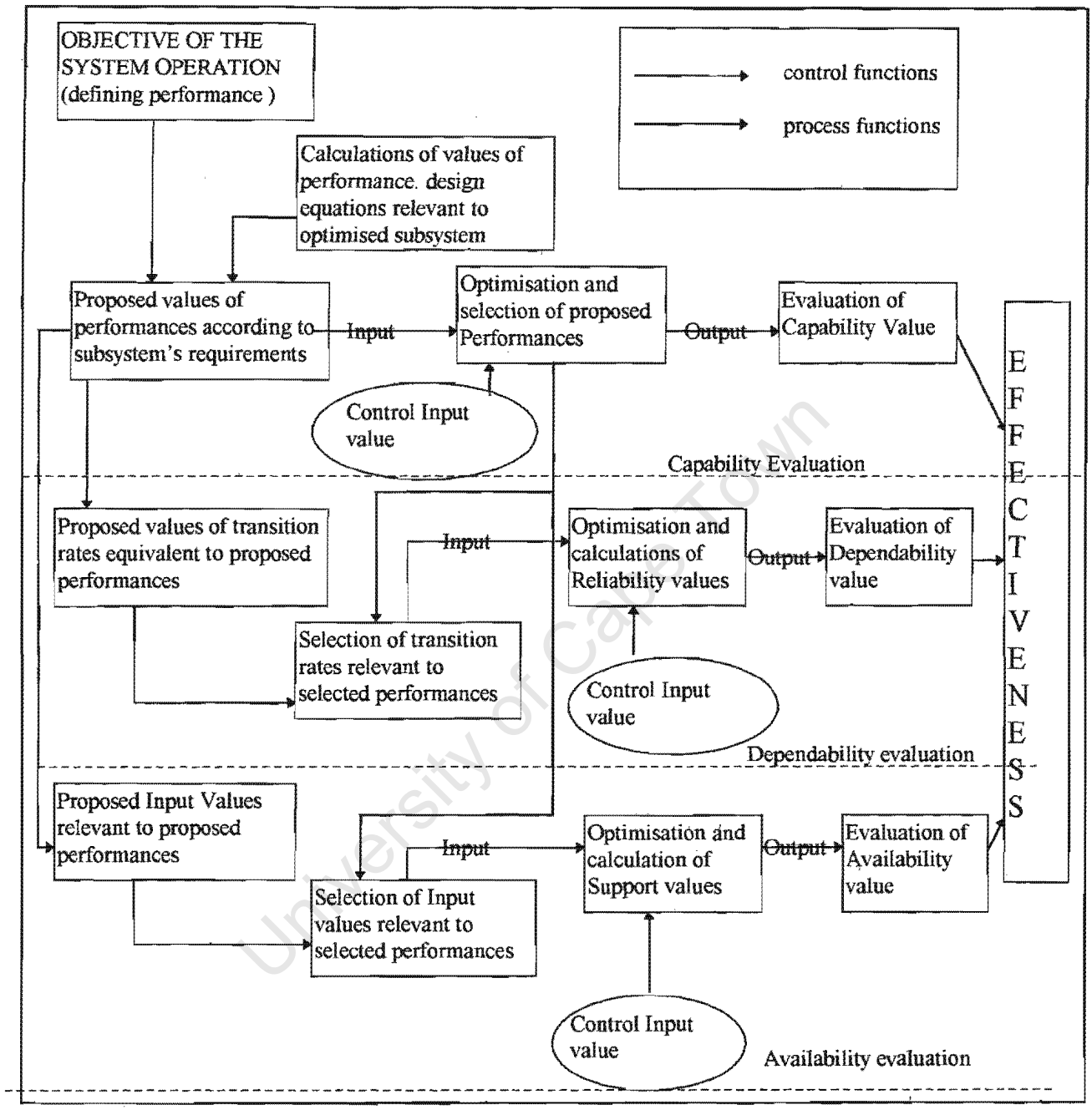
Nevertheless, in the small developing companies, it is Capability which is usually the primary attribute. The rest of the attributes are designed according to the selected Capabilities. The reason for this is that the restructure of reliability design or supportability design for Capability purposes does not require fundamental changes in the entire system's structure.

4.3 Summary of the structure proposed for modeling a small developing business

4.3.1. General model of the small developing system

Figure 4.4 presents a general model of the small developing system which is based on the findings made so far.

Figure 4.4 General optimisation model for Main Subsystem in small developing business



The above figure can also be applied to the Secondary Subsystem, however, the selection of the input elements to the Capability, Dependability and Availability attributes is dependent on the optimised performances of the Main Subsystem.

Figure 4.4 shows that the small developing business has the following characteristics:

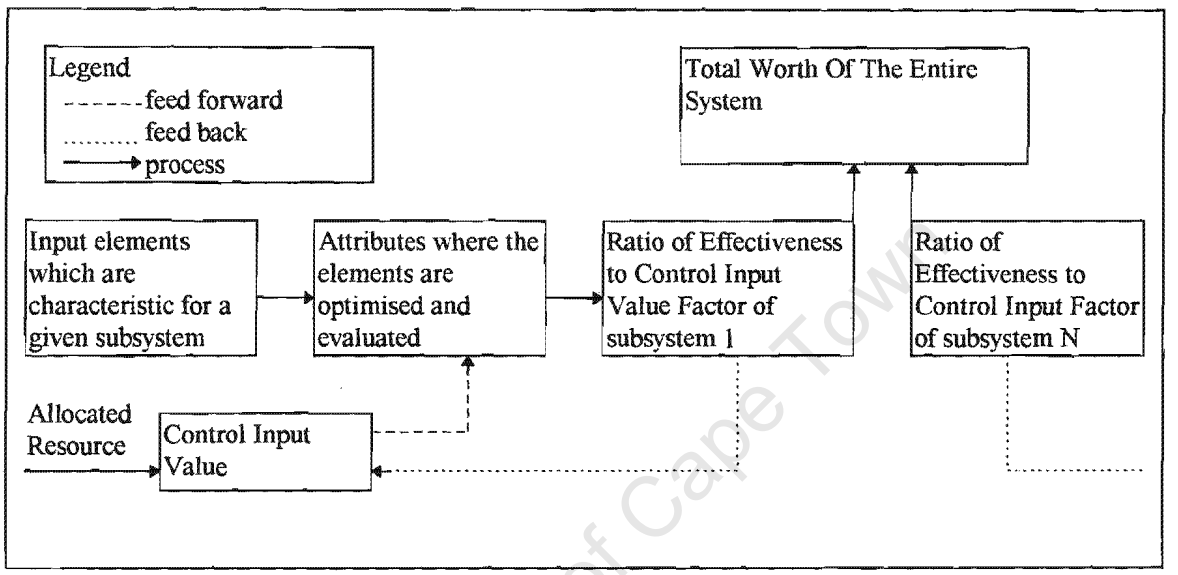
- the main objective of any subsystem is usually reflected in the performance/Capability evaluation/optimisation. Performance is the primary input element of the subsystem.
- selection of values of performances directly influences both the selection of respective transition rates for reliability evaluation and support input values for support design evaluation
- every optimisation process (for Capability, Dependability, Availability) is limited to a certain extent by the Control Input Value. The Control Input Value works like a 'valve' which allows only selected input values to undergo optimisation. The resource requirements will control the action of this 'valve'.
- The result/output of every optimisation (i.e. optimal performance, reliability, value of support) is used to evaluate the quality value of every attribute (Capability, Dependability, Availability). The three attributes combine into the effectiveness of the whole subsystem

The process of the structure shown in figure 4.4 may be repeated the same number of times as the number of subsystems which are present in the considered system. Every analysed subsystem results in one effectiveness value. Effectiveness values are combined with Control Input Values to form the ratio of Effectiveness to Control Input Value Factor. The ratio of Effectiveness to Control Input Value Factor measures the amount of the resource utilisation to achieve the required value of effectiveness. These ratios of Effectiveness to Control Input Value Factor finally combine into the Total Worth of the considered system. The methods applicable to evaluate the total worth of a system were presented in section 2.5.

It is important to note that the Control Input Value affects the optimisation of the effectiveness through the optimisation of the three attributes (Capability, Dependability, Availability). The Control Input Value also relates to the Control Input Value Factor (see subsection 4.1.2). Therefore, the Control Input Value can alter the final value of the ratio

of Effectiveness to Control Input Value Factor. Thus, if one would like to optimise the final value of any subsystem (ratio of Effectiveness to Control Input Value Factor) he/she should identify the characteristics of both the attributes and the Control Input Value for that particular subsystem (see figure 4.5 below).

Figure 4.5 General characteristics of the entire system's evaluation



The Control Input Value in figure 4.5 allocates the resources using the feed forward and feed back processes. A certain amount of the resources is allocated due to the attributes' resource requirements (feed forward). The optimised values of the attributes and the resource requirements, together form the ratio of Effectiveness to Control Input Value Factor (output of a subsystem). If the value of the ratio is too small then the Control Input Value is altered again (feed back). This allows a new allocation of resources for optimisation of new input elements through the subsystem's attributes.

4.3.2 Considerations towards the computer model

The presented theory shows a general structure of the systems engineering model which may be applied to the optimisations of the small developing business/system. The structure

of the model remains unchanged. Therefore, it is possible to write a computer programme which optimises the total worth of the small developing system.

Although the small developing system contains only three types of input elements, there is still a large amount of data to process. Thus the computational analysis becomes more of a necessity rather than a luxury. The available programmes in the form of spreadsheets allow the user to perform statistical analysis on a large amount of data. Spreadsheets are usually easy to operate, practical and relatively inexpensive. However, the optimisation tools such as Dynamic programming, Monte Carlo Simulation and Fuzzy Logic (see Chapter 3) require a specialised software. This means that it is necessary to develop a programme based on one of the popular computer languages. Therefore, the computer software needed to optimise the proposed systems engineering model will have to consist of both the spreadsheet's facilities (to process a large amount of data) and the computer language (to optimise data).

The newly developed computer language, Delphi, contains the Pascal computer language, a database and the possibility to interface with most Windows-based spreadsheets.

5. GENERAL CHARACTERISTICS OF THE DEVELOPED SOFTWARE

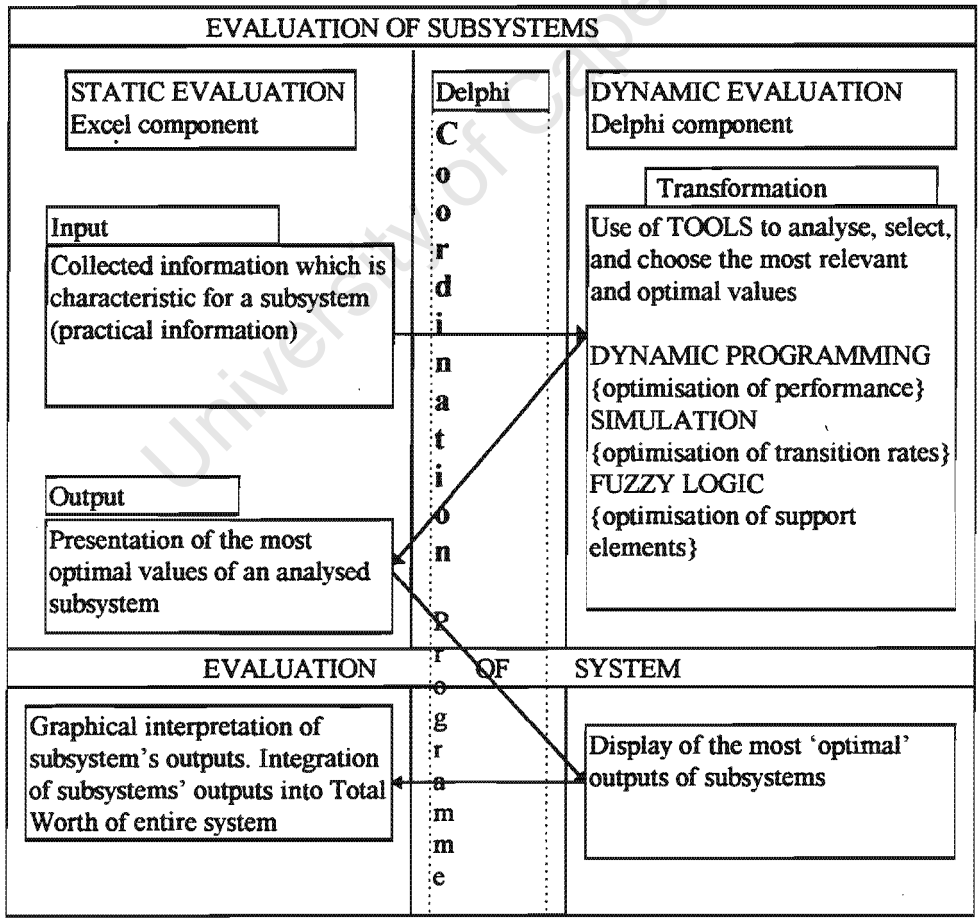
In the previous chapter, the final general structure of the systems engineering model for optimisation of the small developing business is presented. The computer model of the structure analyses and optimises the typical small developing systems. The computer programme, Systems Optimiser, is based on the Delphi computer language. The presented software operates on Windows 3.1 because Excel, as well as Delphi, requires Windows 3.1 or higher for its operation.

The entire programme is divided into two major parts:

Delphi components
Excel spreadsheet components

Figure 5.1 shows a general structure of the programme:

Figure 5.1 General structure of the programme



The description of the computer programme structure is given in Appendix A and details of the listed programme are shown in Appendix C. The software operational manual is presented in Appendix B. The present chapter will describe the structure of the programme.

The software can analyse one main and one Secondary Subsystem at a time. Based on observations related to small developing businesses, the author found that usually one subsystem is in the main focus while the rest of the subsystems are of secondary importance.

If an analysed system contains more than one Secondary Subsystem, then the optimisation procedure (using the developed software) has to be repeated as many times as the number of Secondary Subsystems which are present. Unfortunately, lack of adequate computer systems resources prevented the expansion of the computer programme to more than two subsystems.

5.1. The Excel part of the software

The Excel part of the software is used to display inputs and outputs values of the analysed subsystems. Every spreadsheet (except FGR.xls) represents the subsystem (part of the entire system under consideration) which is to be optimised. The following spreadsheets represent subsystems:

- (Main.xls) contains inputs as well as outputs values of the primary/Main Subsystem
- (Second_1.xls) also contains input and output values but for the Secondary Subsystem.

In addition to the above two Excel files, there is the FGR.xls file which displays the final, optimised values of all considered subsystems.

The two spreadsheets (main.xls and second_1.xls) have nearly the same pattern. These spreadsheets consist of four sections of which the major ones evaluate Capabilities, Dependabilities, Availabilities and the ratio of Effectiveness to Control Input Value Factor.

Every spreadsheet consists of areas where the designer can change values (light blue sections) and those areas where values are calculated by the Excel programme (yellow sections).

5.1.1 Main Subsystem (Main.xls file)

The purpose of this file is to collect all the input information needed for the optimisation and display of the optimised values (input values are optimised on the Delphi component of the software). The optimised values are further converted into qualitative values of the subsystem's three attributes (Capability, Dependability, Availability). Finally, values of these three attributes are integrated into ratios of Effectiveness to Control Input Value Factor of the whole subsystem.

(a) Capabilities Section

The main objectives of this subsection is to collect all the relevant data for performance evaluation; and collect all the optimised performance values to transfer them to qualitative values of Capabilities. The above will be accomplished through the following subsections:

- Control Panel for Capabilities Calculation
- Table of Main Subsystem's Performances
- Critical Path
- Table of Capabilities of The Main Subsystem

(i) Controlling variables of Capability optimisation (Control Panel)

This section starts with information about control variables.
The **Control Panel** contains variables which control dynamic programming optimisation. The values which have to be fixed by the designer are : *Control Input Value*, *Step of Input*, *Minimum Total Investment*.

It has been mentioned in Chapter 4 that optimisation of any attribute is controlled by the *Control Input Value*. In this section, Control Input Value works as a 'valve' which increases or decreases the value of resources needed to select the most optimal values of performances. The *'Control Input Value'* is the maximum value of the resource (total investment) allocated for the considered subsystem. This value may be a sum of Control Input Values of Dependability, Availability and Capability. However, if units of the *Control Input Values* are not all the same, then usually the *Control Input Value* of the Capability attribute is used to evaluate the *ratio of Effectiveness to Control Input Value Factor* of the subsystem.

The significance of the *Control Input Value* is to match the resource requirement of optimal performances. The *Control Input Value* is divided into small fractions (steps). These steps, which represent potentially allocated resources, are compared with every resource requirement of every performance in a particular stage. The step value is called *Step of Input*.

The *Step of Input* is the interval of the total investment which is used when dynamic programming calculations are in progress. Obviously, the smaller the interval, the greater is the possibility of finding more critical paths (sets of optimised performances) but at the same time, the computational effort will increase. The *Step of Input* should be the value (the interval of investments) which is a factor of every proposed investment value. However, sometimes it will be difficult to find (e.g. by inspection) the correct factor of every total investment. If this is the case (e.g. if the total investment i.e. the *Control Input*

Value value is an integer), then the best Step of Input would be 1. In the case of a non-integer value (e.g. 10.5) the Step of Input could be 0.1.

Minimum Total Investment is the value of the minimum total resource which could be allocated to optimise performances. This value is needed to evaluate the *Control Input Value Factor* which is further used to evaluate *ratio of Effectiveness to Control Input Value Factor*.

The rest of the information in the 'yellow areas' of the Control Panel is for the designer's information.

The *Number of columns* (number of Investments and Revenues together) and the *Number of rows* (number of proposals) inform the user of the size of the dynamic programming. The dynamic programming is one dimensional (Number of columns/2 = number of stages). Thus, eight columns would mean that four stages are present where four investments and subsequently, four revenues are to be selected.
The *Number of rows* shows the number of proposals (i.e. proposals of Investments and Revenues) for different performances for each respective stage.
The *Initial value* is the value which initiates the allocation of investments to the respective stages. It is the first value of the accumulated *Step Of Input* value (see Appendix A ' Step Of Input').
The *Number of stages* is the total number of stages in the given system.

(ii) Inserting input values for Capabilities evaluation (Table of Main Subsystem's Performances)

The values of performances and their respective resource requirements are collected for further optimisation. These values are stored in the **'Table of Main Subsystem's Performances'**. In this table, values of performances of the subsystem are called Revenues and values of resources are called Investments.

The maximum available number of proposals of performances and respective revenues can be five (see explanation in Appendix A). The maximum number of available stages is also five. However, this can be easily changed in the programme (see computer code in Appendix D).

The optimised values of performances and their resources (Delphi component) are transferred back to Excel to the 'Critical Path' table.

(iii) Results of the dynamic programming (Critical Path)

The **'Critical Path'** table shows the proposed optimised values of Investments, Revenues and names of the Proposals under which optimised values are selected. Therefore, the user can easily identify the 'routes' which were taken during the optimisation. Every row of the chosen Investment and Revenues is called an option. An option is the number of a critical path which was created during selection of the optimised values.

(iv) Outputs of the Capability evaluation (Table of Capabilities of The Main Subsystem)

Once all the optimised values of performance have been collected in the 'Critical Path table, they can be further converted to qualitative Capability values. The **'Table of Capabilities'** converts the optimised values of Revenues (performances) into relevant quality values of Capabilities. The table is marked in light blue because the formula used to calculate Capabilities is dependent on the designer's standards.

In the table, there is one row where the ideal performance value for each stage can be inserted. This row is called *Ideal Performance Value*. Beyond the space reserved for ideal performances, under the headings for stages, the user can insert the formula which can be used to evaluate Capabilities' values. The name 'Option' in the yellow area informs the

user of the number of the critical path (of the dynamic programming) which corresponds to the evaluated Capabilities.

(b) Dependabilities Section

The objective of this section is to collect the transition rates which can be further simulated and averaged to evaluate reliability and finally, Dependability values. The evaluation of reliabilities is done in this section using relevant probability functions. These functions transfer transition rates into probabilities. The probabilities of reliabilities are (the same as values of performances) limited by the Control Input Value which, in this case, is represented by time of transition occurrence. In this evaluation, values of reliabilities are exactly the same as values of Dependabilities.

The above is achieved by the following sections:

- Table of Proposed Transition Rates
- Table of Selected Transition Rates
- Average Values of Transition Rates
- Table of Dependabilities

(i) Inserting input values (Table of Proposed Transition Rates)

The Table Of Proposed Transition Rates is used to enter the transition rates which are collected from the observations. Before the values of transitions are entered, the designer must distinguish between the stages in which values of Dependabilities will be calculated directly for the collected data (independent) and those stages which will be evaluated by the formula (dependent). For example, if one is dealing with a three stage system, then every stage would have a maximum of three possible transitions or Dependabilities. It is only required to know two transition rates because the third will be calculated by the Excel spreadsheet.

The sets of transition rates, which are collected in the table (Table of Proposed Transition Rates), must be selected before these sets are optimised. The selection would depend on the selected values of Revenues (performances) and Investments (resources) in the Capability section.

(ii) Selection of the input values (Table of Selected Transition Rates)

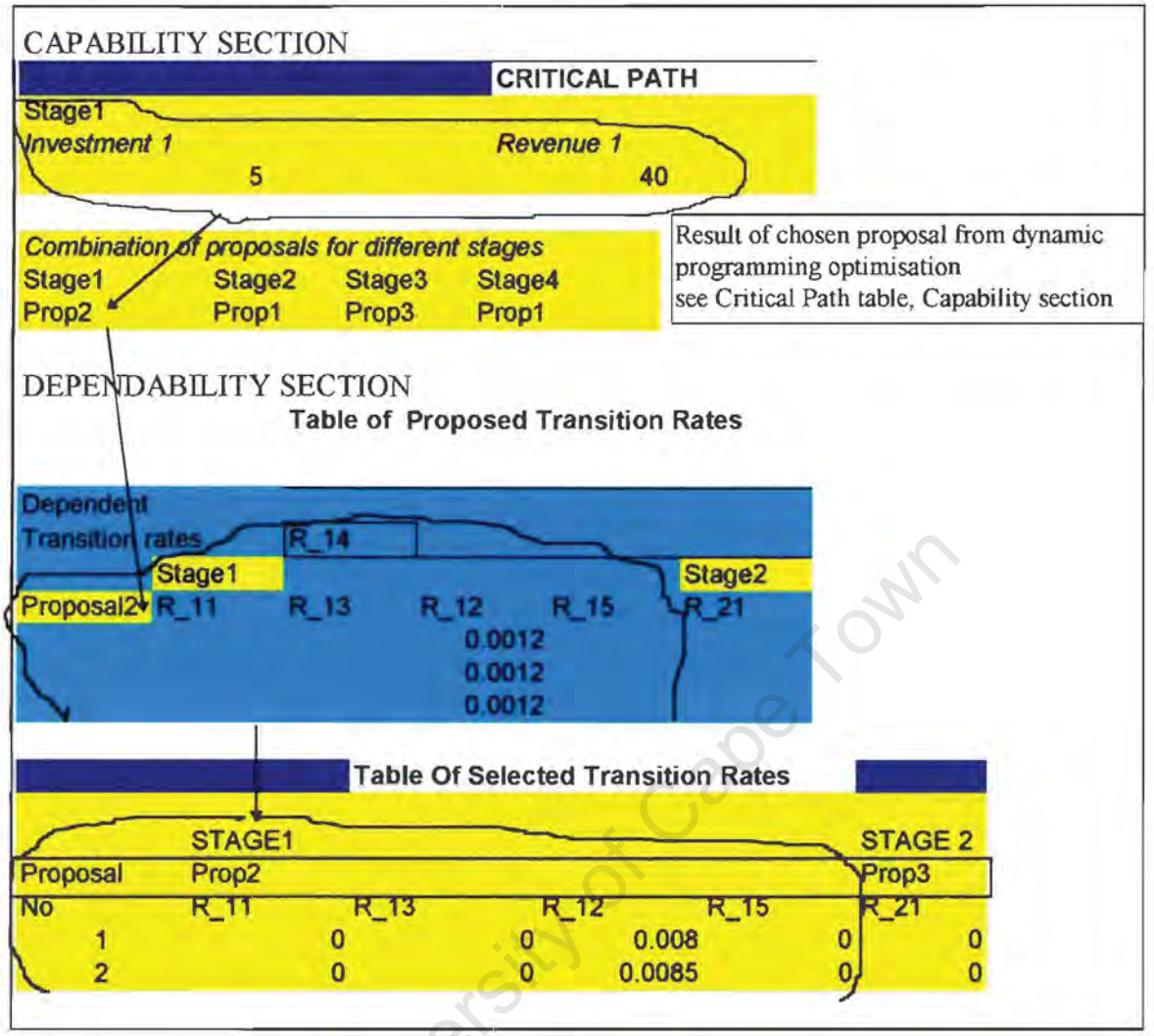
The Table Of Selected Transition Rates contains values of groups of transition rates which are selected through the choice of performance values (see Chapter 4). This table is connected with the Table of Proposed Transition Rates.

Under the heading 'Stage', there is the name of the proposal which is changed in the same way as the proposal of the performance in dynamic programming is changed (see Combination of proposals for different stages in Critical Path table of Capability section in figure 5.2).

Depending on the proposal and the stage in which the performance value is chosen during dynamic programming optimisation, the respective set of transition rates (independent set of transition rates) is selected. The procedure is shown in figure 5.2.

For example, if Revenue and Investments of Stage1 and Proposal 2 were selected (Capability Section), then all transition rates representing transitions from Stage 1 to other stages of the system are selected. These transitions must come from the rows which are marked as Proposal 2. Figure 5.2 explains the above considerations.

Figure 5.2 Selection of Transition rates through the selected performances values



Selected values of transition rates are transferred to the Delphi module, where they are averaged through Monte Carlo Simulation. Then the results of the simulation are entered into the Excel spreadsheet again.

(iii) Results of Monte Carlo Simulation (Average Values of Transition Rates)

The **Average Values of Transition Rates** table contains outputs of simulated average values of transition rates. These average values are further transferred to reliability functions.

The heading of every row of the table has a name 'Option'. Every option indicates the number of the critical path of performances which is related to its respective transition rates. For example, for the three stage system, the first Critical Path (Option1) in the *Critical Path* table (Capability Section) would have optimised revenues (performances) for Stage 1, Stage 2, Stage 3. In the *Average Values of Transition Rates* table, under the heading of the first row (Option1), one would find the first three optimised transition rates (i.e. R_11, R_12, R_13). These transition rates would characterise the performance value of Stage 1. The same applies to the other six transition rates.

(iv) Output and controlling variable of Dependability evaluation (Table of Dependabilities)

The average values of transition rates are transferred to the **'Table of Dependabilities'.** In this table, average values of transition rates are transformed into probabilities. Every probability represents reliability (as well as Dependability) of the transition from one stage to another.

The extent to which reliability can occur is limited by the *Control Input Value*. The *Control Input Value* is represented by the time when all transitions occur. The time must be inserted before the simulation begins. The reliability directly represents the qualitative value of Dependability. The function for evaluation of Dependabilities is of an exponential type (i.e. $e^{x \cdot t}$, see Subsection 2.2.1). In the case where the state model has a different nature, the functions in the area of cells C121 to A131 must be changed.

(c) Availability Section

In this section, subjectively evaluated input elements of the support design are transformed into one unified output value. The output value represents the strength of the support design of a particular stage.

The input elements of Availability are optimised in the Delphi component (Fuzzy Logic). The following subsections model Availability:

- Table of Proposed Inputs
- Table of Selected Inputs
- Table of Outputs and Availabilities
- Table of Firing Rule
- Table of the Magnitude of Inputs and Output

(i) Inserting the input values (Table of Proposed Inputs)

The support design values, contrary to the values of performance and reliability design, are based on the subjective judgements of an observer. These values are collected in the **Table of Proposed Inputs**. As it was mentioned in Chapter 2, the environment of a subsystem may be internal as well as external. Therefore, Inputs A&B (see Chapter 3 on Fuzzy Logic) may be used as input values of the internal and external environment. The designer enters Inputs into *Table of Proposed Inputs*. Similarly, as for transition rates, the inputs are grouped into stages. The grouped inputs are related to the respective grouped values of performances in the Capability section.

For example, the Input A and Input B of the Availability value under heading STAGE 1 in the table, refers to both the Revenue (performance) and Investment (resource) value of STAGE 1 in the 'Table of Main Subsystem's Performances'. The name 'Proposal' with its respective number is related to the number of proposal of both Revenues (performance) and Investments in the 'Table of Main Subsystem's Performances'.

(ii) Selection of the inputs (Table of Proposed Inputs)

The value of inputs of the Table of Proposed Inputs is selected according to patterns of critical paths where respective values of performances and resources are chosen. The

results of the selection are shown in the '**Table of Selected Inputs'**. In this table, the selection of the Inputs A & B takes place. The selection of values of Revenue and Investment (*Table of Main Subsystem's Performances'*) influences the selection of values for Inputs A & B, which are chosen in the *Table of Proposed Inputs* (see explanation in the *'Table of Proposed Inputs'* subsection (i)).

The selected values of inputs are transferred to the Delphi programme (Fuzzy Logic module) to be transformed into one output. The value of the output is the result of the combined strengths of input values. The magnitude of the output value in relation to input values depends on the Fuzzy Rule arrangement (Control Input Value of Availabilities).

(iii) Outputs of the Fuzzy Logic (Table of Outputs and Availabilities)

Finally, outputs are displayed in the **Table of Outputs and Availabilities**. These values are the results of Fuzzy Logic calculations. All of the calculations are influenced by the given/chosen Input A & B values.

On the right-hand side of the table, the relevant quality values of Availabilities are given. These quality values are calculated by dividing every value of Availability (Availability value which belongs to a certain stage) by the total sum of all Availability values of the one optimised option (option refers to the optimised set of Availability values).

(iv) The variables controlling the Availability optimisation (Table of Fuzzy Rule & Table of the Magnitude of Inputs and Output)

The *Control Input Value* of Availabilities controls all of the Inputs which are combined into respective values of outputs. This *Control Input Value* is called a Fuzzy Rule and is placed in the '**Table of Fuzzy Rule'.**

(d) Final results

This section combines all the results from Capability, Dependability and Availability sections. The final result is the ratio of Effectiveness to Control Input Value Factor.

The way the Capabilities, Availabilities and Dependabilities are combined in the ratio of Effectiveness to Control Input Value Factor, is shown in subsection 2.3.1. The results of these ratios are displayed in the ratio of Effectiveness to Control Input Value Factor row of the Final Results section (this row starts from cell D275).

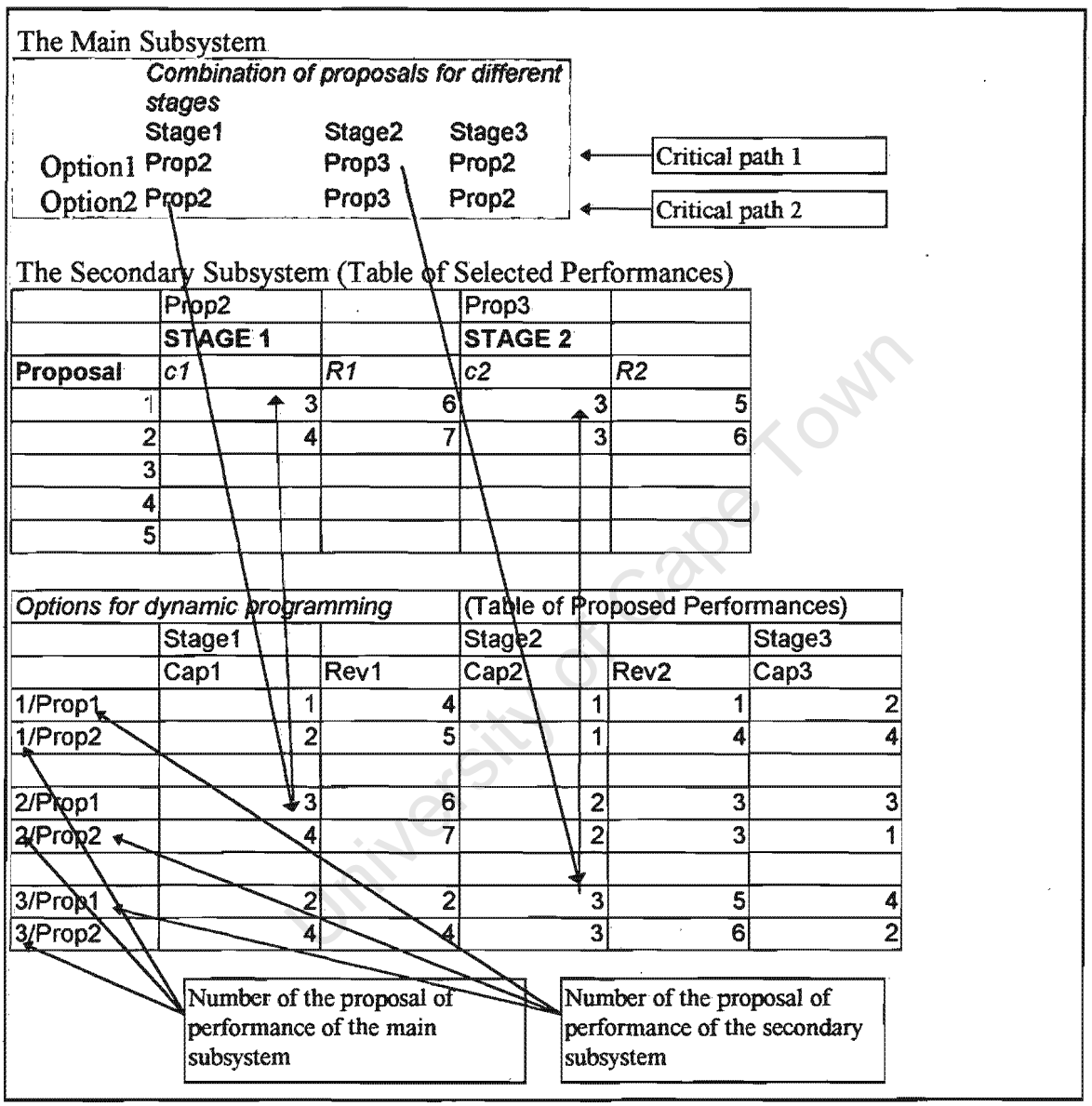
5.1.2 Secondary Subsystem

The Secondary Subsystem depends on the Main Subsystem. As it was previously mentioned, the connection between both subsystems are assumed through selected values of Revenues (performance values) and Investments (respective resources of performances) in the Main Subsystem. Therefore, if in the Main Subsystem the value of the performance and its resource, characteristic for Stage 1 and proposal 2, is selected (as the result of dynamic programming optimisation) then these values select the respective values of performance and resource of the Secondary Subsystem (dynamic programming optimisation of Secondary Subsystem). To every value of performance characteristic of the Main Subsystem, there are a maximum of two values of performances of the Secondary Subsystem.

It was mentioned earlier that the optimal path connects optimised values of performances (i.e. Critical paths). The sum of all the optimised performances which belong to one critical path gives the total value of performance. However, there can be more than one way (more than one critical path) which may lead to the same optimal value of total performance. Therefore, more than one set of optimal performances (Critical Paths) may be created and more than one set of optimal Capability values evaluated. This applies to optimisation of main and secondary performances. Therefore, for each critical path found

in the Main Subsystem there is more than one critical path in the Secondary Subsystem. Figure 5.4 illustrates the above considerations.

Figure 5.4 Relationship between the Main and Secondary Subsystems' performance values (Capability section)



The values of 'Revenues' and 'Investments' are selected, by dynamic programming of the Secondary Subsystem, in the first table of the Secondary Subsystem (see figure 5.4). The selected values of 'Revenues' and 'Investments' are inserted into the '*Critical Path*' table of the Secondary Subsystem (in the same way as in the Main Subsystem).

The procedure to evaluate Dependabilities and Availabilities is exactly the same as the one shown above (see Main Subsystem). It means that the optimised performances of the Main Subsystem selects the input elements of Dependabilities and Availabilties of the Secondary Subsystem.

5.1.3. Final display of results

The final display of the results of all ratios of Effectiveness to Control Input Value Factors (the ratios for the Main and Secondary Subsystems) is shown in FGR.xls spreadsheet file. Every ratio of the Main Subsystem may have more than one dependent ratio of Effectiveness to Control Input Value Factor of the Secondary Subsystem. The relationship between the ratios of Effectiveness to Control Input Value Factor of the Primary and Secondary Subsystem is the result of interconnections between the performance values of the Main and Secondary Subsystems (see Subsection 4.2.3 and Subsection 5.1.2). The author mentioned that for every critical path of the Main Subsystem there is more than one respective critical path of the Secondary Subsystem (see section 5.1.1(a)). Therefore, for every ratio of Effectiveness to Control Input Value Factor of the Main Subsystem there is more than one related ratio of Effectiveness to Control Input Value Factor of the Secondary Subsystem. Excel graphs are a helpful tool to display the relationship between the Main and Secondary Subsystems' ratios.

5.2. Delphi Programme

The main objective of this part of the software is to optimise the previously selected input elements to a subsystem. The boundary in which input elements are optimised depends on the designer's (or system's) requirements. These requirements are implemented through the control variables of the Delphi modules (see Control Panels in every Delphi Module) and those variables which can be changed by the user.

There are four modules in the Delphi component:

- System Optimisation module which co-ordinates the exchange of information between Excel and Delphi components
- Dynamic Programming module which optimises performances for Capabilities evaluation (Capability module)
- Monte Carlo Simulation module which optimises transition rates for Dependabilities evaluation (Dependability module)
- Fuzzy Logic module which optimises support input values for Availabilities evaluation (Availability module)

Therefore, except for the co-ordination module, three Delphi modules work as internal functions of the subsystem's three attributes (Capability, Dependability, Availability).

5.2.1 Dynamic programming (Capability module)

This module optimises the values of performances and their respective values of resources. Usually, there is more than one set of optimal performances, therefore these sets of performances are called optimal or critical paths (i.e. in every stage of operation there can be more than one optimal performance - see explanation in Section 3.2).

There are five main features of the module:

- 'Table of values'
- 'First Stage' table
- 'Subsequent Stages' table
- 'Critical Path' table
- 'Control Panel'

The proposed values of performances and resources are transferred from the Excel files to the 'Table of values'. In the table, various proposals of performances and their respective resources are grouped into the stages of the considered system. Therefore, every stage may contain more than one performance (state).

When the optimisation commences, stage one is optimised. The optimised performance values of stage one are shown in the 'First Stage' table in the column 'Optimum'. According to the dynamic programming methodology, every stage is given a set of proposed resource values (these are placed in the column 'Allocated input'). The proposed values of resources are checked against the requirement of the resource of every performance of the stage.

The performance values of the rest of the stages are shown in the 'Subsequent Stages' table. This table looks the same as the 'First Stage' table.

Finally, the optimal values of performances and their respective resources are displayed in the 'Critical Path' table. This table displays a maximum of five sets of optimal performances (computational problems explained in Appendix A did not allow for more than five).

The control of the optimisation process is, to a great extent, influenced by the control variables which are present in the 'Control Panel'. The user can change *Control Input Value* and *Step of Input* value which are described in detail in Section 5.1.1(a). The green box (*Optimise)* informs the user about the type of the optimisation (i.e. maximisation or minimisation).

The rest of the features inform the designer about the size of the optimisation process and are not to be altered. *'The number of columns'* informs the user about the size of the table and also about the number of stages (i.e. number of stages is equal to number of columns divided by 2). The number of proposed performances and their respective revenues in each stage can be obtained from the *'number of rows'*.

5.2.2 Monte Carlo simulation (Dependability module)

The objective of this module is to simulate and to average values of transition rates for every transition which can take place between stages of the analysed system.

The 'Table of Transition Rates' shows the rates which are used for simulation. These values were previously transferred from Excel files. The values are changed according to the changes in the Table of Proposed Transition Rates in the Excel spreadsheets.

The 'Table of Average Transition Rates' contains averaged values of transition rates after the simulation process. The table also contains deviations of the last five values of average transitions. The average values and standard deviations are in the rows indicated by "Ave" and "Dev" respectively.

The standard deviation is used to see how far the last simulated value oscillates from the rest of the averaged values (these average values represent the same transition). If the oscillation is small, the simulated values of transition rates are most likely to be accepted.

In the 'Table of Transition Rates' and the 'Table of Average Transition Rates' every column has a code which represents the transition from one stage to another (e.g. R_12 represents the transition from stage 1 to stage 2).

The control of the simulation process is possible by the controlling variables which are present in the 'Control Panel'. In the Control Panel the user is required to insert the correct *'Control Input Variable'* which represents the time for transitions between all stages of a system, and *'Trial'* value which represents the number of simulations required to give the best average values of transition rates.

The other information (values which can not be changed by the user) includes the name of a subsystem under consideration, the *Number of events* (sample size of transition rates) and the *Number of transitions*. The user is only requested to input both the number of trials needed for simulation and the value for the Control Input Value (in most cases, this will be the time).

5.2.3 Fuzzy logic (Availability module)

This module optimises subjectively evaluated values of input support elements into one unified value of output. The output represents the combined strength (subjective value) of the support design for a stage.

The programme allows the user to enter two input elements (Input A and Input B). The two inputs as well as one output element are interpreted, according to Fuzzy Logic theory, (see Section 3.3) as triangular graphs. For the simplicity of modeling, these graphs have the geometry of isosceles triangles. Three vertices of the triangles represent the 100% value as being Small, Medium or Large. Therefore, three triangles were used to interpret input and output values. Since the triangles are isosceles, so only one top value (Target value) and one space value (Space) are required for the programme to build three triangular graphs for every input and output value (see Subsection 5.1.1(c)).

The input values are automatically transformed from the Excel files through the Fuzzy Rules into the output. The results of the outputs (one output per stage) are displayed in the *'Output of Fuzzy Rule'* window (see Manual for Systems Optimiser Software).

5.2.4 Co-ordination module

This module has been created to co-ordinate the flow of data between the Excel files and Delphi modules. Therefore, this module activates the Capability, Dependability and Availability modules where the subsystem's inputs elements are optimised. When the optimisation is finished, these optimised elements are transferred to Excel spreadsheets, which convert these elements into qualitative values of Capabilities, Availabilities and Dependabilities. Depending on the Excel file (Main.xls file evaluates values for Main Subsystem, Second1.xls file evaluates values for Secondary Subsystem) the Capabilities, Dependabilities and Availabilities are integrated into the Effectiveness factor which finally is incorporated into the ratio of Effectiveness to Control Input Value Factor.

The ratios of the Main Subsystem will be displayed in the 'Main Sub' table and the ratios of the Secondary Subsystems are in the 'Second1 Sub' table. These ratios are finally transferred to the FGR.xls file for further graphical interpretations (see section 5.1.3).

Every single ratio of Effectiveness to Control Input Value Factor of the Main Subsystem may have more than one ratio of Effectiveness to Control Input Value Factor of the Secondary Subsystem. This is because the one critical path of the Main Subsystem may contain more than one critical path in the Secondary Subsystem.

6. GENERAL CHARACTERISTICS OF INTEGRATION OF THE THEORY AND COMPUTER SOFTWARE INTO THE MODELING STRUCTURE

The suggested theory as well as the developed computer software will need to be integrated into the structure, which characterises a small developing business.

The role of a systems designer in a small developing business is to identify the system's objectives, the system's structure , the subsystems' attributes and the subsystems' input elements.

The computer programme is expected to be the tool which should help the designer to find the most optimal system's configuration. In this case, the software's task is to optimise and transform, quickly and efficiently, the subsystem's input elements into the respective qualitative values of the subsystem's output using Control Input Values. The subsystem's outputs are finally combined into the total worth of a considered system (i.e. the qualitative value of the system's objective).

6.1 Recursive structure in a systems engineering model

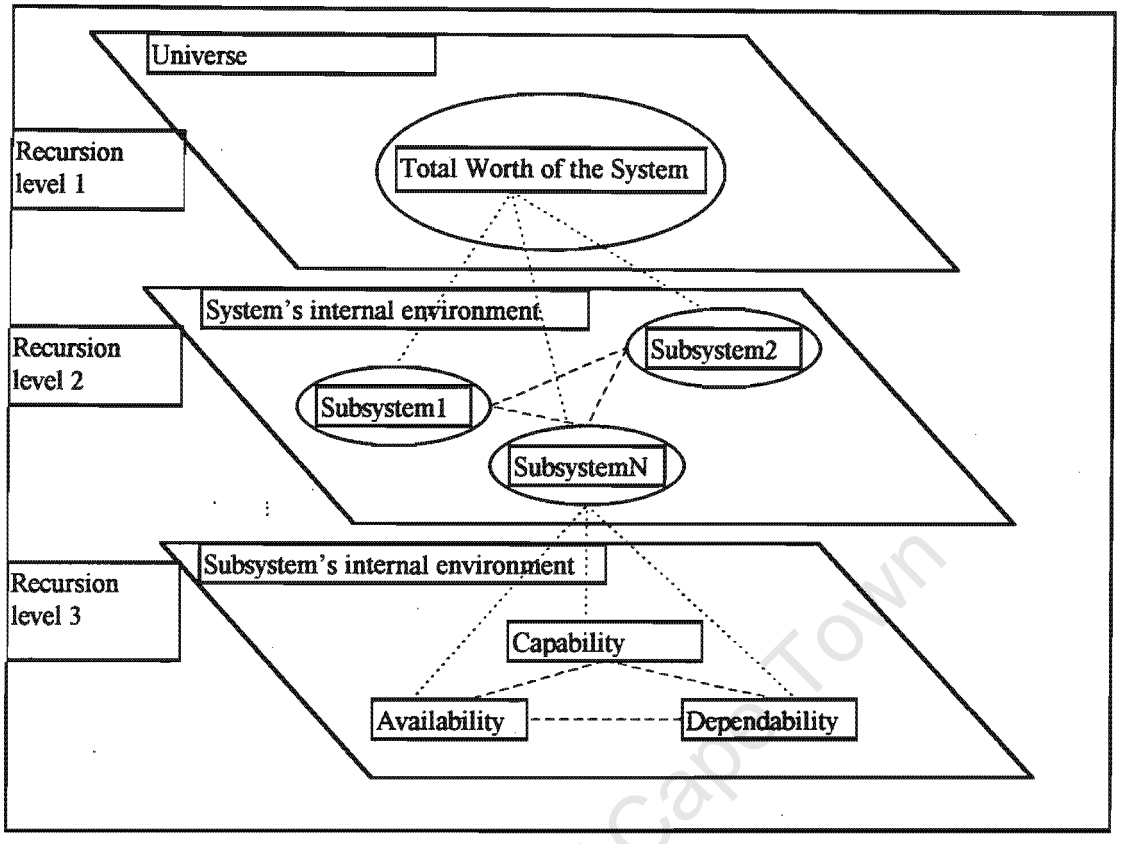
The designer should always start to identify the system's nature, its subsystems and components of subsystems before any modeling procedure can take place. Because the systems engineering model has a hierarchical structure, it is suggested to use a three stage recursive structure which can describe the general structure of the analysed system.

The three stage recursive structure will focus the analysis on the following three main areas:

- identification of objectives of the system in focus - recursion level 1;
- identification of the subsystems and their relationships - recursion level 2;
- identification of the subsystem's input elements and attributes - recursion level 3.

The above general recursion model is shown in figure 6.1.

Figure 6.1 The general recursive level structure



6.1.1 Recursion level 1

At the Recursion level 1, one must identify the nature and objectives (i.e. purpose) of the considered system. The developing business often has more than one objective. There is one main objective which characterises the system's main direction of development, and the secondary objectives which support the main objective and (the same as the main objective) the development of the entire system.

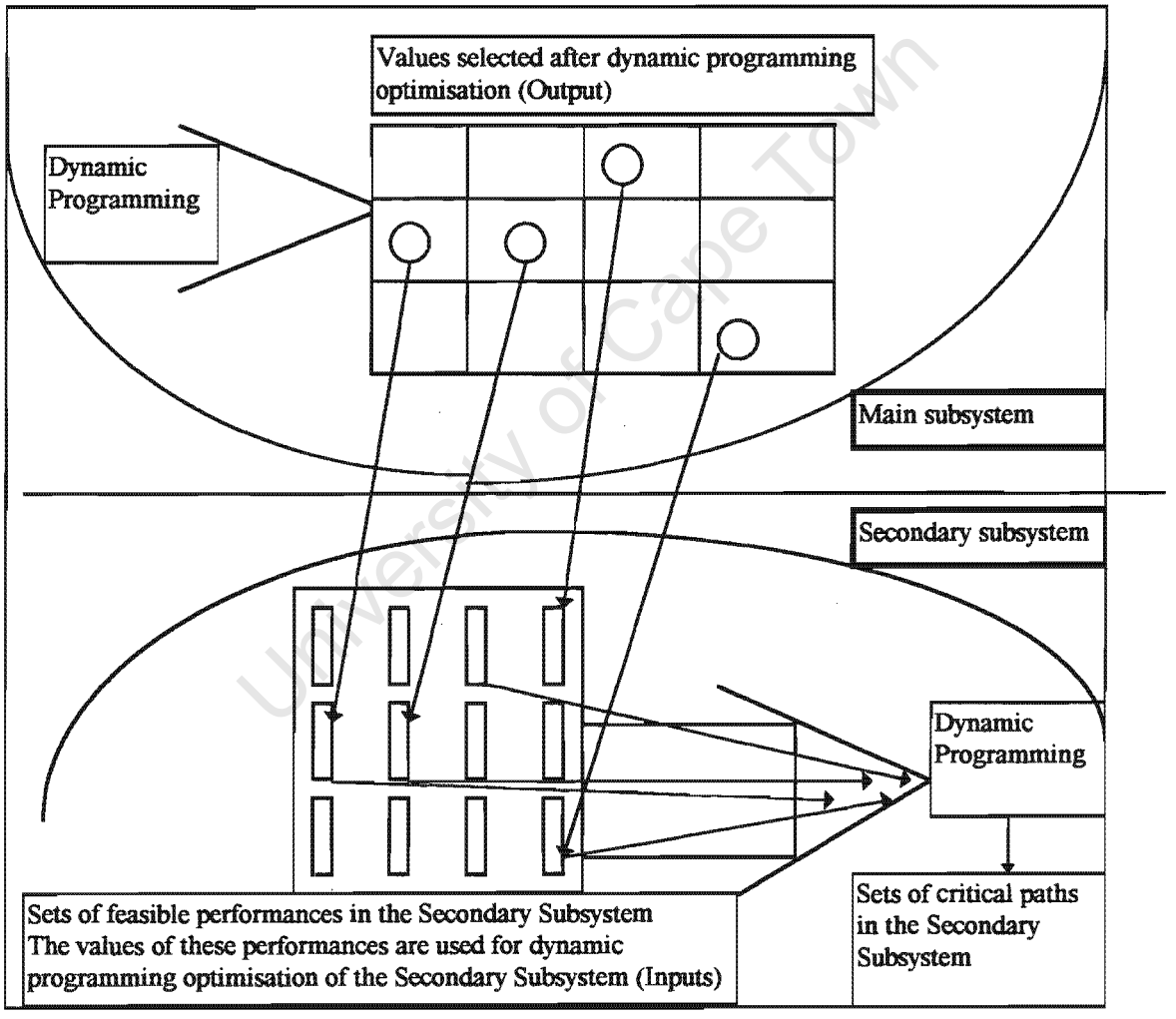
6.1.2 Recursion level 2

The Recursion level 2 considers the system's internal environment. The designer has to identify subsystems and the interrelationships between them. The number and type of subsystems are related to the number and nature of the system's objectives. The

interrelationship between the subsystems depends on the hierarchy between the system's objectives. Thus, the main objective may influence the formulation of the secondary objectives (see Section 4.2).

In the small developing organisations, subsystems are interconnected through the Capability attribute. The connections between the attributes of different subsystems are designed on Excel (these connections do not change - see Chapter 4). Figure 6.2. shows schematically the connections between subsystems.

Figure 6.2 Suggested connections between subsystems

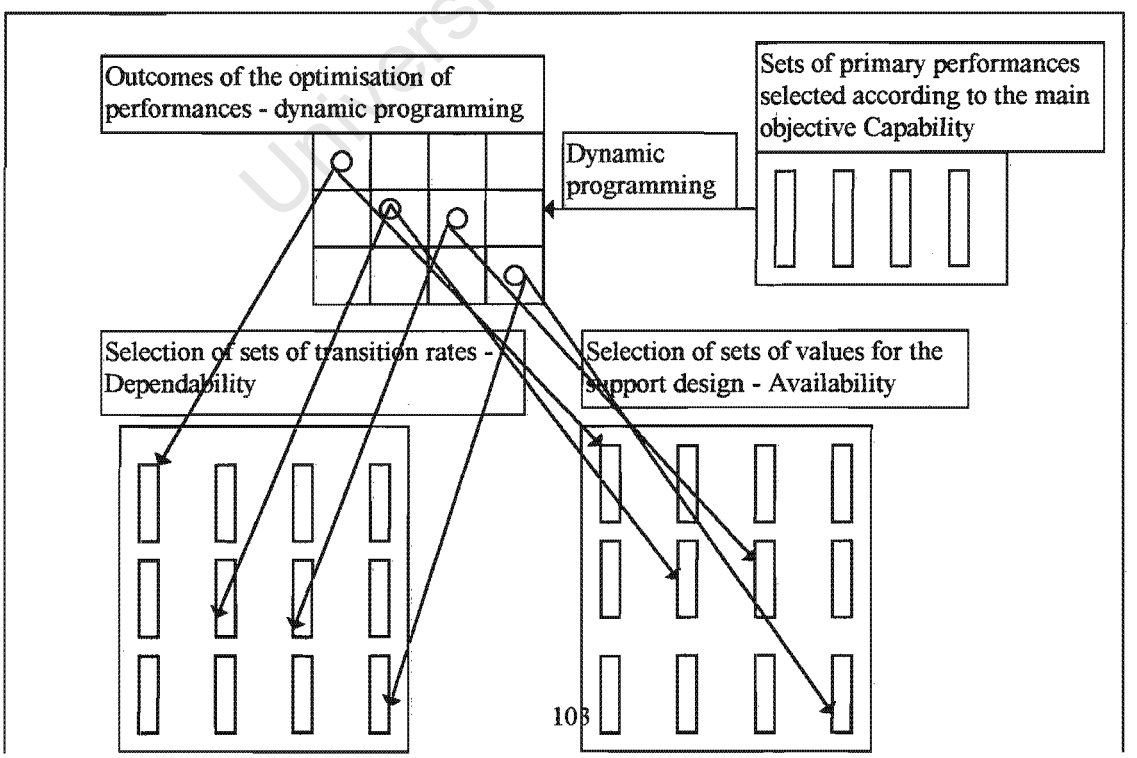


6.1.3 Recursion level 3

The Recursion level 3 deals with the identification of the subsystems' attributes and input elements.

Each subsystem has a maximum of three attributes, namely Capability, Dependability and Availability. The input elements to the subsystem (inputs to respective attributes) are performance, reliability and support. The interrelationship between the subsystem's input elements is arranged in the hierarchy where Capability (function of performance) is the primary attribute while the Availability (function of support elements) and Dependability (function of reliability) are the secondary attributes (see Section 4.2). The designer collects the input elements and sorts them into sets of values corresponding to the three above mentioned attributes. The most optimal input elements are selected in the two phase process. The best optimal performance values are selected by the designer. The most optimal sets of transition rates (for reliability evaluation) and support elements are selected through the dynamic programming process. Schematically the connections between the subsystem's elements are shown in figure 6.3:

Figure 6.3 Selection of the input values for the subsystem's elements



Since the connections between the three attributes are static, the selection procedure and the connections between the sets of elements is done on the Excel spreadsheet.

6.2 Characteristic elements of the optimisation process

At this stage, the designer identified the system, internal structure of the system and input elements to the system. Therefore he/she has enough information to start searching for the optimal configuration of the system.

The most optimal configuration of the system can be found by optimising system's subsystems. The optimisation of each subsystem is conducted through optimisation of the subsystem's three attributes (see figure 4.6 in Subsection 4.3.1)

The most important elements which control optimisation of the three subsystem's attributes are:

- Control Input Value of Capability
- Control Input Value of Dependability
- Control Input Value of Availability

Optimisation of each of the subsystem's attributes takes place within the boundary drawn by the Control Input Value (see Subsection 4.3.1). The role of the computer software (see Chapter 5) is to optimise the system more efficiently.

6.2.1 Control Input Value of Capability

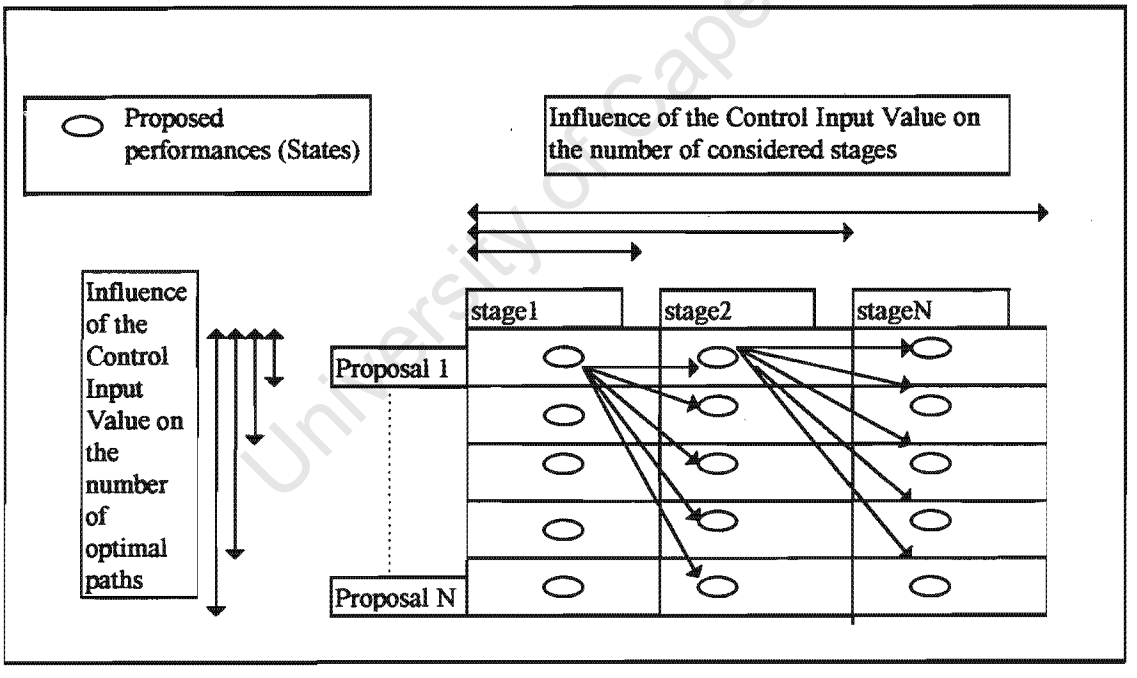
It is up to the designer how the Control Input Value is related to the Effectiveness of the entire subsystem. Usually, the most important Control Input Value is the one which influences the performance (Capability attribute). This Control Input Value characterises the total resource which the entire subsystem needs. Therefore, this Control Input Value

affects the ratio of Effectiveness to Control Input Value Factor not only through the Capability attribute, but also through Control Input Value Factor.

The Control Input Value of the Capability attribute is the total amount of the Investment allocated to the performance values of each operational stage of a subsystem. Each operational stage contains a set of performances and their respective resource requirements. The proper choice of the Control Input Value magnitude and the allocation of its fractions to every operational stage will select the most optimal performance values.

The way in which the magnitude of the Control Input Value influences the size of dynamic programming, is shown in figure 6.4.

Figure 6.4 Influence of the magnitude of the Control Input Value on the size of dynamic programming



The optimisation continues as long as the magnitude of the Control Input Value is larger than the total sum of all allocated investments of the proposed alternatives (states).

The bigger the Control Input Value the better is the chance to obtain the higher effectiveness, but, at the same time, the Control Input Value Factor becomes lower. Therefore, the output of the subsystem (i.e. ratio of Effectiveness to Control Input Value Factor) may not be as high as it is needed. If the Control Input Value is lower than the lowest total investment of all stages, then not all the optimal paths or even not all stages may be considered.

Thus, the Control Input Value must reflect the optimisation of performances within the limits of feasible investment proposals. Finally, the most optimal values of performances will evaluate qualitative values of the Capability attribute.

6.2.2 Control Input Value of Dependability

The Control Input Value of Dependability usually characterises the time needed to transfer the subsystem from the first to the last operational stage. This Control Input Value does not represent the resource which is of primary importance for the subsystem's optimisation. Thus, the Control Input Value of Dependability is usually constant during subsystem optimisation and it is not included as the component of the Control Input Value Factor. Therefore, this Control Input Value affects the ratio of Effectiveness to Control Input Value Factor through the Dependability attribute.

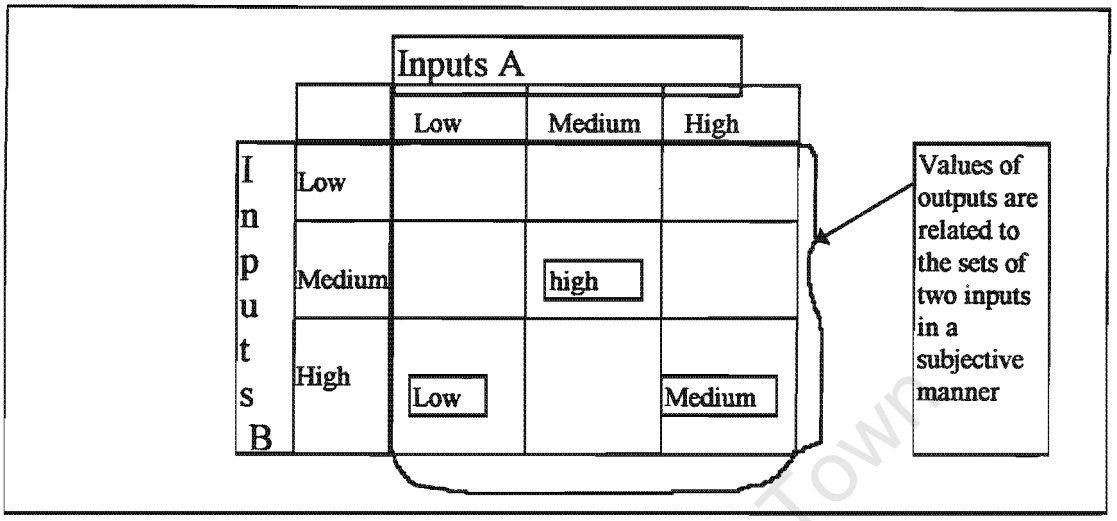
In this computer model the relationship between the Control Input Value and reliability (see Subsection 2.2.1(b)) is exponential.

6.2.3 Control Input Value of Availability

The Control Input Value of Availability is represented by the Fuzzy Rule. This rule is set up before the optimisation of the entire subsystem takes place. Thus, this Control Input Value (the same as the Control Input Value of Dependability) does not influence the Control Input Value Factor of the subsystem.

The general structure of the Fuzzy Rule is shown in figure 6.5. The Inputs (A & B) and Output are described in Section 5.1.1(c)(iv).

Figure 6.5 General structure of the Fuzzy Rule (Firing Rule)



In the case when the Fuzzy Rule is not established, the designer must decide on the relationship between Inputs and the Output. For example, Output described as high (see figure 6.5 middle cell) may have a different value if this Output relationship to the two Inputs (A & B) is changed from medium - medium to, let's say, high - high.

6.3 Evaluation of the subsystem's value (Sensitivity analysis)

Finally, all values of Capabilities, Availabilities and Dependabilities are combined (see Chapter 2) into the Effectiveness value. The Effectiveness combined with the Control Input Value of the primary element (performance) gives the ratio of Effectiveness to Control Input Value Factor.

If the Control Input Values of the Dependabilities and Availabilities are constant throughout the optimisation, then the sensitivity analysis of the subsystem can be accomplished by varying only the Control Input Value of the Capability attribute with respect to the ratio of Effectiveness to Control Input Value Factor.

If the configuration of all Control Input Values is not constant, then all the Control Input Values must change until they reach the highest ratio of Effectiveness to Control Input Value Factor. The author suggests that two of the Control Input Values should have fixed values while the third one is changed until it reaches the best ratio of Effectiveness to Control Input Value Factor. In this case, a four dimensional model can be built for each subsystem.

Figure 6.6 summarises the method which optimises the ratio of Effectiveness to Control Input Value Factor.

Figure 6.6 Optimisation of the ratio of Effectiveness to Control Input Value Factor

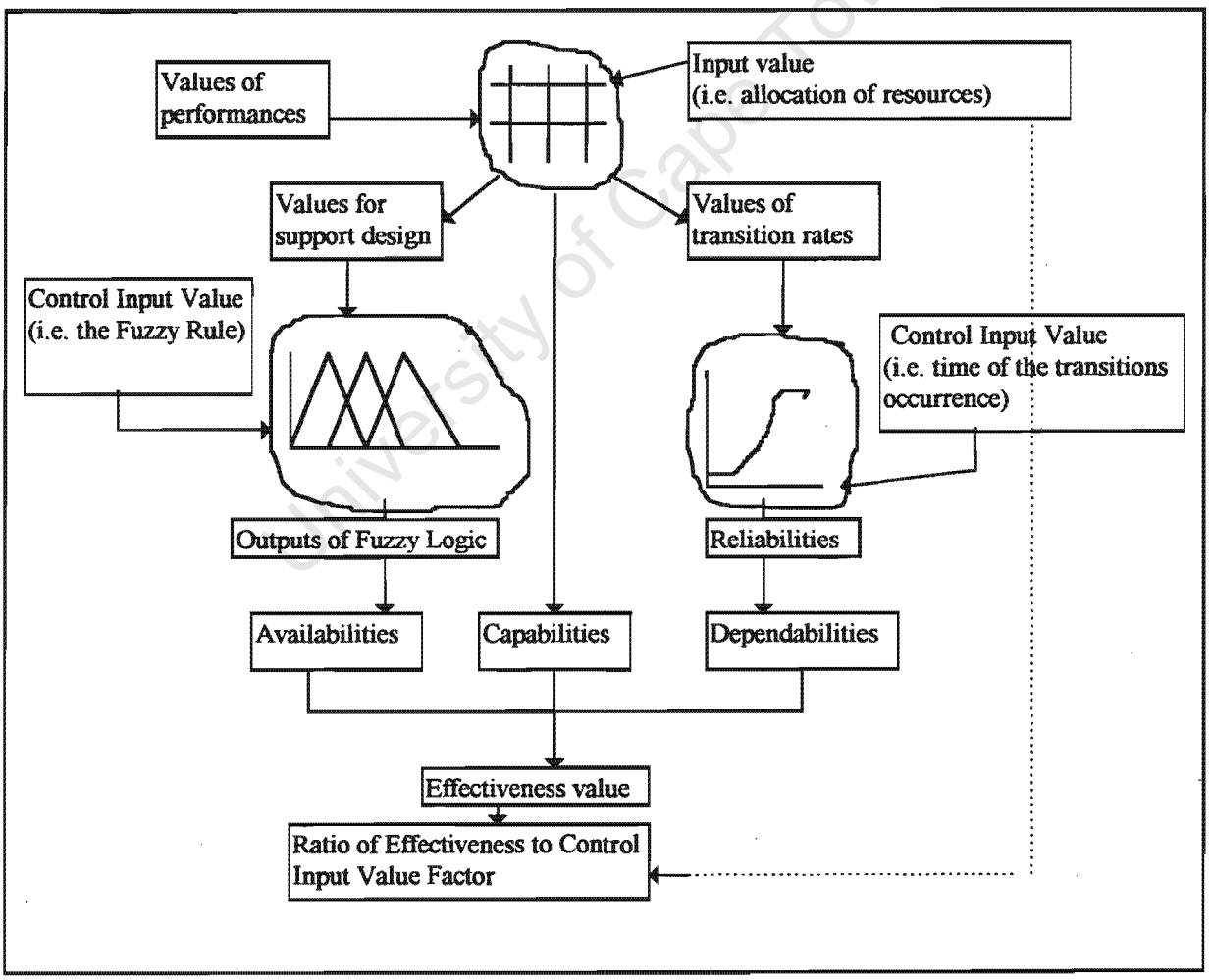
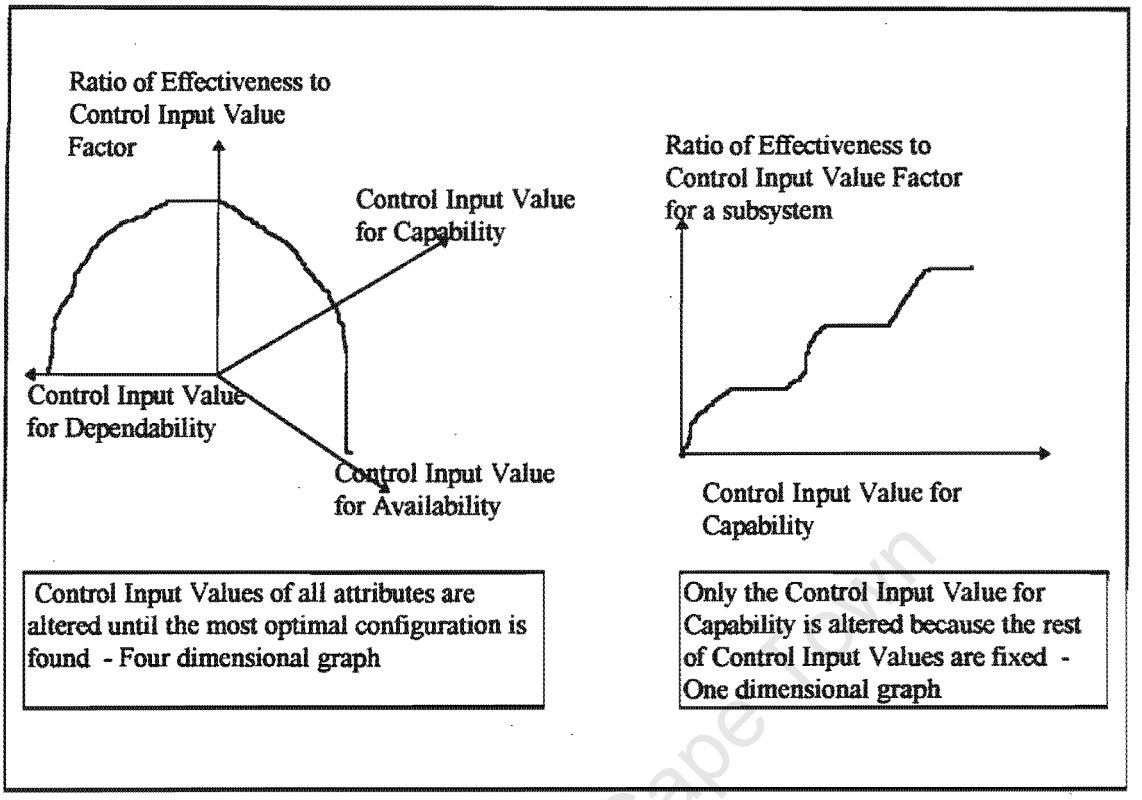


Figure 6.6 (continued)



The most optimal ratios of Effectiveness to Control Input Value Factor are combined into the total worth of the system. These ratios are displayed in the FGR.xls file.

7. APPLICATION OF THE THEORY AND THE MODEL IN A DETERGENT MANUFACTURING COMPANY

The model was tested in an environment of a detergent manufacturing company. This type of business grew very rapidly in the Cape region because there was limited competition in this sector of the industry.

The manufacturing process of finished detergents is, in most cases, very simple (sometimes one big drum and the motor with a propeller may suffice). Therefore, the number of new detergent manufacturers steadily increased. These manufacturers look for the production systems which can deliver a faster and more reliable service under reasonable prices. Thus, the primary and secondary objectives of this project were:

a) to find the least time-consuming production system under reasonable capital cost;
b) to find the economic plan which could support the selected production system.

First, the author will identify the objectives of the system. Then, the relationship between subsystems and characteristics of the subsystem's internal structure will follow. Finally, the results and conclusions will be presented.

7.1 Identification of the system

The manufacturing system of finished detergents (small business) contains two major subsystems. They are the Technical and Economic Subsystems. The question may be asked: what about the Environmental and Social Subsystems?

The impact of the manufacturing process on the environment is minimal because raw materials are already half products. The mixing process of detergents usually does not produce a chemical reaction. Therefore, there is a very limited number of unwanted products (e.g. pollution). Thus the Environmental Subsystem may be omitted.

The Social Subsystem in South African small companies is very basic. As a result of the high rate of unemployment, work standards are kept at a minimum level. Therefore, social benefits in these manufacturing companies are kept according to the minimum labour law requirements. This means that workers very often do not get bonuses or other financial incentives for their efforts. Thus, the Social Subsystem will have a very limited impact.

7.2 Relationship between the subsystems

The primary objective of the system is to find the fastest manufacturing time. Therefore, the Technical Subsystem is a Main Subsystem while the Economic Subsystem falls into a secondary category.

The choice of the most feasible working time (performance) influences the selection of the manufacturing unit. These selected units have to be supported by the feasible economic plan. The plan is based on the selection of the best repayment policies (i.e. repayment of the selected unit). Each manufacturing unit has two characteristic repayments. In this case the repayments are the performances of the Economic Subsystem.

The selection of the technical performance influences the selection of the sets of economic performance. Thus, two subsystems are connected together by their performance (Capability) elements.

7.3 Characteristics of the subsystems' elements

Every subsystem has three main attributes: Capability, Dependability and Availability. These attributes are characterised by respective input elements (performance, reliability and support design). The Control Input Value will optimise the configuration of these input elements.

7.3.1 Technical Subsystem

(a) Capability

The selection of performances of the Technical Subsystem (minimum time spent at each working stage) depends on the selection of suitable units/states of the manufacturing system. To find the smallest working time for each operational working stage it is important to select appropriate equipment (see table 7.1). This selection will be governed by the cost of the equipment. The sum of all these costs is the Control Input Value of Capability.

The Control Input Value defines the cost of capital. The company defines it as the purchase and installation costs. The fact that other costs (see Section 2.4.1) are not included greatly simplifies this project.

The characteristics of all manufacturing units for every production stage is shown in Table 7.1:

Table 7.1 Table of proposed performances for the Technical Subsystem

Proposals ↓	**Stage1** **Preparation and transport of raw materials (working time and capital cost)**	**Stage2** **Mixing of ingredients (working time and capital cost)**	**Stage3** **Packaging of products (working time and capital cost)**
Automatic line	Electrical pumps	Tank controlled by computer	Automatic filling with packaging machine
Semi-automatic line	Hand operated pumps	Tank operated by operator	Semi automatic controlled by worker
Manual line	Buckets and trolleys	Motor with propeller put in and out of the drum	Filling and packaging done by workers
Use of outside service	Not shown by the contractor	Not shown by the contractor	Not shown by the contractor

The numerical values of the Technical Subsystem's performances are given in Appendix E.

In this case, it is assumed, after consultation with the management and workers, that the qualitative value of Capability of the Technical Subsystem defines the ratio of the smallest working time in a given stage to the value of a chosen (optimal) working time of the same stage. These ratios are calculated for every stage.

(b) Dependability

The next step is to define the behaviour of the subsystem's stages. In other words, we analyse the subsystem's Dependabilities.

The Control Input Value is identified as the working time of 200 hours. After this time, the system undergoes maintenance before it will operate another 200 hours.

Three stages are identified for this subsystem. Optimisation of the three stages of the subsystem results in three optimal states which represent the chosen equipment (see table 7.1). The success of the operation in the previous state (e.g. in mixing stage) leads to the transfer of the production process to the next state (e.g. in packaging stage). A fourth stage is also added to represent a production failure. This stage is activated (i.e. the production system is stopped) if any of the other stages fail.

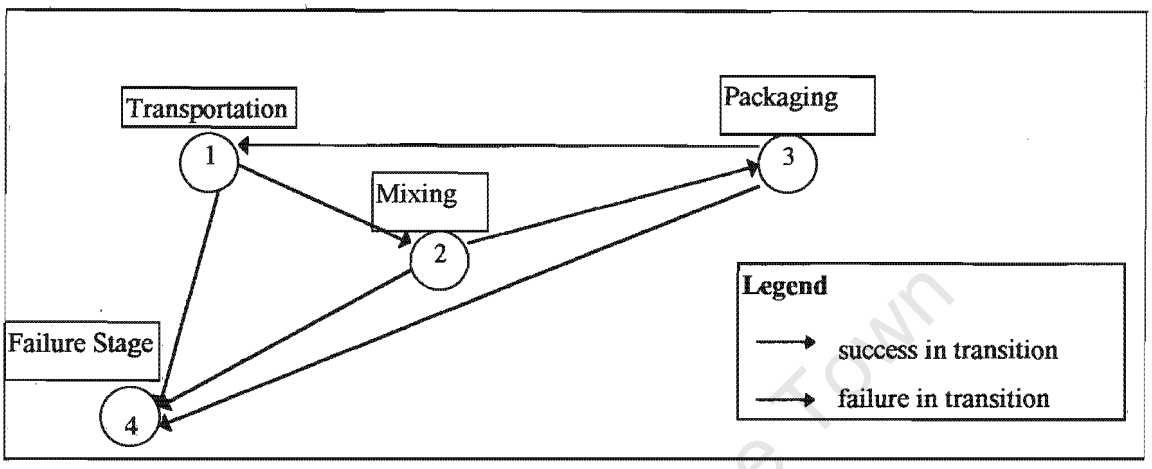
The model considers the fourth stage for Dependability analysis only. It was simply impossible to include this stage for other attributes because we did not have probabilities values of transition from stage four to other stages.

The rates of transitions are selected according to the selection of Capabilities (see computer modeling process, Chapter 5). The equations of states (equations for transition rates) are exponential functions because no further suggestions were made in this matter. In a case

where a subsystem is more precisely known, these equations could have other mathematical forms.

Figure 7.1 shows the state-space model for the Technical Subsystem's behaviour.

Figure 7.1 State-space model for Technical Subsystem



Since the values of Capabilities are selected out of twelve performance elements (table 7.1) therefore, 3 times12 sets of transitions may exist. The number "3" comes from three stages which characterises the Technical Subsystem.

The formula for failure rates evaluates the transition rates (see Chapter 2). This is done according to the following relationship:

$$\text{Transition rate} = \frac{\text{number of failures}}{(\text{total available working time} - \text{time spent for down time})}$$

For automatic and contractors options (see table 7.1), the possible number of failures and the average time spent for down times (e.g. time for replacements of parts) is constant. The number of failures for manual and semi-automatic options is based on data from observations. The table of values for every set of failure rates is given in Appendix E.

The probabilities of failures and successes of transitions are evaluated according to the following exponential equation:

$$\text{Probability of transition} = e^{(-(\text{transition rate})*(\text{time of expected operation}))}$$

(c) Availability

Finally, the support design (Availabilities) of every stage needs consideration. According to the computer model two Availability inputs (Input A and Input B) characterise every operational stage. The Input A defines the suitable worker with the required experience to repair/maintain the equipment. The Input B defines the complexity of replaceable parts of which the considered equipment is made.

These two inputs are evaluated subjectively by giving them values from 1 to 5 for workers' experience and 1 to 10 for the spare parts complexity (see Appendix C on the allocation of Inputs A and B to every stage).

The Input A (worker's experience) has three target values:

Low -1

Medium - 3

High - 5

The Input B (complexity of spare parts) also has three target values

Low - 2

Medium - 5

High - 8

The outputs define average times spent by the workers to make the equipment fully operational. The relationship between the output and the input is also subjective and based on the past experience of both workers and the management.

The respective three target values of Outputs are:

Low - 15
Medium - 10
High - 5

The Control Input Value, which is the Fuzzy Rule, has the following characteristics (table 7.2):

Table 7.2 Fuzzy Rule for Availability of the Technical Subsystem

LOW = 1
MEDIUM = 2
HIGH = 3

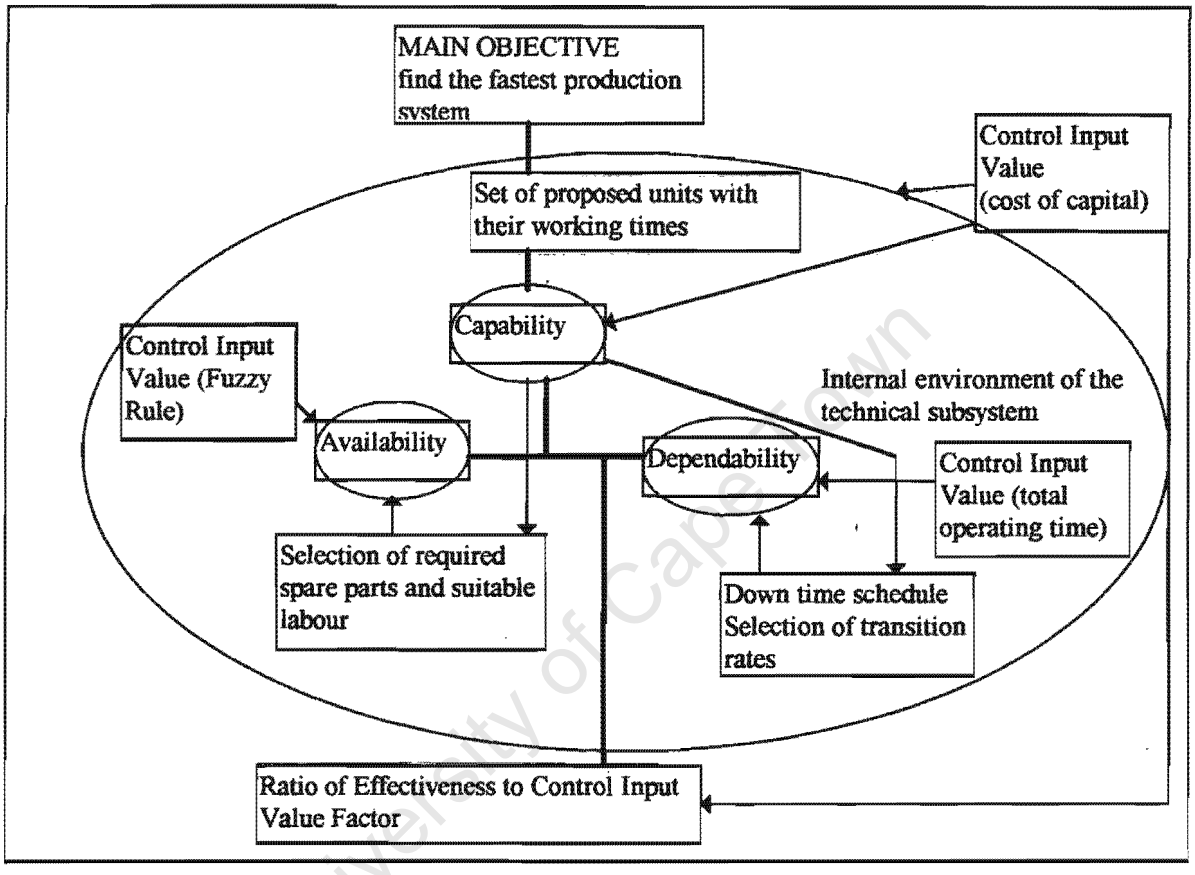
INPUT B	INPUT A		
	LOW	MEDIUM	HIGH
LOW	2	1	1
MEDIUM	3	2	1
HIGH	3	3	2
	OUTPUTS		

The values of Availabilities (qualitative ratios) are calculated according to the theoretical description shown in Section 2.3.1 (c).

(d) Remarks about the Technical Subsystem

Figure 7.2 shows the relationships between the elements of the Technical Subsystem.

Figure 7.2 The relationships between the elements of the Technical Subsystem



Capabilities dominate because they influence the selection of input elements for Availabilities and Dependabilities. However, there is not a direct influence of Capabilities on the optimisation of Availabilities and Dependabilities. These two attributes of the subsystem are optimised independently. The bold linework in figure 7.2 represents the core of the subsystem's optimisation which starts from the formulation of the objective to the final evaluation of the subsystem's ratio of Effectiveness to Control Input Value Factor.

The Control Input Value of Capabilities controls the optimisation for the entire subsystem. The Control Input Values of other attributes (Availability and Dependability) are less important because they have constant values.

7.3.2 Economic Subsystem

(a) Capabilities

The Economic Subsystem evaluates the repayment policy to finance the purchase of the equipment. The procedure for modeling the Economic Subsystem is the same as in the case of the Technical one. The only difference is that in the Economic Subsystem the selection of the sets of performances depends on the performance selected in the Technical Subsystem (see Section 5.2). Thus, the selected equipment which represents the optimal performance in the Technical Subsystem, selects the economic performances.

The economic performances represent monetary values of interests. For every stage which characterises the selected equipment, the repayment policy is chosen out of two proposals (see table 7.3). The proposals differ in values of both the interest and the time of repayment.

Table 7.3 explains the economic performances. The numerical values of economic performances are given in Appendix E.

Table 7.3 General characteristics of economic performances

	Stage 1 Transport	Stage2 Mixing	Stage3 Packaging
Option 1	interest and time	interest and time	interest and time
Option 2	interest and time	interest and time	interest and time

The time of repayment defines the Control Input Value of economic Capabilities. This time limits the boundary within which the Economic Subsystem is evaluated. The evaluation of economic Capability is similar to that of the technical one.

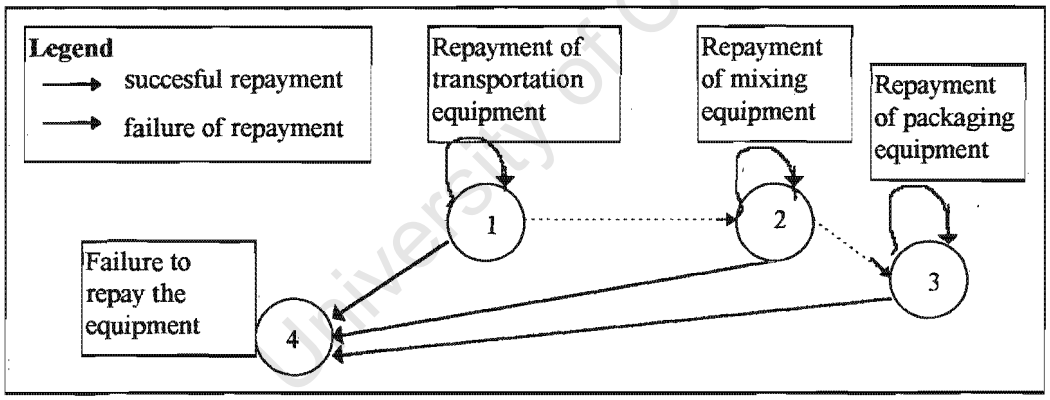
(b) Dependabilities

Dependabilities evaluate the probability of the repayment of interest. The probability of the success in the repayment of interest depends on the successful sales of products. The repayment policy is stage wise. The repayment of the first unit allows consideration of the repayment of the second one. Therefore, the structure of the state model for the Economic Subsystem has the same structure as the Technical Subsystem (see figure 7.3).

The probability of failure or the success of repayment depends on the choice of the manufacturing unit. Therefore, the high quality equipment increases the quality of the production and this increases the probability of sales generation. Therefore, the probability of repayment increases.

Figure 7.3 shows the state-space model of the probability of repayment of interests.

Figure 7.3 State-space model of Economic Subsystem.



In this model, the success of the economic plan depends on the probability of the successful repayment of each production unit. Thus, the probability of successful transition from one stage to another is not as important as the probability of transition within the same stage (i.e. success in being in the particular stage). This is because the probability of transition from the previous stage to the current stage indicates that the current stage of repayment will be successful. Nevertheless, the successful repayment of the previous stage means that

the repayment of the current stage can be initiated. Figure 7.3 shows the pattern of the repayment as the dashed lines.

In the case when the repayment of the equipment would be not successful, the entire economic process is stopped and then transferred to the Failure Stage (stage 4). The stage four, in figure 7.3, is only shown to explain the state model for the Economic Subsystem. However, this stage is not included in the models of Availability and Capability attributes because of the same reasons as those explained in the case of the Technical Subsystem (subsection 7.3.1.(b)).

The transition rate in the Economic Subsystem represents the number of successful repayments which took place during the sales time. Thus the transition rate is:

$$\text{Transition rate} = \frac{\text{number of succesful sales}}{\text{(total available working time - time spent for unsuccesful sales)}}$$

The numerical values for transition rates are given in Appendix E.

Similarly as for Dependability of the Technical Subsystem, the probability of successful repayment has an exponential relationship:

$$\text{Probability of succesful repayment} = e^{(-(\text{transition rate})*(\text{time of expected operation}))}$$

The sets of transition rates in the Economic Subsystem do not depend on the selected economic but on the selected technical performances. In this case, the number of transitions is the same as in the case of the Technical Subsystem (see subsection 7.3.1.(b)).

The total time at which Dependabilities are evaluated (the time executed for the repayment plan) is the same as the total time allocated for the economic performances. Therefore, in

the Economic Subsystem, the Control Input Value of Dependabilities is the same as the Control Input Value of Capabilities.

(c) Availability

The economic Availabilities are evaluated in the similar way as the technical Availabilities. However, the Inputs A and B to the Availability attribute are chosen through the selection of performances of the Technical Subsystem. This means that the chosen technical performance selects the set of dependent values of the Inputs A and B. Thus, a more sophisticated production unit will increase the number of new products and improve their quality. This then increases the amount of sales, which increases the chances of repayments.

Inputs A and B are defined as the work quality improvement and percentage introduction of the number of new products respectively. The Output of Inputs (A and B) is assumed to be the number of customers. The relationship between the Output and the two Inputs is subjective and depends on the management observations.

Three targets of Inputs A (quality improvement) are:

Low	1
Medium	3
High	5

Three targets of Inputs B (percentage of introduction of new products) are:

Low	10
Medium	30
High	50

The targets of the Output (number of customers) are:

Low	5
Medium	10
High	15

The values of Inputs and their respective Outputs are given in Appendix E.

The Control Input Value of Input A and B, that is the Fuzzy Rule, is shown in figure 7.4:

Figure 7.4 Table of the Fuzzy Rule for economic Availabilities

LOW = 1
MEDIUM = 2
HIGH = 3

INPUT B	INPUT A		
	LOW	MEDIUM	HIGH
LOW	1	1	2
MEDIUM	1	2	2
HIGH	1	2	3
	OUTPUTS		

The values of Availabilities (qualitative ratios) are calculated according to the theoretical description shown in Section 2.3.1.(c).

(d) Remarks about the Economic Subsystem

The internal structure of the Economic Subsystem is shown in figure 7.5.

Figure 7.5 Economic Subsystem

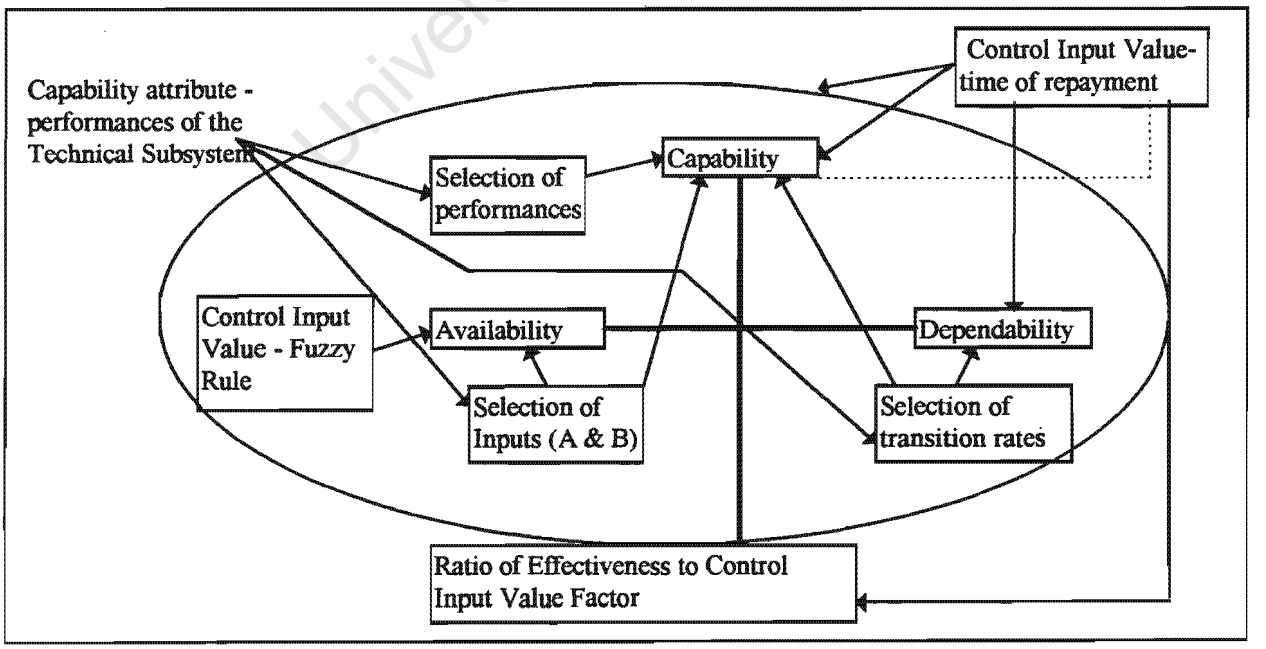


Figure 7.5 shows how the Capability attribute dominates in the Technical Subsystem. This attribute influences not only the selection of sets of economic performances but also sets of transition rates for Dependabilities evaluation and sets of Inputs A and Inputs B for Availabilities evaluation.

Practically, there are only two Control Input Values. The Control Input Value of economic Capabilities optimises both Capabilities and Dependabilities. The Control Input Value of the Availability attribute is constant and, therefore, it is used to evaluate the qualitative ratios of Availabilities.

7.4 Results

The results are compared among three different approaches, namely the Chemical Manufacturer's approach, M'Pherson's theory and the author's approach based on M'Pherson's philosophy. All three theories use the same numerical values.

7.4.1 Manufacturer's approach

The company's approach is to screen the available values of economic and technical performances to find the most optimal setup for the manufacturing system. This approach considers neither the qualitative ratios of the Capability attribute nor the evaluation of input elements and outputs of the Dependability and Availability attributes. In this case, only the performance values are compared with one another.

(a) Results of the Technical Subsystem evaluation

The maximum Investment for the selection of the suitable equipment is limited, by the company, to R45000. It should be clear that the highest total investment would result in the highest total performance value. Therefore, the Technical Subsystem is rather evaluated than optimised.

Table 7.4 shows the results of the analysis.

Table 7.4. Results of screening technical performances

Stage1 proposal 3 (Investment requirement of R 5000)	Stage2 proposal 2 (Investment requirement of R 20000)	Stage3 proposal 4 (Investment requirement of R 20000)
40 min	60 min	50 min

The screening process appeared to be time consuming. It is worth mentioning that, when all values of performances were typed into the computer programme, it took only a fraction of time for the developed programme to find the same values of performances as those which were found during the screening process.

At this stage, one may ask the question: Why was the investment of R45000 chosen as the boundary for the Technical Subsystem? The company did not have an official method to optimise performance values. The management claimed that a relatively high investment would influence not only performance, but also other components of the subsystem's structure such as Availability (function of support/logistics) and Dependability (function of reliability).

The value of the subsystem's reliability and support are assumed to be within satisfactory limits for all options of performances. Therefore, they are not further investigated.

(b) Results of the Economic Subsystem optimisation

The optimised performances of the Technical Subsystem have characteristic sets of values of economic performances. The selected values of economic performances are then screened to find the most optimal choice of repayment . The results of this selection is shown in table 7.5

Table 7.5 Selected sets of economic performances related to the selected optimal technical performances

Proposal 3 (from Technical Subsystem)	Proposal 2 (from Technical Subsystem)	Proposal 4 (from Technical Subsystem)
Stage1	Stage2	Stage3
interest 0.63 months 1	interest 2.32 months 2	interest 1.055 months 4
interest 039 months 2	interest 1.12 months 5	interest 3.71 months 2

The company uses three plans of repayments. These plans are scheduled for 12, 9 or 6 months and they characterise the boundaries of the Economic Subsystem. The *present value factors* (see subsection 2.2.2(a)) calculates monetary values of interest accumulated during the time of repayment. The interest rates are imposed by the bank. The details of interests calculations are given in Appendix D. Again, the management uses the screening process to select the most optimal proposals of economic performances. These are shown in Table 7.6.

Table 7.6 Selection of the most optimal economic performances

Months	Stage1	Stage2	Stage3	Total values accumulated * 1000
12	Proposal 2	Proposal 2	Proposal 1	R10.6
9	Proposal 2	Proposal 1	Proposal 1	R9.64
6	Proposal 2	Proposal 1	Proposal 2	R12.846

Table 7.6 shows the 9 month plan as the most optimal. The numerical values of the above proposals are given in Appendix E. Since the manufacturer does not consider the Dependability and Availability attributes, therefore the choice of values in Table 7.6 does not show the system's true effectiveness.

7.4.2 M'Pherson's approach

M'Pherson's approach dynamically optimises the Technical Subsystem and leaves other subsystems for a static evaluation (see Chapter 4). This means that M'Pherson recognises three attributes to evaluate the Technical Subsystem.

None of M'Pherson's papers (M'Pherson[20,21]) refers to the specific methods to evaluate components of a Technical Subsystem in the small developing business. Therefore, the author uses his own techniques (see Chapter 3) to evaluate attributes of the Technical Subsystem. In this case, the Technical Subsystem is a subject of optimisation of Capabilities, Dependabilities and Availabilities. The Economic Subsystem's values are evaluated in the same way as those in the Manufacturer's approach because M'Pherson does not include a specific method for the optimisation of the Economic Subsystem.

(a) Results of the Technical Subsystem optimisation

According to the theory presented in Chapters 2 and 4, the result of the Technical Subsystem optimisation is the ratio of Effectiveness to Control Input Value Factor. This ratio is optimised against the Control Input Value, which is the total Investment allocated for the purchase of the manufacturing equipment. The results of optimisation between the ratio of Effectiveness to the Control Input Value Factor and the Control Input Value are shown in table 7.7.

Table 7.7 Ratio of Effectiveness to Control Input Value Factor for the Technical Subsystem

Control Input Value *R1000	20	25	30	35
Ratio of Effectiveness to Control Input Value Factor	*0.0328*	*0.0562*	*0.0367*	*0.0530*

Control Input Value *R1000	40	40	45	50	50	55	55
Ratio of Effectiveness to Control Input Value Factor	*0.0614*	0.07561	*0.0640*	0.103298	*0.0769*	0.1034	0.0769

Control Input Value *R1000	60	60	65	70	75
Ratio of Effectiveness to Control Input Value Factor	*0.0943*	0.1399	*0.1480*	*0.1428*	0.1522

It should be clear that M'Pherson's approach gives a greater scope for the Technical Subsystem's optimisation. Here, the three recognised attributes, combined into the effectiveness value, are balanced against the subsystem's main resource requirement. The resource requirement is used to evaluate the Control Input Value Factor. The optimisation of the Technical Subsystem is achieved through the balance between the Control Input Value and the ratios of Effectiveness to the Control Input Value Factor.

If the company still has an upper investment limit of R45000 then, according to the table above, the better option would be to choose the R40000 investment with the ratio of Effectiveness to Control Input Value Factor equal to 0.07561. The reason that the R40000 investment has the higher effectiveness value than the R45000 investment is mainly in the introduction of attributes to the system's optimisation - Capability, Dependability, Availability.

(b) Results of Economic Subsystem optimisation

Again, the screening method optimises the Economic Subsystem. The table of proposals of economic performances for the R40000 Investment with the ratio of Effectiveness to Control Input Value Factor of 0.07561 (i.e. the Technical Subsystem) is shown in table 7.8 below.

Table 7.8 Proposed economic performances for the R40000 Investment from Technical Subsystem (all interests must be multiplied by R1000)

Proposal 2 (from the Technical Subsystem)	Proposal 3 (from the Technical Subsystem)	Proposal 4 (from the Technical Subsystem)
Stage1	Stage2	Stage3
interest 1.33 months 1	interest 0.3 months 2	interest 1.055 months4
interest 1.04 months 2	interest 1.65 months 1	interest 3.71 months 2

Once the screening process is finished the following results are obtained:

Table 7.9 Selection of the most optimal economic performances

Months	Stage1	Stage2	Stage3	Total accumulated interest *1000
12	Proposal 2	Proposal 1	Proposal 1	6.9
9	Proposal 2	Proposal 1	Proposal 1	6.9
6	Proposal 1	Proposal 2	Proposal 1	7.2

Table 7.9 shows that the 12th and the 9th month has the lowest interest value. Obviously, the 9 month plan is shorter, therefore it is the most optimal. The numerical values of the Economic Subsystem (Secondary Subsystem) are given in Appendix E.

M'Pherson's method, the same as the manufacturer, does not consider the economic Availability and Dependability attributes. Only the Technical Subsystem is evaluated not only against its performance but also against its survivability (Availability) and behavioural changes (Dependability).

The final result is, again, based on the value selected from the Technical Subsystem. However, the direct technical performance value is replaced by the Technical Subsystem's ratio of Effectiveness to Control Input Value Factor. The multi-variable nature of this ratio shows new optimal options which in the previous, more standard Manufacturer's approach would not be noticed.

7.4.3 The Author's Approach

The author claims that all subsystems have exactly the same optimisation patterns. Subsystems are optimised according to their Capabilities, Dependabilities and Availabilities attributes and outputs of these subsystems have the ratios of Effectiveness to Control Input Value Factor.

Table 7.10 shows the results of the Technical and Economic Subsystems optimisation based on the author's approach.

Table 7.10 Ratios of Effectiveness to Control Input Value Factor for Technical and Economic Subsystems

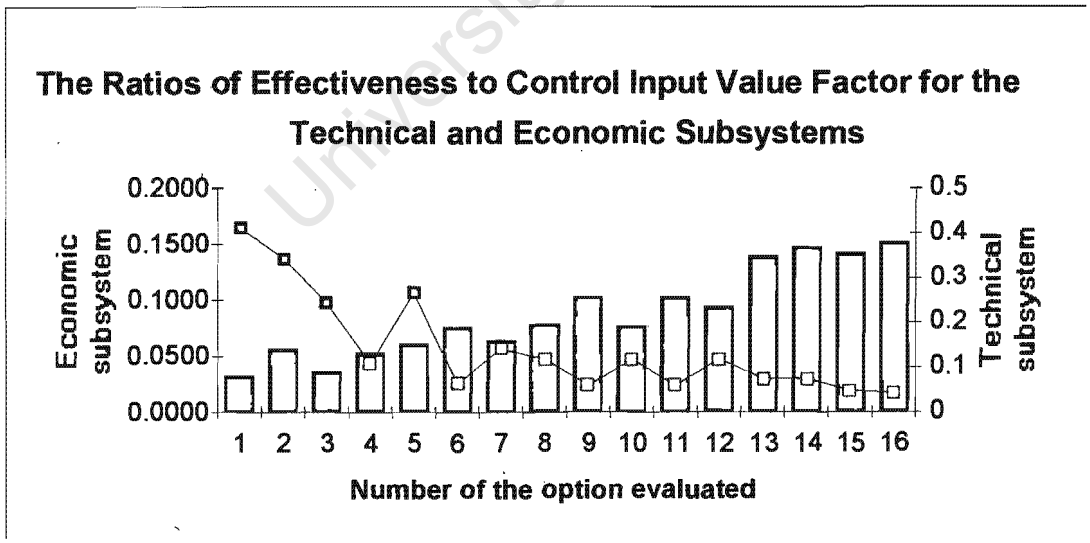
TECHNICAL SUBSYSTEM	Option1	Option2	Option3	Option4	Option5	Option6	Option7
Control Input Value * 1000	20	25	30	35	40	40	45
Technical ratio of Effectiveness to Control Input Value Factor	*0.0328*	*0.0562*	*0.0367*	*0.0530*	*0.0614*	0.07561	*0.0640*
ECONOMIC SUBSYSTEM	Option1	Option2	Option3	Option4	Option5	Option6	Option7
Economic ratio of Effectiveness to Control Input Value Factor							
for 6 months	0.41271253	0.344290	0.24326846	0.107192	0.266074	0.06239	0.141668
for 9 months	0.21709	0.18251	0.13061	0.1421	0.17012	0.111846	0.00652
for 12 months	0.12871	0.10904	0.07385	0.08552	0.10211	0.068316	0.00375

Table 7.10 (continued):

TECHNICAL SUBSYSTEM	Option8	Option9	Option10	Option11	Option12	Option13	Option14	Option15	Option16
Control Input Value * 1000	50	50	55	55	60	60	65	70	75
Technical ratio of Effectiveness to Control Input Value Factor	*0.0780*	*0.103298*	*0.0769*	*0.1034*	*0.0943*	*0.1399*	*0.1480*	*0.1428*	*0.1522*
ECONOMIC SUBSYSTEM	Option8	Option9	Option10	Option11	Option12	Option13	Option14	Option15	Option16
Economic ratio of Effectiveness to Control Input Value Factor									
for 6 months	0.117811	0.057941	0.117811	0.05794	0.118496	0.07402	0.074028	0.047401	0.042071
for 9 months	0.06643	0.03027	0.0609	0.03617	0.0365	0.03064	0.04274	0.03653	0.02627
for 12 months	0.04724	0.02355	0.03469	0.02758	0.0227	0.01749	0.022166	0.02279	0.02216

Graphically, the best technical and economic ratios of Effectiveness to Control Input Value Factor are shown together in figure 7.6

Figure 7.6 Relationship between the Technical and the Economic ratios of Effectiveness to Control Input Value Factor



The total evaluation of all the ratios took 48 simulations after which the above graph was generated.

Careful analysis of the above results shows that Option 5, which is the investment of R40000 with the Technical Subsystem's ratio of Effectiveness to Control Input Value Factor of 0.0614, is the most optimal (see table 7.10). The technical and economic ratios of Effectiveness to Control Input Value Factor are relatively high. The Option 5 uses lower Investment (Control Input Value) for both Technical and Economic Subsystems. The investment is now R40000, R5000 less than under the Manufacturer's approach. The repayment plan is now 6 months, which is 3 months less than under M'Pherson's approach.

The above observations are more obvious if the best values of economic and technical ratios are compared to the rest of the ratios. Thus, it was decided to find the relative distance between the best and the other values of ratios of Effectiveness to Control Input Value Factor, for each subsystem. The distance shows how far the considered option is from the ideal case. The sum of two distances (economic and technical) represents the total distance, which shows how far the Total Worth of the entire system is from the ideal value (see table 7.11). Usually, if the distance between the best and the investigated ratio of Effectiveness to Control Input Value Factor is less than 30%, then the combined solution of the Economic and Technical Subsystem optimisation is not feasible.

For example, an option with the very high technical ratio will usually show a very low economic ratio and vice versa. The calculations of the distances for Technical and Economic ratios of Effectiveness to Control Input Value Factor are shown in Table 7.11 below:

Table 7.11 Percentage distance of the proposed options from the ideal option

Option	1	2	3	4
% total distance	78.4541	79.6864	116.9317	139.2377
%technical distance	78.4541	63.1077	75.8755	65.2101
% economic distance	0.0000	16.5787	41.0562	74.0275
Main (invest.)	20	25	30	35

Option	5	6	7	8	9
% total distance	95.1869	135.2098	123.6025	120.2290	118.0976
%technical distance	95.1869	50.3268	57.9285	48.7745	32.1367
% economic distance	35.5304	84.8829	65.6740	71.4545	85.9609
Main (invest.)	40	40	45	50	50

Option	10	11	12	13	14	15	16
% total distance	120.9536	118.0309	109.3558	90.1555	84.8322	94.6797	89.8063
%technical distance	49.4991	32.0698	38.0672	8.0905	2.7691	6.1650	0.0000
% economic distance	71.4545	85.9612	71.2886	82.0650	82.0631	88.5147	89.8063
Main (invest.)	55	55	60	60	65	70	75

(The options in the blue frames are not feasible (see explanation in the text)

The Author's aim is to find the smallest distance between the ideal ratio of Effectiveness to Control Input Value Factor and the optimal ratio of Effectiveness to Control Input Value Factor. For both the economic and the technical ratio of Effectiveness to Control Input Value Factor, the total sum of the two distances is used to find the most optimal option.

Again, Option5 is the most optimal because it has the smallest percentage value of the total distance (i.e.95.65 %).

If one would like to still insist on the investment of R45000 (Manufacturer's approach) instead of R40000, then the author's approach would still be more optimal. For example, Option7, which uses R45000 has a 6 month repayment plan instead of the 9 month plan proposed by the Manufacturer's approach.

The above observations show that not only Capability, but also Availability and Dependability, influences the final outputs of Technical and Economic Subsystems. Therefore, a system with a good performance is not necessarily an effective one.

The introduction of the ratio of Effectiveness to Control Input Value Factor to the Economic Subsystem shows that the Economic Subsystem is independently optimised from the Technical Subsystem and that the Economic Subsystem is now more significant than it was in previous approaches.

7.5 Conclusions

The proposed Systems Engineering model may be used in a small sized developing system such as the detergent-manufacturing small business corporation. The model showed the ability to resolve the conflicting objectives of the Economic and Technical Subsystems through the ratios of Effectiveness to Control Input Value Factor.

The Capabilities, Dependabilities and Availabilities may evaluate not only the effectiveness of the Technical Subsystem (M'Pherson[20,21]), but these attributes can also be used to evaluate the Economic Subsystem.

The model may not be used in applications where one would exclusively use an objective analysis. Subjectivity is, nevertheless, a part of the management of the small developing system and, therefore, the subjective approach can not be avoided. Thus the small optimal developing system will always partially depend on the designer's intuition.

Finally, it is difficult to validate that the final value of the presented model is the most optimal because:

- there are no objective mathematical models which one would apply for this particular example; and

- the author's model contained elements which were not considered in the other approaches (Manufacturer' approach has not considered Dependability and Availability for the Technical and Economic Subsystem; M'Pherson's approach has not considered Dependability and Availability for the Economic Subsystem).

8. REFLECTIONS

On the basis of the presented work the author would like to reflect on the developed computer programme, theory and the application of the developed theoretical model in practical situations.

8.1 Reflections on the computer model

The size of the programme may not be suited for the analysis of very large sized problems. All of the computer modules are limited due to the memory problem or the spreadsheet's limitations.

8.1.1 The limitations of the size of dynamic programming

Only seven nested IF-statements are possible in the Excel spreadsheet. Therefore, the dynamic programming module has a limited number of proposals, otherwise known as optimised critical paths, which in this case, is five (according to H.A. Taha[7] more than one critical path is possible). The problem of IF-statements may be solved if more than one input element's selection process is used on the same sheet. This selection process of input elements applies to Dependabilities and Availabilities in the Main and the Secondary Subsystems as well as to the selection of input elements of the Capability attribute in the Secondary Subsystem (see subsections 5.1.1(b)(ii), 5.1.1.(c)(ii) and 5.1.2). This solution should be applied with some caution because eventually the spreadsheet will be difficult to read once it is overcrowded with many IF-procedures. Perhaps the best method is to programme everything on Delphi.

The computer programme uses one-dimensional dynamic programming. Of course, one can programme multi-dimensional arrays but the programme execution will be limited by the available memory. At the moment, a two-dimensional array is used to optimise the

one-dimensional performance element (i.e. two-dimensional array has one dimension allocated for Resource and the other for Investment).

8.1.2 Limitations of Monte Carlo Simulation

This simulation is limited in the number of trials. The maximum number is seventy. Obviously the number can be doubled or tripled if the [Activate] button is clicked twice, three times, or more. The reason for this is the limitation of the computer operational memory.

8.1.3 Limitations of Fuzzy Logic

The number of fuzzy triangles is three. Obviously, if the situation demands more than three triangles, the fuzzy module programme can be increased by the introduction of new 'IF' statements (see the programme code in Appendix C).

The programme uses isosceles triangles because they are simple to model Fuzzy Logic rules but of course other geometrical shapes are possible to programme (such as trapezoidal, parabolic etc.).

8.2 Reflections on the systems engineering model

The proposed model is primarily based on the author's knowledge from practical experience because none of the similar systems engineering structures were found in the literature. Nevertheless, depending on the type and nature of the system, the user of this model is free to change the present pattern of the optimising methods in relation to the subsystem's elements.

The proposed model uses the dynamic programming to optimise performance values for Capability evaluation. Since Capability of the Main Subsystem is the dominating attribute

in the subsystem's structure, therefore, the dynamic programming primarily influences the selection of input elements to Availability and Dependability of the Main Subsystem and the Capability of the Secondary Subsystem.

The dynamic programming optimises values in the stage-wise process. Therefore, the input elements of every attribute of each subsystem requires the stage-wise optimisation (see section 3.2). This type of approach may limit the model to problems where the system's components or units are arranged in series. If there is a non-series arrangement of components then the best solution is to reduce the components into a 'black box'. Before the dynamic programming procedure commences, every 'black box' has to be evaluated separately. Each 'black box' would represent the operational stage of the system.

8.3 Reflections on Application

The presented systems engineering model showed its applicability to the systems in which subsystems' elements are diversified in nature. This model could be used in organisations which consist of departments or subsystems with conflicting objectives. Nevertheless, the model has not been fully utilised because the investigated system had only two subsystems. In the case of more than two subsystems, the calculation for the Main or the Secondary Subsystem must be repeated.

The developed theory should be applied to business structures which consist of more than two subsystems. Ideally, the investigated company should have its own optimising methodology which has the same nature as the proposed model. Only then the results may be fully validated.

9. BIBLIOGRAPHY

BOOKS

1. John Gill and Phil Johnson in "Research Methods" (U.C.T., Mechanical Engineering. Department. copy 1994).

2. B.S. Blanchard, W.J. Fabrycky "Systems Engineering and Analysis (Prentice-Hall International Editions, second edition)

3. Richard de Neufville in Applied systems Analysis"(McGraw-Hill 1990)

4. de Neufville&Stafford "Systems Analysis for Engineers and Managers" (McGraw-Hill Inc,1974)

5 D.J. White "Dynamic Programming" (Wiley, 1978)

6. N.A.J. Hastings " Dynamic Programming with Management Applications" (London:Butterworths, 1973)

7. H.A. Taha " Operations Research An Introduction"(MacMillan, 5th ed., 1992)

8.Bierman, Bonnini & Hausman "Quantitative Analysis for business decisions" (Richard D Irwin, Inc., 5th ed, 1977)

9. G.C. Dandy & R.F. Warner "Planning and design of Engineering Systems" (Unwin Hyman1989)

10 L. Fang, K.W. Hipel, D. M. Kilgour "Interactive Decision Making" (Wiley1993)

11. D.J. Sherwin&A.Bossche "The Reliability, Availability and Productiveness of Systems" (Chapman&Hall 1990)

12. W.L. Winston "Operations research" (Duxbury , ITP, third edition, 1994)

13. A. Bendell, J. Disney, W.A. Pridmore " Taguchi Methods: Applications in world Industry" (Springer, 1989)

14 D.Driankow, H. Hellendoorn, M. Reinfrank " An Introduction to Fuzzy Control" (Springer - Verlang, 1993)

15 B. Kosko "Neural Networks and Fuzzy Systems" (Prentice Hall, Englewood cliffs, NJ 07632, 1992)

16 A. Kaufmann, M. Gupta " Fuzzy Mathematical models In engineering and Management science" (North-Holland 1988)

17 Howard R. "Dynamic Probabilistic Systems" (Wiley, New York, 1971)

JOURNALS

18. A Bastiani,C. Speedy "An Introduction to fuzzy Control"(Chemical Engineering In Australia June 1996).

19. T.Sudkamp, R.J. Hammell II "Interpolation, Completion, and Learning Fuzzy Rules" (IEE Transactions on Systems, Man, And Cybernetics, Vol 24 No2, February 1994)

20. P.K. M'Pherson "A framework for systems engineering design" (The Radio and Electronic Engineer vol51, No 2 pp 59-93, February 1981)

21. P.K. M'Pherson "Systems Engineering: an approach to whole-system design" (The Radio and electronic Engineer, Vol 50, No 11|12, pp545-558, November|December 1980)

22. R.L. Ackoff "Mechanisms, Organisms and Social Systems" (Strategic Management Journal Vol. 5 1984)

23 T. Sudkamp&R.J. Hammell II "Interpolation, Completion and learning fuzzy Rules" IEEE Transactions on systems, Man, and Cybernetics, Vol 24, No 2 February 1994

24 D.G. Swartz&G.J. Klir "Fuzzy logic flowers in Japan" (IEEE Spectrum July 1992)

25. P.C. Brewer, A.W. Gatian, J.M. Reeve "Managing Uncertenity" (Management Accounting, October 1993)

APPENDIX A: Details of the Computer Software Development

A-1. DEVELOPMENT OF THE COMPUTER MODEL

The development of the computer model focuses on the subsystem's three attributes which are the most important tools to optimise the entire system. This computer model consists of four major sections. These sections are : Capability, Dependability and Availability attributes, and the final display of the results of the subsystem and system optimisation.

Each of the three attributes' sections are further divided into three major subsections:

- First subsection deals with the selection of the input elements characteristic for the optimised attribute. The selection of the input elements is done on the Excel spreadsheets.
- Second subsection optimises the selected input elements. This selection procedure is programmed on the Delphi computer language.
- Third subsection displays the optimised input elements. The optimised input elements are transferred into the qualitative ratios of Capabilities, Avalabilities and probability values of Dependabilities. The display of the optimised values of the subsystem's three attributes is shown on the Excel Spreadsheet.

A-1.1 Programming the optimisation of the Capability attribute

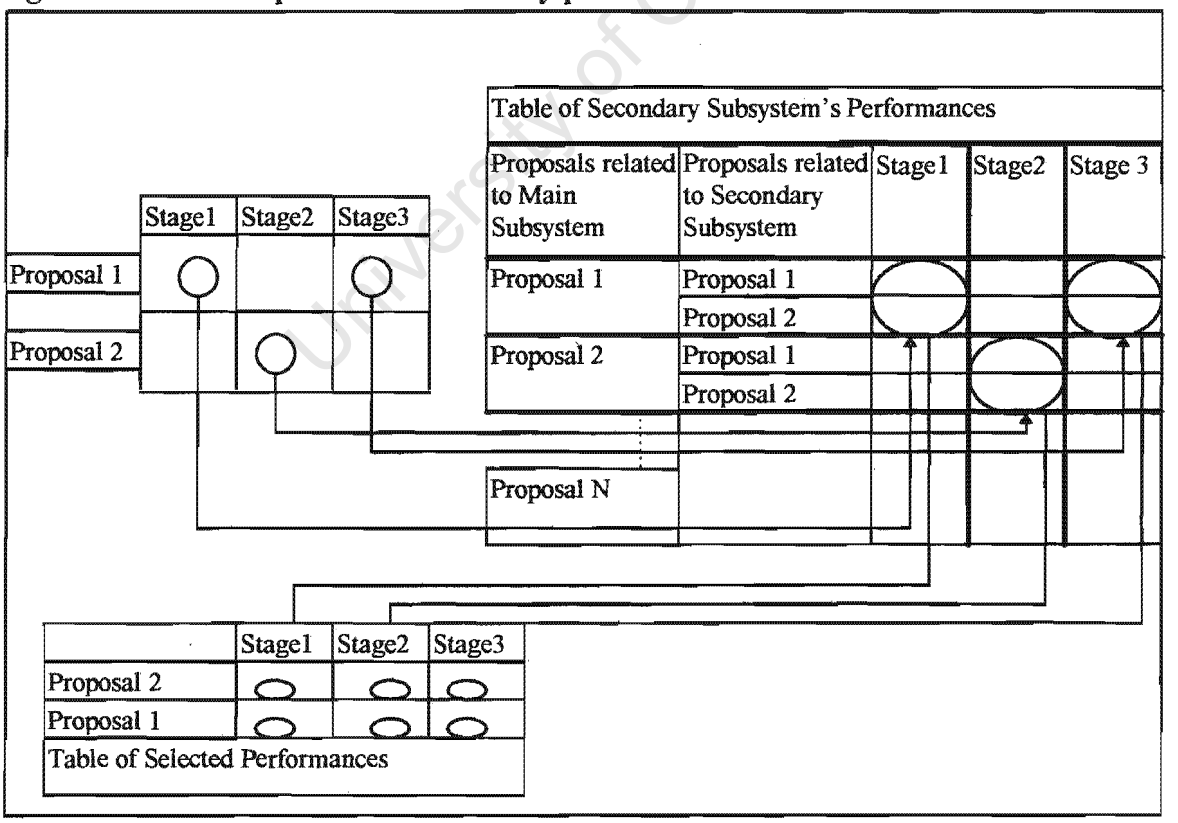
A- 1.1.1 Selection of performances

Capabilities are those attributes which define a system's performance. The performance value of the Main Subsystem has the most important role in the selection of the input elements to the attributes of the Main and Secondary Subsystems. Optimised performances of the Main subsystem select input elements to the Dependabilities and Availabilities of the Main and Secondary Subsystems and Capability of the Secondary

Subsystem. Therefore, the selection of performances of the Main Subsystem differs from the selection of performances of the Secondary Subsystem.

The designer selects the performances of the Main Subsystem (Main.xls file). The user manually inserts the values of the performances (Revenues) and their respective resouce requirements (Investments) to the 'Table of Main Subsystem's Performances' (cells B11-K15 of Main.xls). However, the selection of the performances of the Secondary Subsystem (Second1.xls) is influenced by the previously optimised values of performances from the Main Subsystem. There are two proposed values of performances of the Secondary Subsystem per each performance value of the Main Subsystem (Table of Secondary Subsystem's Performances - cells B32-K46). The procedure of secondary performances selection, which connects Main.xls with Second1.xls, is programmed in the 'Table of Selected Performances' (cells B11-K15 of Second1.xls). This is shown schematically in figure A1.

Figure A1 Selection process of secondary performances



A.1.1.2 Optimisation of performances

The procedure which optimises performance values is programmed on Delphi (Capability Module). The programme is divided into three sections:

- Connection to the static part of the programme (extraction values from Excel)
- Programming dynamic programming procedure using Delphi codes
- Transfer of outputs from Delphi to Excel.

Connection of Excel to Delphi uses the DDEClient method (see Delphi User's Guide). DDE codes allow dynamic exchange of information from any cell of Excel to the Delphi programme. In the case of a change of value on the Excel spreadsheet, the same value is updated simultaneously on the Capability Module. This type of arrangement allows the designer to share variables with other programmes or spreadsheets without breaching integrity of the computer programme on Delphi. The performance values are then optimised according to the dynamic programming approach used in H.A.Taha[7] .

The Dynamic Programming consists of stages in which the given Investment is allocated and respective Revenues are optimised (at present, additive maximization and minimization is used - H.A.Taha[7]).

The optimization process is initiated with calculations of the first stage (the forward recursive procedure). The first stage is unique because only sorting of the optimal revenues takes place. Therefore, the first stage forms a separate computational part in the programme.

The subsequent stages are programmed as one object (one subroutine). This sorts cumulative values of Revenues coming from the previous stages. To find the optimal Revenue of each stage, except the first one, the calculation of the dynamic programming formula (figure A-2) is repeated as many times as there is a number of stages minus one. The formulas for the first and the subsequent stages is given in figure A-2:

Figure A-2 Formulas used for Dynamic Programming optimisation

x - allocated fraction of the Total Investment (of the Control Input Value)

$f_i(x_i)$ - optimal return from a stage i at x_i

$R_i(k_i)$ - Revenue of alternative proposal k_i at stage i

where $x_{i-1} = x_i - c_i(k_i)$

$c_i(k_i)$ - Investment of an alternative (proposal) k_i at stage i

f1(x1) = max{R1(k1)} - formula for the first stage

fi(xi) = max{Ri(ki) + $f_{i-1}(x_{i-1})$} j = 2 to N - formula for subsequent stages

The final and most important phase of dynamic programming is to find the critical (optimal) path. The critical path is found in the same way as optimal performances (Revenues) but in a reverse order (H.A.Taha[7]). The general formula for the evaluation of the critical path (optimal path) is given in figure A-3, below.

Figure A-3 General formula for evaluation of the critical path

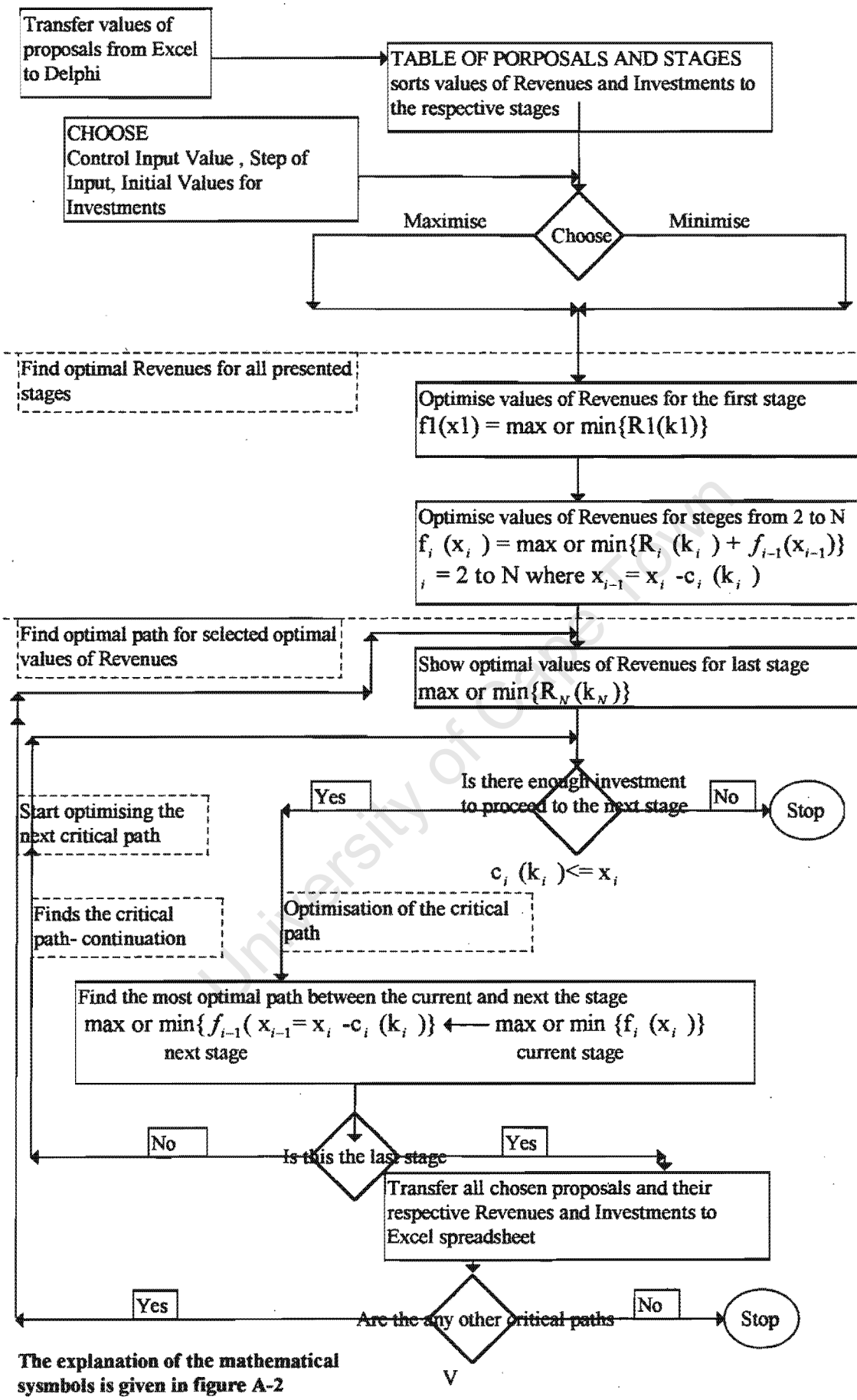
$$\max \text{ or } \min\{f_{i-1}(x_{i-1} = x_i - c_i(k_i)\} \quad \max \text{ or } \min\{f_i(x_i)\}$$

next stage — current stage

(for the explanation of symbols see figure A-2)

The subprogramme which optimises the critical path is divided into three parts. The first part optimises the final stage. This stage is significant because the last row of the table, which contains optimised performance values, has the most optimal value of Revenue. The second part of the subprogramme contains the optimisation code for all of the stages which are between the last and the first stage. The third part optimises the first stage which is unique because it has a different mathematical formula (see figure A-2). The listing of the dynamic programming programme is shown in Appendix C. The computer algorithm used for dynamic programming is shown in figure A-4:

Figure A-4 Algorithm for the dynamic programming



The explanation of the mathematical sysmbols is given in figure A-2

Once the optimisation procedure is finished, all optimal performances (Revenues) and their respective resources (Investments) are transferred to the Excel spreadsheet (Main.xls and Second1.xls). Again the DDE method connects Delphi with the Excel cells.

A-1.1.3. Transformation of performances into Capabilities

The optimised performances are compared with the ideal performance in the Table of Capabilities of the Main Subsystem or Table of Capabilities of the Secondary Subsystem (cells C55-G62). The Capabilities are expressed as the ratio of the ideal performance of a given stage to the optimised performance, from the dynamic programming, in the same stage.

A-1.1.4 Problems in the programming of the Capability attribute

Limit of the nested 'IF' statements in the Excel spreadsheet

Unfortunately, the maximum number of nested 'IF' loops that can be used in the Excel spreadsheet is seven. Therefore the maximum amount of proposed performances of each stage should not increase above seven. However, the author uses a maximum of five because his example does not need more proposals.

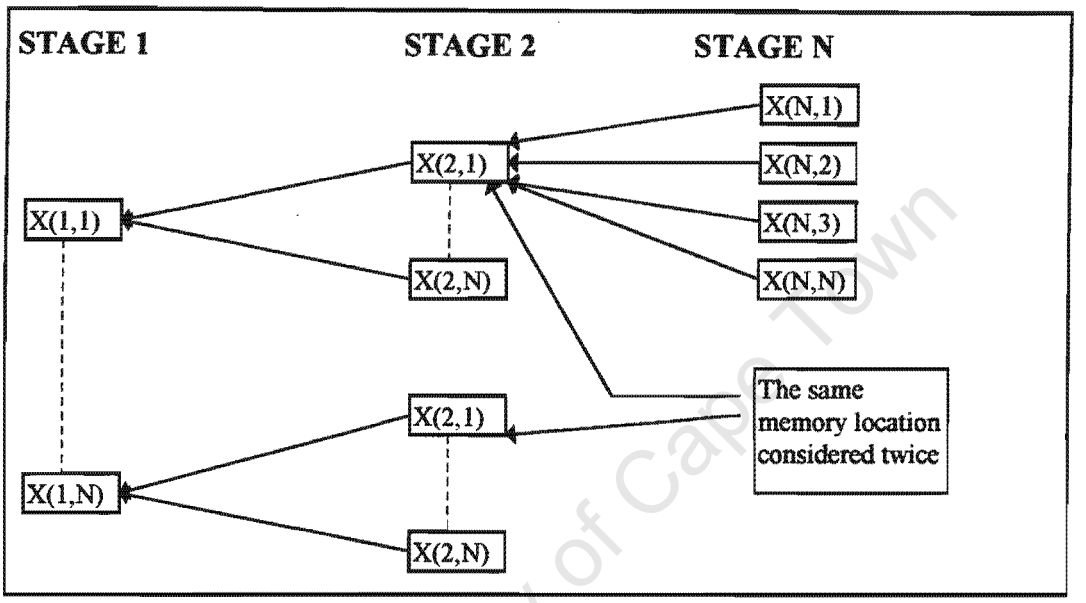
Dimensionality of dynamic programming

The dimensionality of dynamic programming creates problems. These are recorded by G.C Dandy & R.F. Warner[9] and W.L. Winston[12]. These researchers indicate that an increase of state variables would increase the computational effort. For example,. ten proposals at each stage of a four-dimensional state would increase the number of feasible states from ten to ten thousand. Therefore, to simplify matters, the author uses a one-dimensional array with a one-dimensional state variable.

Problem with arrays for automatic critical path

Theoretically, it is possible to form an automatic critical path finder. This, however, requires that the dimension of the array would be equal to the stage number plus one. Otherwise, the memory location will be overwritten by new information. This problem is illustrated in figure A-5 below.

Figure A-5 Problem of memory location for automatic critical path finder



The above figure shows a schematic optimal path from Stage N to Stage 1. As the number of stages increases, so the number of repeated memory locations increases in each stage. In order to have an automatic calculation procedure for finding the optimal path, the dimension of the array (but not the size) increases according to the formula:

dimension = stage + 1.

This is because all the values are needed in advance before the optimal path may be calculated. The above example illustrates a two-dimensional array limitation, where the memory location X(2,1) repeats more than once. Thus, the contents of this array is overwritten each time it is called. Therefore, the calculation for the optimal path may give completely unpredicted values. The only exception is stage 1, but that is the trivial solution anyway.

The only way to solve this problem is to use a semi-manual procedure. In this case the operator is given the calculated optimal values of a stage from which he/she chooses the most optimal one (this is the case when more than one optimal path is found by the computer programme). The chosen optimal values of the previous stage are then used by the computer programme to search for the set of next optimal values for the current stage.

The optimal value of performance, which the operator chooses, is instantaneously written into the Excel spreadsheet (Table of Critical Path in Second1.xls). The code for a critical path optimisation is given in Appendix C.

Probabilities of finding non feasible regions during the optimisation procedure

During development of the dynamic programming code, several conditions with non feasible regions may be found. Therefore, during the modeling process, the designer should be aware of the following situations which could result in the occurrence of non feasible regions. These are:

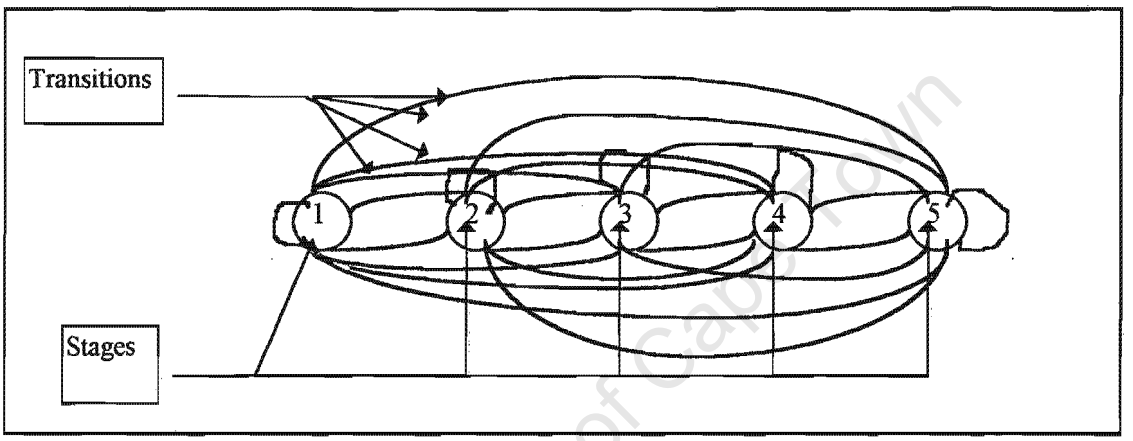
- The Investment of a proposed Revenue does not exceed the amount of the Investment allocated to every particular stage
- The difference between the Investment allocated to a given stage and the Investment of a proposed Revenue is less than the initial value of the allocated Investment.
- None of the differences between the Investment allocated to the given stage and the Investments of the proposed Revenue is equal to the previous stage of proposed Investments.
- There are more than five critical paths (reason explained earlier in the text).

A-1.2. Programming the optimisation of the Dependability attribute

A-1.2.1 Selection of the transition rates

The next optimised attribute is Dependability. Since the programme can optimise a maximum of five stages, therefore, the maximum number of possible transitions is twenty-five. See figure A-6 below.

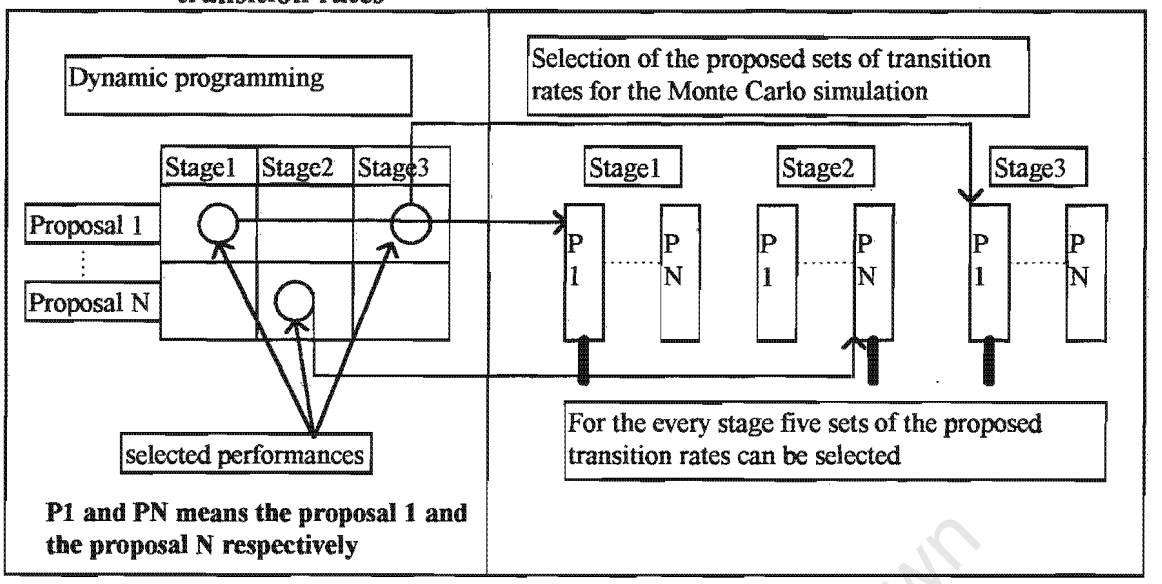
Figure A-6 General State-Space model for the Main and the Secondary Subsystem



The Excel spreadsheet sorts the sets of transition rates (Table of Proposed Transition Rates in Second1.xls and Main.xls - cells BE3-BZ91). Each set of transition rates characterises the probability of transition.

The input elements (i.e. performance values) of the Capability attribute of the Main Subsystem (Main.xls) selects sets of transition rates of the Main Subsystem and the Secondary Subsystem. Figure A-7 shows the connections between the optimal values of performances (selected in the Main.xls) and the respective sets of transition rates (Second1.xls and Main.xls).

Figure A-7 Connections between optimised values of performances and the sets of transition rates



A-1.2.2 Simulation of the transition rates and the optimisation of reliabilities

The previously selected sets of transition rates are transferred to the Dependability Module in Delphi. There, the Monte Carlo method simulates these transition rates. The result is an average transition rate, which characterises one transition between stages or the same stage.

The modelling of the simulation process consists of the following steps:

- There is a selection of a representative range of values out of a possible random sample. The size of the representative range depends on the size of samples (see figure A-8). The values of the ranges are based on the past outcomes and may characterise the nature of an environment. However, the final outcome is not known as the pattern of the sample occurrence is unknown.

- The limits of representative ranges are created from the mid-points between step values. The step values are the intervals derived from the biggest value of the sample

(i.e.the biggest value divided by the number of representative ranges). In the case of the sample smaller than 20, all the steps are the same and no mid-points are used. Twenty ranges are used to decrease the amount of computation.

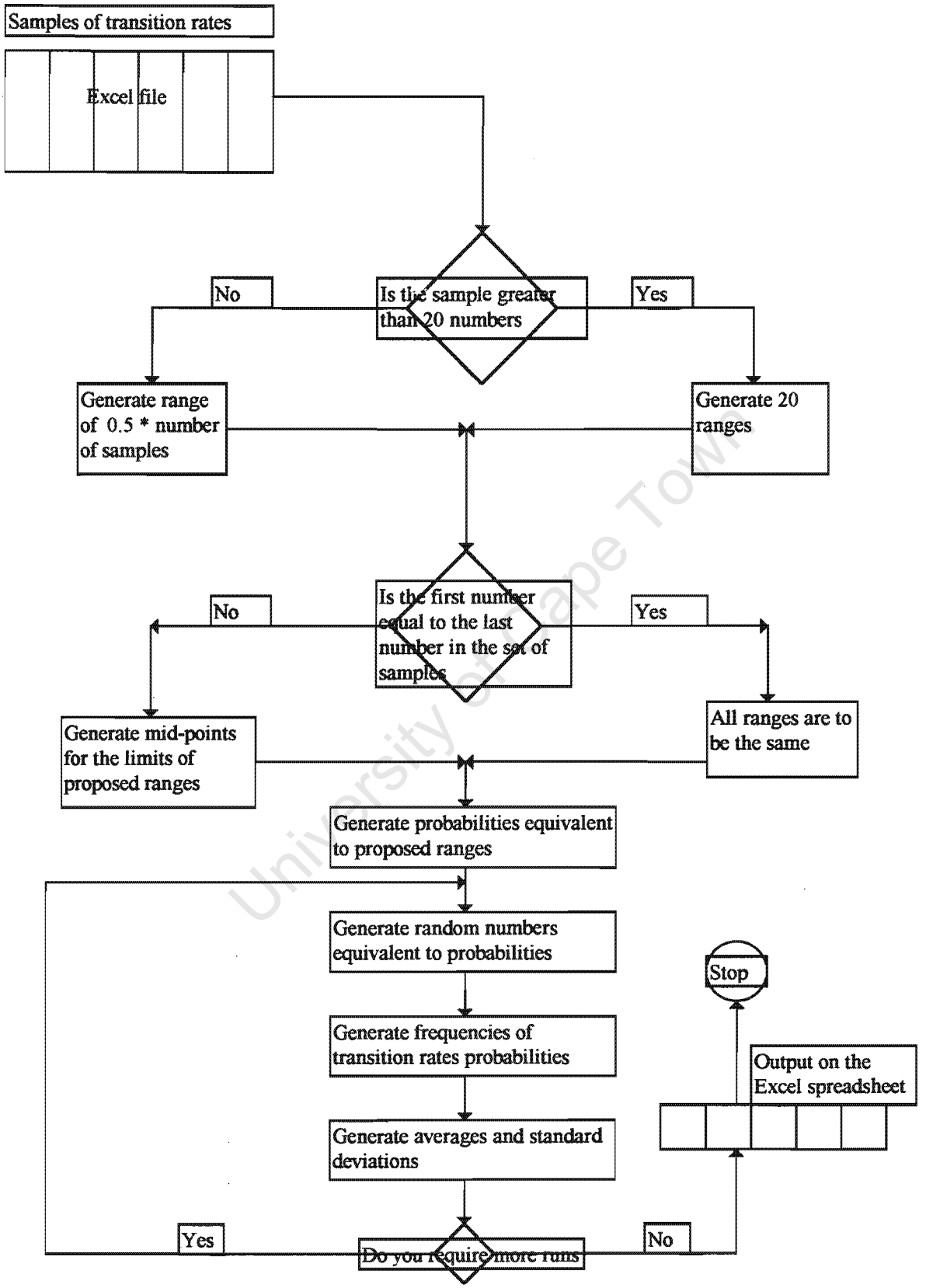
- The random numbers generate the possible outcomes of probabilities.An equivalent random number is assigned to the evaluated probability of the transition rate occurrence. After running the simulation (usually 70 simulation trials because of operational memory limitations) the frequencies of transition rates are calculated.

- Finally, accumulated averages of simulated transition rates are calculated to smooth oscillations between the final values of these rates. A standard deviation checks the deviation of the last five (half of the minimum number of simulation trials) averaged transition rates.

- The DDE method transfers the results of the simulation of transition rates to Excel (Main.xls and Second1.xls cells B107-U116).

Figure A-8 shows the algorithm to programme the Monte Carlo simulation process.

Figure A-8 Algorithm for the Monte Carlo simulation



The simulated values of transition rates are finally manipulated by the exponential equations of reliabilities. The evaluation of reliability has the following form:

$$\text{Reliability} = e^{-(\text{time})\cdot(\text{transition rate})}$$

The reliability itself can be further optimised by the selection of the appropriate value of time.

A-1.2.3 Evaluation of Dependability attribute

The value of the Dependability attribute is the same as the value of reliability.

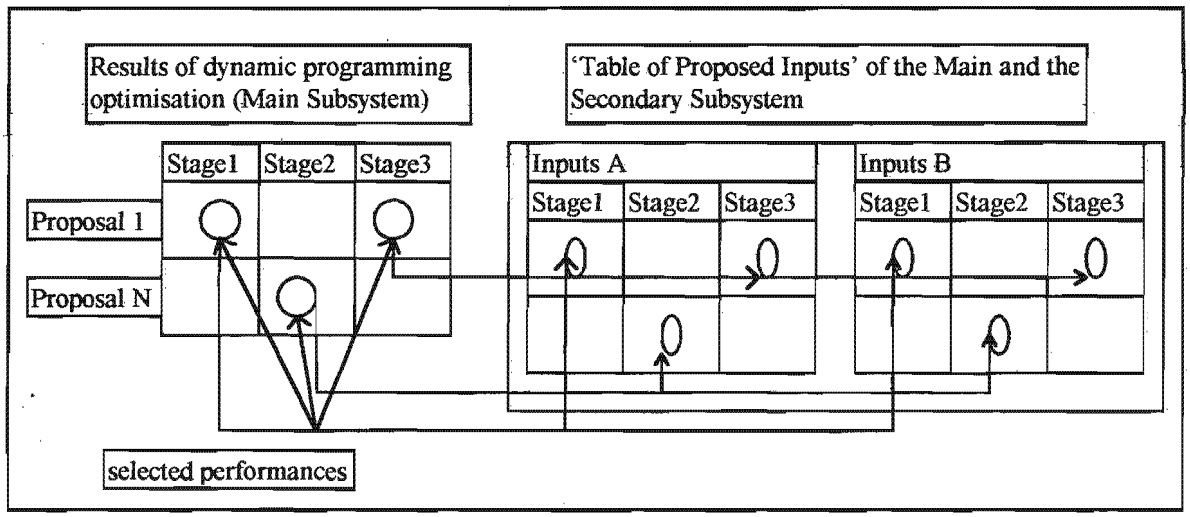
A-1.3 Programming the optimisation of the Availability attribute

A-1.3.1 Selection of Inputs A and B

The selection procedure of Inputs A and B of the Main (i.e. Main.xls) and the Secondary (i.e. Second1.xls) Subsystem is programmed on the Excel spreadsheet.

The Inputs A and B related to the value of proposed performance of the Main Subsystem are stored in the 'Table of Proposed Inputs'. After dynamic programming optimisation of the performances in the Main Subsystem, these optimised performances select the respective Inputs A and B from the 'Table of Proposed Inputs' (cells O148 - Y152). Thus, the optimised performance of the Main Subsystem selects the Inputs A and B of the Availability attribute of the Main and Secondary Subsystem.

Figure A-9 Selection of Inputs A and B



The selected Inputs A and B are transferred to the 'Table of Selected Input Values' (cells B149 - L158).

A-1.3.2 Optimisation of Inputs A and B

The Fuzzy Logic method optimises Inputs A and B for the Availability attribute. This method takes a substantial amount of programme space because every condition of Availability is defined by 'IF ' statements (see section 3.3 and Appendix E - Availability Module). Three independent sets of three triangles describe Input A, Input B and Output functions. These three triangles are sufficient to describe the general nature of Availability (Low, Medium and High) of any subsystem.

The geometry of the triangles is isosceles. This type of geometry simplifies the computational design of the Fuzzy Logic. The mid-points of the triangles' bases are evenly spaced. Therefore, the processing of input values is the same for Input A and B. It means that no special computational method is needed to account for triangular dissimilarities of the input triangles. The mid-point of one triangle is the end or the beginning of the other one.

A triangle offers unique properties for the Fuzzy Logic. It has only one tip value (target value) which points on a number that is 100% relevant to the subjective description made by the observer. For example, a temperature of 20 ^{0}C may be described as 100% low according to some specifications. However, a temperature of 23 ^{0}C which is still low, but is not the target value, may be identified as 70% close to the one described by the observer. Thus, the isosceles triangles are the simplest geometrical figures to model the fuzzy regions.

The triangular membership function has the form shown in figure A-10 (T. Sudkamp & R..J. Hammell II[23]).

Figure A-10 Triangular membership function

$$\mu_{A_i}(x) = \begin{cases} (x - a_i) \div (a_i - a_{i-1}) & \text{if } a_{i-1} \le x \le a_i \\ (-x + a_{i+1}) \div (a_{i+1} - a_i) & \text{if } a_i \le x \le a_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Where

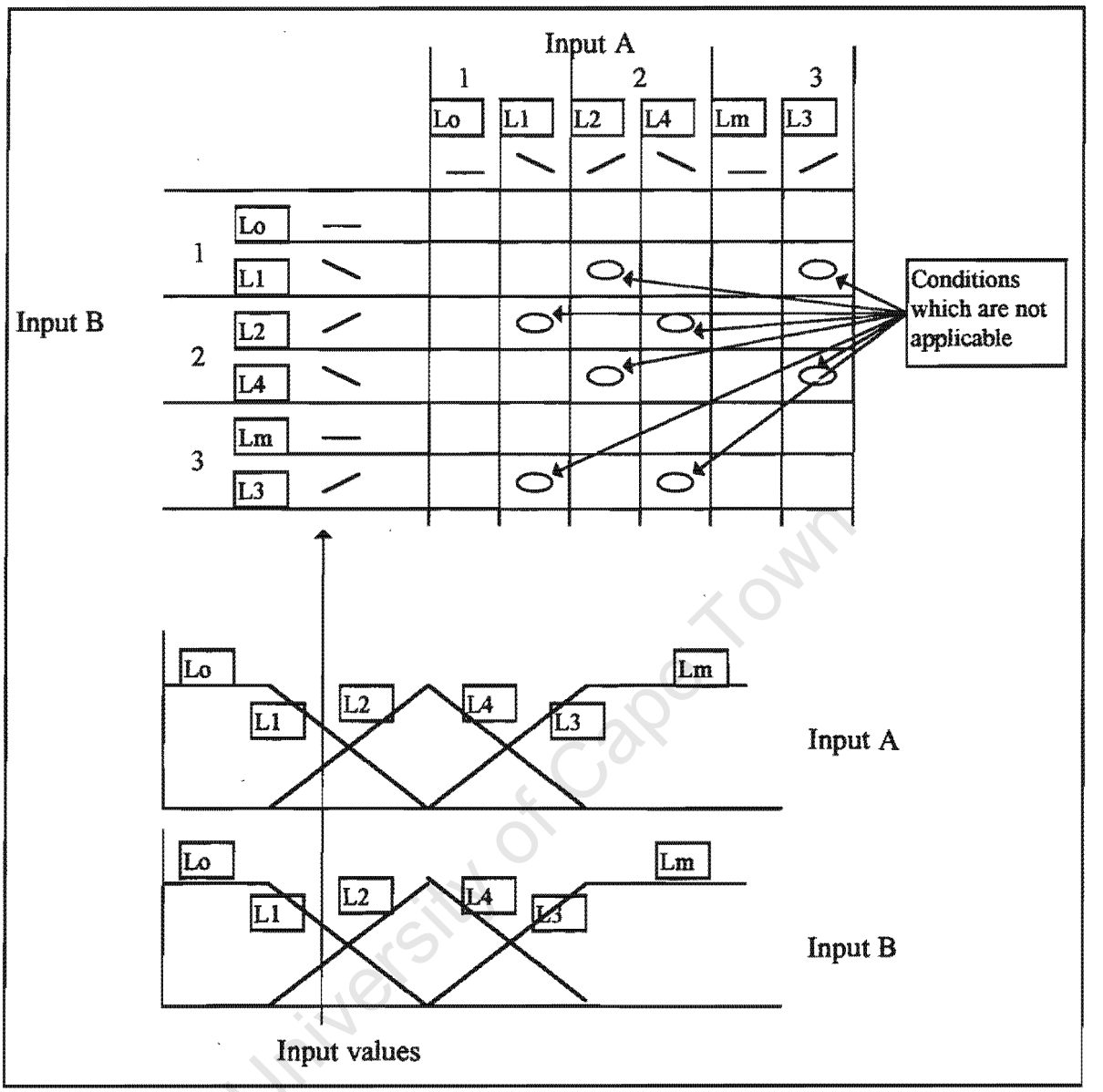
x - Input value

a - unique domain element (e.g. start, end or middle of the base of a fuzzy triangle

μ_A - membership function of the fuzzy triangle (i.e. hight of the fuzzy triangle)

Since three triangles are used to model input functions, therefore, the computation of rules requires nine conditions. However, different combinations of slopes (lines on which the input value can be present) results in thirty six 'IF' conditions. In the cases where slopes of the triangles are of opposite signs, the conditions are not applicable. Thus, only twenty-eight 'IF' conditions eventually need consideration. Figure A-11 illustrates the conditions in Fuzzy Rule.

Figure A-11 Conditions in Fuzzy Rule

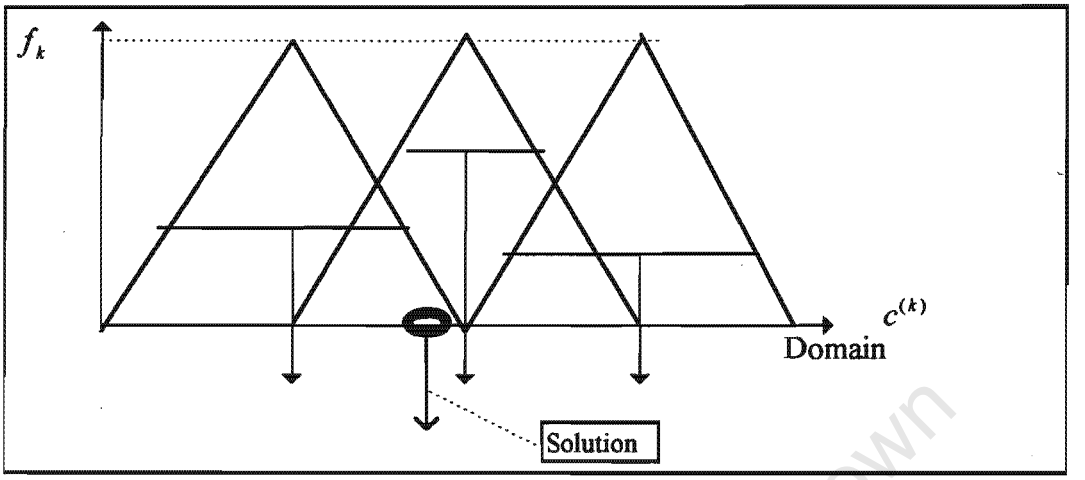


According to figure A-11 only the $L1^{Input\ A}$,$L1^{Input\ B}$ membership value can be considered. Membership values coming from $L1^{Input\ A}$,$L2^{Input\ B}$ and $L2^{Input\ A}$,$L1^{Input\ B}$ are prohibited (see text above).

Finally, the defuzzification procedure commences of which the quickest and the simplest is the Height Defuzzification method. There are other defuzzification methods, but they are

usually much slower and more complex (D.Driankow, H. Hellendoorn, M. Reinfrank[14]). Figure A-12 shows a principle for the Height Defuzzification method.

Figure A-12 Height Defuzzification method



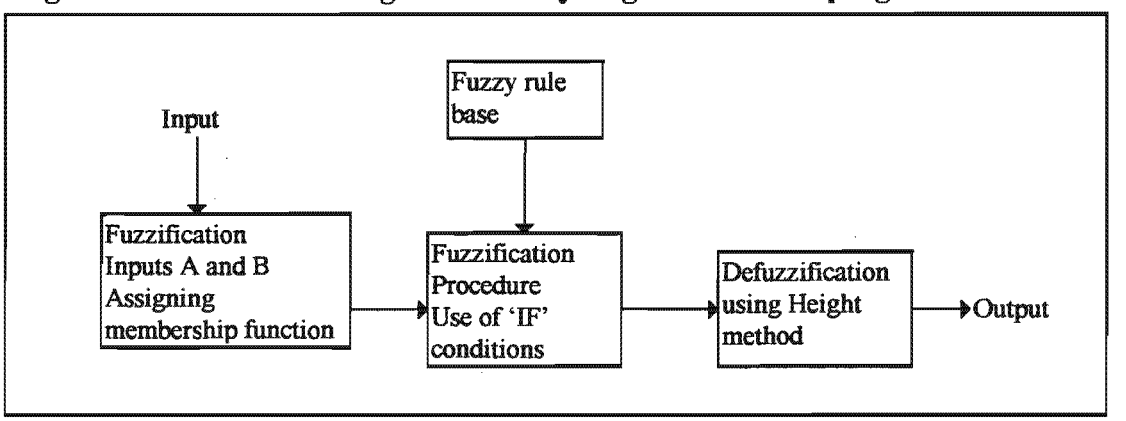
This method takes peak values of each triangle and builds the weighted sum of these peak values. The Height method does not depend on the shape or base of the triangles. The defuzzification output is denoted by the following formula:

$$u = \frac{\sum_{k=1}^{m} c^{(k)} \cdot f_k}{\sum_{k=1}^{n} f_k}$$

f_k - height of the triangle
c^k - peak values of the triangles (i.e. Targets)
m - system of rules
u - output
k - mid - point number

Figure A-13 shows a diagram of the Fuzzy Logic method used in the programme.

Figure A-13 Schematic diagram of Fuzzy Logic used in the programme



A-1.3.3 Transformation of the Fuzzy Logic Output into the Availability attribute

The table 'Output Values'(cells B165-F174) represents outputs of the Fuzzy Logic optimisation. The Output values are finally converted into qualitative ratios of Availabilities (see subsection 2.3.1(c)).These qualitative ratios of Availability values are evaluated in the table 'Availabilities' (cells H165-L174).

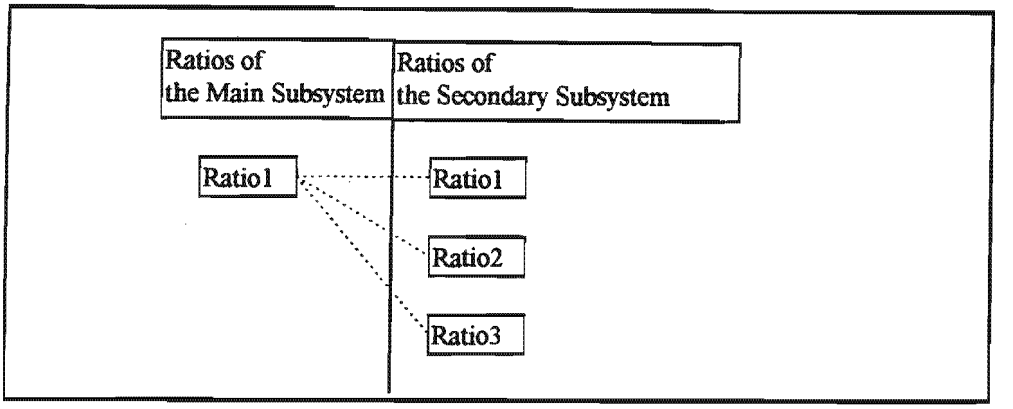
A-1.4 The final display of the results of the subsystem's and system's optimisation

The final results of every subsystem's optimisation are shown on the Excel spreadsheet. Values of the Capability, Dependability and Availability attributes are multiplied together (see section 2.3.1) to evaluate the effectiveness of the subsystem.

The values of the Availability and Capability attributes are expressed in the form of a vector and values of Dependability are in the form of a matrix. Therefore, the matrix multiplication of the three attributes will evaluate effectiveness of the system. Since the Excel spreadsheet has matrix manipulation facilities, it is unnecessary to design an additional Delphi code to perform the task of the matrix multiplication.

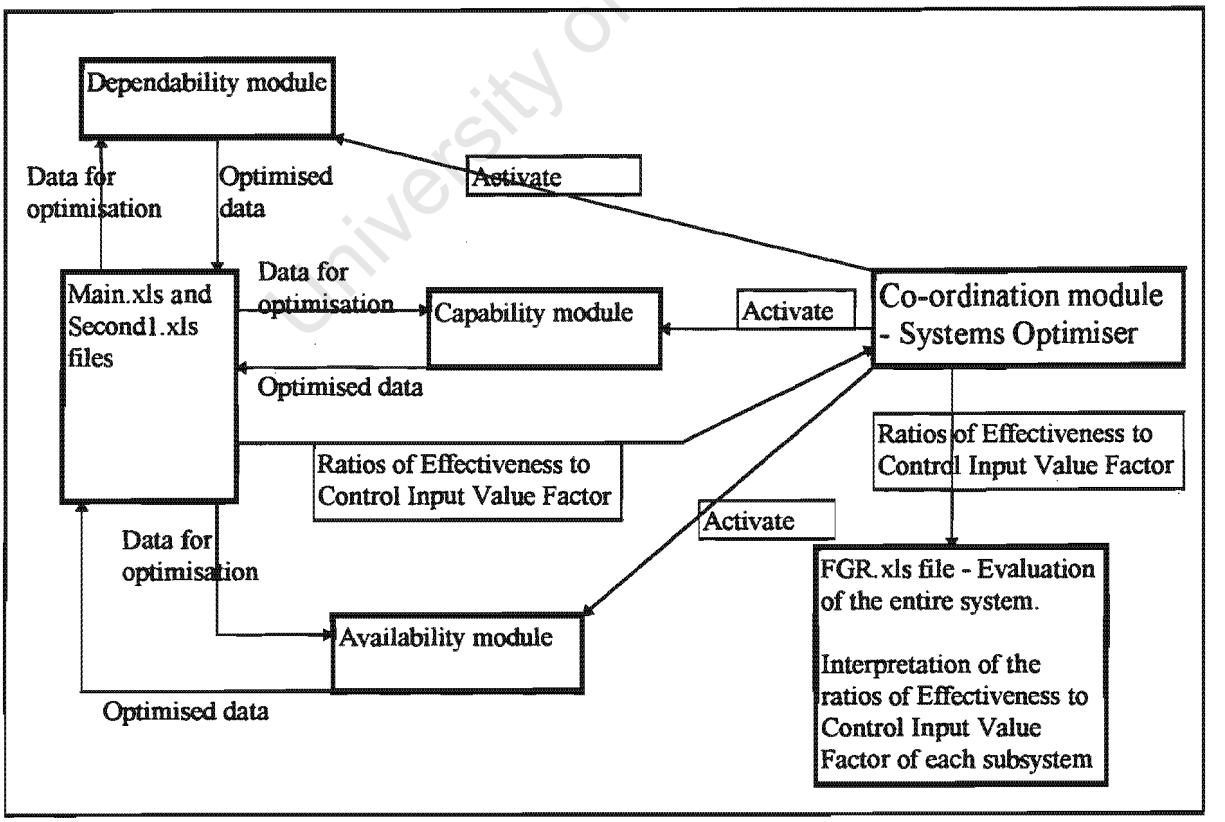
Finally, effectiveness values combine into the ratios of Effectiveness to Control Input Value Factor (see section 4.1.2). The ratios of Effectiveness to Control Input Value Factor, and the subsystem's three attributes of the Main (Main.xls) and the Secondary (Second1.xls) Subsystem are displayed in cells A214-M276. The relationship between the ratios (ratios of the Effectiveness to Control Input Value Factor) of the Main and the Secondary Subsystem are presented in FGR.xls (cells A12-G27) and the general representation of this relationship is shown in figure A-14.

A-14 General relationship between the ratios of the Effectiveness to Control Input Value Factor of the Main and the Secondary Subsystem



The Delphi modules and Excel spreadsheets are activated by the co-ordination module - Systems Optimiser. This module controls the flow of the data (i.e. ratios of Effectiveness to Control Input Value Factor) from the Main.xls and Secon1.xls files to the FGR.xls file. The listing of the programme for this module is shown in Appendix C. The role of this module in the optimisation of the whole system is shown in figure A-15.

Figure A-15 General structure of the programme development



APPENDIX B: Manual for the Systems Optimiser software

B-1. Getting started

The following must be done before the System Optimiser is activated:

1. Windows 3.x or higher must be installed.
2. The full Excel spreadsheet **must be** installed in the directory **C:\EXCEL**. Since all excel programmes, attached to the System Optimiser, were written in Excel 4, it is advisable that the Excel 4 is the chosen version. Please note, that the Excel in a Microsoft Office or the Excel activated from a network will **NOT** be able to connect to the System Optimiser software.
3. The AUTOEXEC.BAT must include the following:

PATH C:\IDAPI;C:\EXCEL;%PATH%

OR

insert the: C:\IDAPI;C:\EXCEL; in already available path.

PLEASE NOTE, THAT THESE CHANGES MAY BE DONE BEFORE OR AFTER SYSTEM OPTIMISER INSTALLATION

The programme is activated by opening the FO - directory and pressing the sysan.exe file. Thereafter, the main window of the software is activated (Systems optimiser window) and simultaneusly the dialogue with Excel is initiated.

First, Excel asks the user to 'Re-establish remote links' with the system Optimiser package. Press 'yes', after which the links are activated.

Finally, Excel will ask the user to 'Update references to unopened documents'. Press yes again, after which the channel of communication between the Excel spreadsheets is activated.

B-2. Entering values used for optimisation

The values are already inserted into the programme. However, if the user wishes to change the values, then it is necessary to follow the instructions below.

Before commencing optimisation, the user has to enter input values relevant to optimisation of three attributes (capability, dependability, availability). These input values are called input elements to the subsystem (See Chapter 2) and characterise the nature of the above attributes (capability, dependability, availability). Due to the theoretical model, the same input elements are present in the Main subsystem (Main.xls) and Secondary subsystem (Second1.xls). In both spreadsheets (i.e. Main.xls and Second1.xls), in the light blue areas, are values which can be changed and in the yellow areas, values which can not be changed by the designer.

The user may start entering values to the Main (primary) subsystem. Therefore he/she must open Main.xls file by simply pressing [Tab] and [Alt] buttons simultaneously (the user's proficiency to operate Windows 3.x is assumed).

Inputs to Performance

In the Main.xls find the Capability section and particulary the 'Table of Main Subsystem's Performances' (the table starts from cell A8). Now, the user can insert values of the main subsystem's performances in the 'Revenue' column of every stage (eg. STAGE1). The relevant resource required to maintain performance is entered in the 'Investment' column.

In the 'Control Panel for Calculations of Capabilities' (the table starts from cell A3), the user has to insert the varable which can alter the dynamic programming optimisation

procedure. The user enters the value of the Control Input Variable, which represents the highest total resource allocated to all stages of the analysed system (must be non-negative in case of maximisation and less then 10^8 during minimisation), and the Step of input value which is a user's defined minimum resource allocated for any performance. Special attention should be given to the choice of the Step of input value. This must be a factor of the Control Input Value (e.g. if the Control Input Value is 10 then possibilities of the Step of input are e.g. 1, 2 or 5; if the the Control Input Value is 2.1, the possibilities of the Step of input are e.g. 0.1, 0.3 or 0.7). Of course, the lower the Step of input, the greater is the chance of finding more values of optimal performance, however, the computational effort is increased. The total number of steps (the number of steps is equal to the Control Input Value divided by Step of input) must not exeed 50. The Control Input Value and Step of input can also be altered in the Dynamic Programming module.

The last value which must be entered by the designer is 'Minimum total investment value'. This value represents the minimum resource which could be allocated to all stages. 'Minimum total investment value' is used to calculate the Control Input Value Factor for the whole subsystem.

Inputs to Reliability

As it was described in the theory, reliabilities are the results of transition rates and time in which the transitions occur. Therefore, the user will have to insert transition rates to the 'Table of Proposed Transition Rates' (the table starts from BF1 cell).

The headings of the columns represent the transition between two stages of the system (e.g. R_12 represents transition from stage 1 to stage 2). Therefore, sets of transition rates which represent certain types of transition must be inserted into the columns which represent those transitions. The maximum size of the sample of transition rates can be ten in every column.

At the top of the 'Table of Proposed Transition Rates' one can see the row with heading 'Dependent Transition Rates'. This row represents those transitions which are calculated

by the Excel spreadsheet and they are not inserted by the user. However, every stage may have only one dependent transition. (e.g. three stage system will have maximum three transitions per stage, so only two samples of transition can be inserted per stage).

The Time is another element which characterises Reliability. The value of time must be inserted in the 'Table of Dependabilities' (the table starts from the A117 cell). For every expected option of transition rates a value of time should be inserted in the column called 'Control Input Value'.

Inputs to the Support Design

The Inputs to the Support Design are inserted in the 'Table of Proposed Inputs' (the table starts from the cell N142). The user must be aware of the fact that for every stage, two characteristic Input values (Input A & Input B) must be inserted.

The Fuzzy Rule must also have an initial set-up so the interrelation between the two input values and the output is relevant to the system's support requirements. The set-up of the Fuzzy Rule is done in the 'Table of Fuzzy Rule' (the table starts from the cell C186).
This is shown in figure B-1 below:

Figure B-1 Fuzzy Rule Table

LOW =	1
MEDIUM =	2
HIGH =	3

INPUT B	INPUT A		
	LOW	MEDIUM	HIGH
LOW	2	1	1
MEDIUM	3	2	1
HIGH	3	3	2

The values in the blue cells are the outputs which are the results of the interrelationships between input values. These values can be altered by the designer. For example, if Input A will be subjectively evaluated as 'medium' and Input B will be regarded as 'low', then the user claims that the output value will result as the 'low' value (therefore he/she entered the number one).

If all the required information has been inserted, the user may minimise the Main.xls file window (the user's ability in window manipulation in Excel is assumed) and enter the Second.xls file window (start entering values to the Secondary subsystem).

The entered values to the Secondary.xls are the same as in the case of the Main.xls. The only slight difference is in the insertion of the performance values. The values of performance of the secondary subsystem are dependent on the selection of performances in the Main subsystem (see Chapter 5 for explanation).

Therefore, the user must insert performance values which will be related to the values of the main subsystem's performances. The 'Table of Secondary Subsystem's Performances' starts from cell A28 (e.g. in The Table of Secondary Subsystem's Performances ,under the heading "Stage 1", the first two proposals of Revenues and Investments are connected with the first proposal of Revenue and Investment of Stage1 in the Table of Main Subsystem Performances).
The maximum number of proposals is two per stage (two proposals from the secondary subsystem are related to one proposal of the main subsystem).

The rest of the procedure of entering information is the same as in the Main.xls file.

After finishing entering the required information, the user can move back to the main window (Systems Optimiser window) by pressing the [Tab] and [Alt] buttons simultaneously (Sysan should appear).

B-3. Commencing the optimisation procedure

The optimisation usually starts with the analysis of the Main Subsystem (Main Sub). In the main window, press the [Main Sub] button. Three boxes representing three subsystem's attributes appear (capability, dependability, availability). In the left corner of the window, the name of the currently analysed subsystem is shown.

The optimisation of the main subsystem starts with the analysis of the capability attribute. By double clicking the capability box, the dynamic programming module is activated.

Modeling Performances (Capability Section)

In the Control Panel, the user finds information about the number of columns and number of rows. The number of columns informs the user of the maximum number of columns in the Table of value. By dividing the number of columns by two, the user can calculate the number of stages. Then, the number of rows tells the designer about the number of proposed performances and resources for every operational stage of a system.

Also in the Control Panel of the Dynamic Programming window, go to the green box called Optimise. In the Optimise box, click on the type of the dynamic programming optimisation which is relevant to the modeling procedure (i.e. Maximise or Minimise). Next, insert the Control Input Value (must be non-negative in case of maximisation and less then 10^8 during minimisation) and the Step of Input in the indicated boxes (if necessary, these values may be altered to serve the purpose of optimisation). The user must remember that the Step of Input is the unit step which divides the Control Input Value into equal steps. If the Control Input Value is divided by the Step of Input, the number of the steps can be calculated. The maximum number of steps must not exceed 50.

In order to accept the inserted information press the [Accept CIV] button. Press the [Open Link] button to transfer the values of performances and resources from the Excel

spreadsheet to the Dynamic Programming module. These values can be seen in the 'Table of Values'.

Press the [Stage1] button to initiate calculations of the first stage in the dynamic programming. The optimised values of the first stage are seen in the table called 'First Stage'. The allocated steps of the Control Input Value are shown in the 'Allocated input' column and respective optimised values of performance are shown in the 'Optimal' column.

Press [Stage N] to optimise the subsequent stages. The values of the optimised performances of the subsequent stages are shown in the table called 'Subsequent Stages'. Press the [Initialise Path] button to start Critical Path Optimisation. In the Critical Path window, the most optimal proposal of performances of the last stage are shown. Next to every number of proposal, is the amount of the resource left after allocation of the proposal to the particular stage.

Press [Optimise path] and the Input Box appears where the user inserts the number of the required proposal of performances (i.e. Revenues and Investments of the optimised stage). After inserting the number of the required proposals, the user has to press 'OK'.

The user can select the numbers of proposals (numbers of optimal proposals are given for every stage) only from the 'Critical Path' window. For example if in the stage 2, proposals 1 (i.e.Prop1)and 4 (i.e. Prop4) are displayed as optimal, the user can only choose one proposal from the two (i.e. proposal 3- Prop3 is not permitted). Also in the window 'Res left' represents the amount of the resource which is left for currently optimised stage. (e.g. if there is more than one stage which has not been optimised yet then the 'Res left' represents the total value of the resource left for all non optimised stages)

The above procedure is repeated until the following message appears in the box - 'Do you want to continue ?' Insert 'N' if optimisation is finished or press 'Y' if the optimisation has to be continued (i.e. not all combinations of the critical paths have been identified).

Assuming that optimisation has been completed and 'N' has been inserted into the box, the user has to click 'OK' after which the box disappears. All selected values of performances (Revenues) and their respective resources (Investments), as well as the names of the chosen proposals, are shown in the table 'Critical Path'.

If one wants to start new calculations, by simply pressing the [New] button, old data are erased giving space for new values. If one wants to finish optimisation, the [Close] button returns the user to the main window (Systems Optimiser window).

Modeling Reliability (Dependability Section)

In the main window (Systems Optimiser), the user should click on the [Dependability] box to simulate the transition rates. The Monte Carlo Simulation window appears.

In the Control Panel, the user has to establish both the total number of the simulation trials in the [Trial] box and the time of transitions in the [Control Input Value] box. The rest of the boxes show the calculated values which characterise the optimised system. The information in these boxes cannot, and should not, be changed.

The 'Number of events' shows the number of transition rates. The 'Number of transitions' show the number of transitions between the stages of the system. The 'Number of paths' shows the total number of the optimised critical paths (from the dynamic programming). The optimisation is simply activated by pressing the [Activate] button.

In the 'Table of Average Transition Rates' two rows are present. The row with heading 'Average' displays the simulated average values of transition rates and 'Std Dev' shows values of respective standard deviations. If the deviation values are relatively big then the

[Activate] button should be pressed as many times as is necessary for the average values of transitions to be satisfactory. Otherwise, the number of the simulation trials must increase (unfortunately, the maximum number of trials can only be 70).

In order to see the final values of reliabilities, the user must check the Excel spreadsheet (Main.xls) and particularly the 'Table of Dependabilities' (the table starts from cell A117).

The table called the 'Table of Selected Transition Rates' (table starts form cell A77) contains imported valuesof transition rates (from the Excel spreadsheet). Both the 'Table of Selected Transition Rates' and the 'Table of Average Value of Transition Rates' (the table starts form A105) contain columns with symbols which represent the transitions from one stage to another (e.g. R_23 represents transition from stage 2 to stage 3).

In spite of the fact that simulation has been completed, the optimisation may still continue. If the 'Number of path' (this number represents the number of Critical Path form dynamic programming) is greater than one, then the path number must be increased to the next number (i.e. from 1 to 2) and the simulation must be repeated for the next critical path.

If the optimisation is complete, then the user may press the [Close] buton and move again to the main window (Systems Optimiser - Sysan).

Modeling support design (Availability Section)

The attribute left to be optimised is availability. By pressing the [Availability] box in the main window, the user is transferred to the Fuzzy Logic window.

Before the user starts the optimisation, he/she must insert the Target values and Space values for the InputA, the InputB and the Output. The Target value represents the tip of the first fuzzy triangle. The space value shows the equal distance between the other two tips of the fuzzy triangles. After inserting values for Targets and Space, the user can start optimisation.

In order to get only numerical results the user must press four buttons in the following sequence: the [Input A] , the [Input B], the [Fuzzyfy] and finally the [Activate] button. After that procedure, the output of three support values can be seen on the 'Output of Fuzzy Rule' window.

The [No. of The Path] box requires additional explanation. This box has a slider which has two arrows. By pressing the up-arrow, the No. of the Path value increases. The maximum number in the box will not exceed the number of critical paths optimised in the Dynamic Programming module. Similarly, the down-arrow decreases the No. of the Path value from maximum to one.

If the user wishes to see the graphical representation of the fuzzification of input and output values, then every stage must be checked separately. This means that the input values presented in the list boxes placed next to the Input A or Input B boxes must be chosen separately for each stage of the system. This is done in the following way: choose the number in the list box next to Input A and then press the [Input A] button. Do the same for Input B. Then press the [Fuzzify] and [Close rule/Defuzzify] button. The output value of the Input A and B will appear in the [Output] box. The top two graphs (next to Input A and B) will display the Input A and B values for a Target value, Space and the Input (yellow stripes). Press the [Close rule\Defuzzify] button. The graphical representation of the fuzzy output will appear (bottom graph).

If the user wants to check the Fuzzy Rule (Fuzzy Rule can be altered on the Excel spreadsheet cell B177), he/she can do it by pressing the [Open rule] button. The [Close rule\Defuzzify] button closes the Fuzzy Rule table.

Finally, the [Close] button closes the Fuzzy Logic module and returns the user to the main window (press [Alt] and [Tab] buttons to get to Sysan - main window).

Final values

At this stage the optimisation of the main subsystem is completed. The user can press [Activate] to see the ratio of Effectiveness to the Control Input Value Factor in the [Main Sub] table.

Secondary subsystem's optimisation

The optimisation of the secondary subsystem is almost the same as the optimisation of the main subsystem. The only difference is that after pressing the [Second1 Sub] button, not only three boxes of the subsystem's attributes appear, but also the 'Number of critical path' box appears on the screen. The maximum number which can be found in the box is equivalent to the ratio of Effectiveness to the Control Input Value Factor in the Main Sub grid table. The optimisation of the Secondary Subsystem follows the same procedures as those presented in the Main Subsystem. This must be repeated the number of times equal to the number found in the [Number of critical path] box (the slider in the 'Number of critical path' box. is used to change the value of the 'Number of critical path'). The ratios of Effectiveness to the Control Input Value Factor of the Secondary subsystem are related to the one ratio of Effectiveness to the Control Input Value Factor of the Main Sub table. The number in the 'Number of critical path' box tells the user in which row of the Main Sub grid table the ratio of Effectiveness to the Control Input Value Factor is present. For example, the number 2 in the 'Number of critical path' means that the values in the [Second Sub] table represent the value in the second row of the [Main Sub] table.

After changing the number in the 'Number of critical path' box, the user starts the optimisation of the secondary subsystem again (the same way as the main subsystem optimisation process). The optimisation is finished if the value of the 'Number of the critical path' box cannot be increased further.

In order to erase all information from the grids and start the analysis again, the user must press the [New calculations] button.

Before closing the programme, the user may see all of the results of the ratio of Effectiveness to the Control Input Value Factor of both subsystems in the Excel spreadsheet (Fgr.xls). Figure B-2 below shows how these values are displayed:

Figure B-2 Quality values for the Main and Secondary Subsystems

	Quality Values for the Main and Secondary Subsystems Effectiveness / Resource factor ratios			
Main Subsystem		*0.075618121*	*0.139973362*	
Secondary subsystem		0.13167863 0.12168000	0.07402842	

One can see that for the ratio of Effectiveness to the Control Input Value Factor of the Main Subsystem more than one ratio of the Effectiveness to the Control Input Value Factor of the Secondary Subsystem can be calculated. The above results can be shown graphically. For this purpose, Excel makes a large variety of graphs available to the user.

APPENDIX C : Listing of the computer programme

Dynamic Programming module

```
unit Danprg;

{ Program copyright (c) 1996 by Slawomir Jaroslawski }
{ Project Name: Dynamic Programming }

{ Program copyright (c) 1996 by Slawomir Jaroslawski }
{ Project Name: Dynamic Programming }

{NAMES OF THE VARIABLES USED IN THE MODULE}
{
 Number of columns - inv, i, i_f, a1, i_1, h, col_f, stcol, col, colwiel
                     lc, stg, h_4, nc, h_stage

 Number of rows    - row, a2, prop, Row_1,z,row_f, rowiel,stgh,stgl

 Values of Investments - c[h], res, r,
 Values of revenues    - G, G_1,G_2, P, y, duzy_r, duzy_c, grd2, grd1
 Minimum value - small, b
 Maximum value - big, b, lg
 Fractions of the Investment allocated to stages -x, x_2, x_1, x_p, x_p_stage
 (fractions of the Control Input Value)

 Step of input - xstep
 Difference between the allocated Investment and required resource - d
 Choice of optimisation (maximise or minimise) - opt, optm
 Number of allocated investment/ fractions of Control Input Value - i, j, ns
 Number of stages - incr, stg
 Poking variables - poke, pok, po
 }


interface

uses
{NAMES OF THE DRIVERS USED}
  SysUtils, WinTypes, WinProcs,
  Messages, Classes, Graphics,
  Controls, Forms, Dialogs,
  StdCtrls, DdeMan, Grids, DB, DBGrids, WinCrt, DBTables, Spin, ExtCtrls;
```

```
type
  {NAMES OF THE VISUAL OBJECTS USED}
  TForm1 = class(TForm)
    DdeClientConv1: TDdeClientConv;
    DdeClientItem1: TDdeClientItem;
    DdeClientItem2: TDdeClientItem;
    DdeClientItem3: TDdeClientItem;
    OpenLink: TButton;


    Button1: TButton;
    StringGrid2: TStringGrid;
    Edit1: TEdit;
    StringGrid3: TStringGrid;
    Button2: TButton;
    StringGrid1: TStringGrid;
    Edit2: TEdit;
    Edit3: TEdit;
    Button3: TButton;
    Button4: TButton;

    Memo1: TMemo;
    Button5: TButton;
    StringGrid4: TStringGrid;
    Label1: TLabel;
    Label2: TLabel;
    Edit4: TEdit;
    DdeClientItem60: TDdeClientItem;
    Edit5: TEdit;
    DdeClientItem4: TDdeClientItem;
    DdeClientConv2: TDdeClientConv;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Label10: TLabel;
    Panel4: TPanel;
    GroupBox1: TGroupBox;
```

```
    RadioButton2: TRadioButton;
    RadioButton1: TRadioButton;
    Label11: TLabel;
    Label12: TLabel;
    DdeClientItem5: TDdeClientItem;
    DdeClientItem6: TDdeClientItem;
    Edit6: TEdit;
    DdeClientItem7: TDdeClientItem;
    DdeClientItem8: TDdeClientItem;
    DdeClientItem9: TDdeClientItem;
    DdeClientItem10: TDdeClientItem;
    Button6: TButton;
    Button7: TButton;
    DdeClientItem11: TDdeClientItem;
    Label13: TLabel;

    procedure OpenLinkClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
   G : TEDIT;
   i,i_f: Integer;
   Row,Optm : Integer;
                          {10 }
      G_1,G_2 : array [0..10,0..50] of Real;
      x : integer;
      a2,a1,col_f,row_f: integer;
      i_1,Row_1,res: Real;
      xstep:extended;
```

```
    x_2: array [0..50] of Real;
    x_p,x_p_stage: array [0..1,0..50] of Real;
    h_stage: array [0..50] of integer;

implementation

{$R *.DFM}


 procedure TForm1.FormCreate(Sender: TObject);
  var
    i_1,Row_1,res: Real;
    i_2,Row_2,poke: String;
    i_f: Integer;
    DDECli : TDDEClientConv;
    Sptr: array[0..200] of Char;

Begin
{This procedure connects Fgr.xls and one of the considered
 subsystem's files (i.e. Man.xls or Second1.xls) with Dynamic Programming
 module}

 DDEClientConv2.SetLink('Excel','C:\sysopt\excelfil\FGR.xls');
 DDEClientItem60.DDEItem := 'R4C2';
 Edit5.text := DDEClientItem60.text;

 Form1.Width := 648;
 Form1.Height := 484;
 Form1.Left := 0;
 Form1.Top := 0;

   If Edit5.text = 'Main  ' then
     begin
     DDEClientConv1.SetLink('Excel', 'C:\sysopt\excelfil\main.xls');
     end
   else if Edit5.text = 'Second_1  ' then
       begin
       DDEClientConv1.SetLink('Excel', 'C:\sysopt\excelfil\second_1.xls');
       end;

 { 'Number of columns' for the 'Table of Values'  imported from Excel file}
 DDEClientItem2.DDEItem := 'R3C3' ;
 { 'Number of rows' for the 'Table of Values' imported from Excel file}
 DDEClientItem3.DDEItem  := 'R4C3' ;
 { 'Control Input Value' imported from Excel file}
```

```
 DDEClientItem4.DDEItem  := 'R3C6' ;
 Edit3.Text := DDEClientItem4.Text;
 {'Step of Input' imported from Excel file }
 DDEClientItem5.DDEItem  := 'R4C6' ;
 Edit4.Text := DDEClientItem5.text;
 {'Initial value' (starting investment value) imported from Excel file}
 DDEClientItem6.DDEItem  := 'R4C8' ;
 Edit6.Text := DDEClientItem6.text;

 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
 {The logic of the buttons' appearance/disappearance}
 OpenLink.Visible := false;
 Button1.visible := false;
 Button2.visible := false;
 Button3.visible := True;
 Button4.visible := false;
 Button5.visible := false;
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

End;


procedure TForm1.Button3Click(Sender: TObject);
 {This procedure pokes the value of a Control Input Value
 From and To Excel file }

 var
   i_1,Row_1,res: Real;
   i_2,Row_2,poke: String;
   i_f : Integer;
   DDECli : TDDEClientConv;
   Sptr: array[0..200] of Char;

 Begin
    DDECli := DDEClientItem1.DDEConv;

    begin

 { 'Step of Input' value }
       xstep := StrToFloat(edit4.text);

 { 'Control Input Value'}
        if DDECli  <> Nil then
          begin
          res := StrToFloat(Edit3.text);
```

```
          poke := FloatToStr(res);
          DDECli.PokeData(DDEClientItem4.DDEItem, StrPCopy(sptr,poke));
          end
        else
          begin
          Edit3.Text := DDEClientItem4.text;
          end;

    end;

  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
  {The logic of the buttons' appearance/disappearance}
  OpenLink.Visible := true;
  Button1.visible := false;
  Button2.visible := false;
  Button3.visible := false;
  Button4.visible := false;
  Button5.visible := false;
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{End of procedure}
End;



procedure TForm1.OpenLinkClick(Sender: TObject);
{This procedure opens link between 'Table of Main Subsystem's Performances' from
Main.xls(or 'Table of Secondary Subsystem's Performances' from Second1.xls)
and 'The Table of Values' of the Dynamic Progamming module}

type
string10 = string[10];
string_s = ^string10;
list = array[1..7] of string_s;

var
 field : list;
 i_1: Real;
 Row_1: Real;
 i, i_f,inv,prop,rev : integer;
 Row: integer;


Begin
   if (optm =1) or (optm = 2) then
      begin
```

```
    Edit1.Text := DDEClientItem2.Text;
    Edit2.Text := DDEClientItem3.Text ;
    i_1 := StrToFloat(Edit1.Text);
    Row_1 := StrToFloat(Edit2.Text);
    a1 := Round(i_1);
    a2 := Round(Row_1);
    inv := -1;
    rev := -1;
    prop := 0;

    {heading for the 'The Table of Values'}
    for i := 1 to a1 do
      begin
      inv := inv + 2;
      rev := rev + 2;
      prop := prop + 1;
      StringGrid1.Cells[inv,0] := 'Inv' + IntToStr(i);
      StringGrid1.Cells[inv+1,0] := 'Rev' + IntToStr(i);
      StringGrid1.Cells[0,prop] := 'Prop' + IntToStr(i);

  {procedure which enables  importing of values of 'Revenues' and 'Investments'
  from Excel files to Dynamic Programming module}
            for Row := 1 to a2 do
              begin
              New( Field[row]);
              Field[row]^ := 'R'+IntToStr(10+Row)+'C'+IntToStr(1+i);
              DDEClientItem7.DDEConv := DDEClientConv1;
              DDEClientItem7.DDEItem := Field[row]^;
              G_1[i,Row] := StrToFloat(DDEClientItem7.Text);
              StringGrid1.Cells[i,Row] := FloatToStr(G_1[i,Row]);
              end;
      end;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{The logic of the buttons' appearance/disappearance}
 OpenLink.Visible := false;
 Button1.visible := true;
 Button2.visible := false;
 Button3.visible := false;
 Button4.visible := false;
 Button5.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
   end

  else
```

```
        begin

{The logic of the buttons' appeaaence/disappearance}
        Label11.visible := true;
        Label12.visible := true;
        Button1.visible := False;
        Button3.visible := True;
        Label11.Caption := 'ENTER YOUR CHOISE';
        Label12.Caption := 'Maximise/Minimise';

        end;
{End of the procedure}
End;


procedure TForm1.Button1Click(Sender: TObject);
{Initiate Dynamic Programming calculations for Stage 1}
{Calculates optimum values of Revenues for the First Stage table in the
Dynamic Programming module}

var
  h ,row : integer;
  big,small : real;
  x :array [0..50] of Real;
  r : array [0..20,0..50] of Real;
  c : array [0..50] of Real;
  b : string;

Begin
  h := 0;
  row := 0;
  G_2[0,0] := 0;
  x[0] := 0;
  big := -1;
  small := 10000000;
  res := StrToFloat(Edit3.Text);
  Row_1 := StrToFloat(Edit2.Text);

 {Number of Control Input Value fractions allocated for the Stage 1}
 a1 := Round(1+(res-StrToFloat(Edit6.Text))/xstep);
 {Number of proposals related to proposals of the 'Table of Values'}
 a2 := Round(Row_1);

 {counts number of rows in the table}
   for row := 1 to (a1) do
```

```
  begin
{counts number of rows in the table}
      for h := 1 to a2 do
        begin
   {Values of Investments related to 'Revenues' of the Stage 1}
         c[h] := G_1[1,h];
   {Fractions of Control Input Value allocated to Stage 1}
         x[row] := x[row-1] + xstep;
   {Initial fraction of Control Input Value - 'Initial investment' value}
         x[1] := StrToFloat(Edit6.Text);

   {headings of the 'First stage' table}
           StringGrid2.Cells[1,0]:= 'Allocated Input';
           StringGrid2.Cells[0,row]:= 'Stage1';
           StringGrid2.Cells[h+1,0]:= 'Prop= ' + FloatToStr(h);
           StringGrid2.Cells[a2+3,0]:= 'Optimum';
           StringGrid2.Cells[1,row]:= FloatToStr(x[row ]);

   {Interpretation of non feasible regions}

   {indicates the non feasible values- '0' for maximisation and '100000000'
   for minimisation}
      IF Optm = 1 then
        b:= '0';
        IF Optm = 2 then
          b:= '100000000';
      {if value of 'Investment' is less than fraction of 'Control Input Value'
      allocated then}
          if  c[h] <= x[row] then
            begin
       {find the values of respected 'Revenues'}
            StringGrid2.Cells[h+1,row]:= FloatToStr(G_1[2,h]);
            G_2[h,row] := G_1[2,h];
            end
          else
       {otherwise indicate that value is not feasible}
             begin
             StringGrid2.Cells[h+1,row]:= b;
             G_2[h,row] := StrToFloat(b);
             end;

{Stage 1 optimisation procedure - MINIMUM AND MAXIMUM CALCULATION}

 {procedure for finding minimum value}
               If G_2[h,row] <= small then
```

```
                begin
                small := G_2[h,row];
                end;
    {procedure for finding maximum value}
                If G_2[h,row] >= big then
                begin
                big := G_2[h,row];
                end;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{The logic of the buttons' appearance/disappearance}
OpenLink.Visible := false;
Button1.visible := false;
Button2.visible := true;
Button3.visible := false;
Button4.visible := false;
Button5.visible := false;
Label11.visible := false;
Label12.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

        end;

{Stage 1 displaying non feasible regions (MINIMUM OR MAXIMUM)}
      {minimum values}
            IF Optm = 2 then
            begin
            StringGrid2.Cells[a2+3,row]:= FloatToStr(small);
            small:=1000000;
            end;
       {maximum values}
            IF Optm = 1 then
            begin
            StringGrid2.Cells[a2+3,row]:= FloatToStr(big);
            big:=-1;
            end;

  end;
{End of the procedure}
End;


procedure TForm1.Button2Click(Sender: TObject);
{This procedure continues optimising (selects maximum/minimum) values of 'Revenues'
for
```

```
any subsequent stages and displays the optimal values in the table of
'Subsequent Stages'}
{Additive maximisation/minimisation of Dynamic Programming is used}

label stgout;

var
  h ,row,z,row_f,col_f, stcol,x : integer;
  i,j,q,inc,w,ns,flag : integer;
  y,big,lg,stg,small: Real;
  r: array [0..20,0..20] of Real;
  c,P :array [0..50] of Real;
  c1,p1 :array [0..10,0..50] of Real;
  x_2,x_1,d: array [0..50] of Real;
  b : string;

{THE PROCEDURE TO OPTIMISE SUBSEQUENT STAGES OF DYNAMIC
PROGRAMMING
(Stages starts from Stage 2 onwards)}
Begin

{THE PROCEDURE TO OPTIMISE STAGE 2}
   x_2[0] := 0;
   big := -1;
   small := 10000000;
   lg := -1;
   res := StrToFloat(Edit3.Text);
   Row_1 := StrToFloat(Edit2.Text);

   {Number of Control Input Value fractions allocated for the Stage 1}
   a1 := Round(1+(res-StrToFloat(Edit6.Text))/xstep);
   {Number of proposals related to proposals of the 'Table of Values'}
   a2 := Round(Row_1);

    {counts number of rows in the table}
   for row := 1 to a1 do
    begin
       {counts number of rows in the table}
      for h := 1 to a2 do
       begin

        {Importing proposed values of 'Revenues'(from the 'Table of values')
         relevant to Stage 1}
        P[h] := StrToFloat(StringGrid1.Cells[4,h]);
        {Importing proposed values of 'Revenues'(from the 'Table of values')
```

```
 relevant to Stage 1}
c[h] := StrToFloat(StringGrid1.Cells[3,h]);
{Fractions of Control Input Value allocated to Stage 1}
x_2[row] := x_2[row-1] + xstep;
{Initial fraction of Control Input Value - 'Initial investment' value}
x_2[1] := StrToFloat(Edit6.Text);

{headings of the 'Subsequent Stages' table}
StringGrid3.Cells[0,row]:= 'Stage2';
StringGrid3.Cells[1,0]:= 'Allocated Input';
StringGrid3.Cells[h+1,0]:= 'Prop= ' + FloatToStr(h);
StringGrid3.Cells[a2+3,0]:= 'Optimum';
StringGrid3.Cells[1,row]:= FloatToStr(x_2[row]);

{calculates difference between allocated fraction
of Control Input Value and 'Investment' required for
a certain 'Revenue' }
d[row] := x_2[row]-c[h];

{if the above difference is greater or equal to the 'Initial
 Investment value' then optimise 'Revenue' values}
if (d[row] >= StrToFloat(Edit6.Text)) then
  begin
   flag := 0;
   for z := 1 to (a1) do
      begin
      x_1[z]:= StrToFloat(StringGrid2.Cells[1,z]);

      {if the above difference is equal to any fraction
      of the Control Input values allocated to the Stage 1
      then use the proposed 'Revenues' for further optimisation}
      if d[row] = x_1[z] then
      begin
      flag := 1;
        y:= StrToFloat(StringGrid2.Cells[a2+3,z]);
        StringGrid3.Cells[h+1,row]:= FloatToStr(y+P[h]);
      end
      {otherwise the optimised values of 'Revenues'are
      not feasible }
      else
       if flag = 0 then
          begin
          {interpretations of non feasible values
              for maximisation process}
           IF Optm = 1 then
```

```
                    begin                                   {&&&&&&&&&&&&&7}
                     b:= '0';
                      StringGrid3.Cells[h+1,row]:= '0';
                     end;
                     {interpretations of non feasible values
                     for minimisation process}
                          IF Optm = 2 then
                             begin
                               b:= '100000000';
                                StringGrid3.Cells[h+1,row]:= '100000000';
                             end;
                end;



          end;
  end;
{if the above difference is not equal to any fraction
of the Control Input values allocated to the Stage 1 then the optimised
'Revenues' of the first stage are not feasible for further optimisation}
        If d[row] < StrToFloat(Edit6.Text) then
          begin
{interpretations of non feasible values for minimisation process}
            IF Optm = 1 then
              begin
              b:= '0';
              StringGrid3.Cells[h+1,row]:= '0';
              end;
{interpretations of non feasible values for maximisation process}
            IF Optm = 2 then
              begin
              b:= '100000000';
              StringGrid3.Cells[h+1,row]:= '100000000';
              end;
          end;

r[h,row]:= StrToFloat(StringGrid3.Cells[h+1,row]);

{optimisation procedure - minimisation or maximisation}

{ procedure for finding maximum values}
If r[h,row] > big then
  begin
    big := r[h,row];
  end;
```

```
  { procedure for finding minimum values}
  If r[h,row] < small then
   begin
    small := r[h,row];
   end;


end;
{displaying  non feasible regions}
{for maximisation}
   IF Optm = 1 then
    begin
    StringGrid3.Cells[a2+3,row]:= FloatToStr(big);
    big:=-1;
    end;
{for minimisation}
   IF Optm = 2 then
    begin
    StringGrid3.Cells[a2+3,row]:= FloatToStr(small);
    small:=1000000;
    end;
{End of stage 2}
end;


{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
{{{{}
{ THE PROCEDURE TO OPTIMISE SUBSEQUENT STAGES (FROM STAGE 3
ONWARDS)}

x_2[a1+1] := 0{-xstep};

stcol := 4;
x := 1;
i := 0;
j := a1;     {}
stg := StrToFloat(Edit1.Text);

{number of steps/fractions of Control Input value allocated for
Stage 3 onwards}
ns  := Round(1+int((StrToFloat(Edit3.text)-
StrToFloat(Edit6.Text))/StrToFloat(Edit4.Text)));
{evaluating the number of the final row in the table of 'Subsequent stages'}
row_f := ((Round(stg/2)-1)*(ns));
{evaluating the number of columns in the table of 'Subsequent stages'}
```

```
col_f := Round(stg);

{counts number of rows in the table}
      for row := (a1+1) to (row_f) do
        begin
{headinds for the table}
          if row > x*ns then
            begin
             x := x +1;
             stcol := stcol +2;
             end;
             StringGrid3.Cells[0,row]:= 'Stage'+ IntToStr(x+1);

{counts number of columns in the table}
          for h := 1 to a2 do
             begin
        {Fractions of Control Input Value allocated to Stage 3 onwards}
              x_2[row] := x_2[row-1] + xstep;
        {Initial fraction of Control Input Value - 'Initial investment' value}
              x_2[a1+1] := StrToFloat(Edit6.Text);

              if x_2[row] > res then
               begin
               x_2[row] := StrToFloat(Edit6.text);
               end;
       {Importing proposed values of 'Revenues'(from the 'Table of values')
        relevant to Stage 3 or any of the subsequent stages}
       P[h] := StrToFloat(StringGrid1.Cells[stcol,h]);
       {Importing proposed values of 'Investments'(from the 'Table of values')
        relevant to Stage 3 or any of the subsequent stages}
       c[h] := StrToFloat(StringGrid1.Cells[stcol-1,h]);
       {allocated fractions of control Input Value displayed in the table of
       'Subsequent Stages'}
       StringGrid3.Cells[1,row]:= FloatToStr(x_2[row]);

         {calculates difference between allocated fraction
         of Control Input Value and 'Investment' required for
         a certain 'Revenue' }
         d[row] := x_2[row]-c[h];
         {if the above difference is greater or equal to the 'Initial
          Investment value' then optimise 'Revenue' values}
         if (d[row] >= StrToFloat(Edit6.Text)) then
           begin
              flag := 0;
              for z := i+1 to j do
```

```
                begin
                x_2[z]:= StrToFloat(StringGrid3.Cells[1,z]);
                {if the above difference is equal to any fraction
                 of the Control Input values allocated to the Stage 1
                 then use the proposed 'Revenues' for further optimisation}
                if d[row] = x_2[z] then
                  begin
                  flag := 1;
                  y := StrToFloat(StringGrid3.Cells[a2+3,z]);
                  StringGrid3.Cells[h+1,row]:= FloatToStr(y+P[h]);
                  end
                else
                {otherwise the optimised values of 'Revenues'are
                 not feasible }
                   if flag = 0 then
                   begin
                {interpretations of non feasible values
                      for maximisation process}
                  IF Optm = 1 then
                  begin                                  {&&&&&&&&&&&&&7}
                  b:= '0';
                  StringGrid3.Cells[h+1,row]:= '0';
                  end;
                  {interpretations of non feasible values
                     for minimisation process}
                    IF Optm = 2 then
                    begin
                    b:= '100000000';
                    StringGrid3.Cells[h+1,row]:= '100000000';
                    end;
              end;


        end;
  end;

{if the above difference is not equal to any fraction
of the Control Input values allocated to the Stage 1 then the optimised
'Revenues' of the first stage are not feasible for further optimisation}
        If d[row] < StrToFloat(Edit6.Text) then
          begin
{interpretations of non feasible values for minimisation process}
           IF Optm = 1 then
```

```
                begin
                b:= '0';
                StringGrid3.Cells[h+1,row]:= '0';
                end;
 {interpretations of non feasible values for maximisation process}
               IF Optm = 2 then
                begin
                b:= '100000000';
                StringGrid3.Cells[h+1,row]:= '100000000';
                end;
              end;

 r[h,row]:= StrToFloat(StringGrid3.Cells[h+1,row]);


 {optimisation procedure - minimisation or maximisation}

  { procedure for finding maximum values}
          If r[h,row] > lg then
          begin
          lg := r[h,row];
          end;
  { procedure for finding minimum values}
          If r[h,row] < small then
          begin
          small := r[h,row];
          end;

end;
 {displaying non feasible regions}
 {for maximisation}
 IF Optm = 1 then
   begin
 StringGrid3.Cells[a2+3,row]:= FloatToStr(lg);
 lg := -1;
    end;
 {for minimisation}
 IF Optm = 2 then
    begin
 StringGrid3.Cells[a2+3,row]:= FloatToStr(small);
 small := 10000000;
    end;

{procedure for evaluating the size of the 'Subsequent Stages' table}
for w := 1 to 10000 do
```

```
   begin
      if (row/(a1)) = w then
        begin
           i := i+(a1);
           j := j+(a1);
           goto stgout;
        end;
   end;
stgout:
end;


{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
 {The logic of the buttons' appearance/disappearance}
 OpenLink.Visible := false;
 Button1.visible := false;
 Button2.visible := false;
 Button3.visible := false;
 Button4.visible := true;
 Button5.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{End of procedure}
End;

procedure TForm1.Button4Click(Sender: TObject);
{THIS PROCEDURE INITIATES CALCULATIONS FOR THE 'CRITICAL PATH'
EVALUATION}
 {THE EVALUATION OF THE 'CRITICAL PATH' FOR THE LAST STAGE}

 label st3,stout,down;
var
  duzy_r,duzy_c: array [0..10,0..50] of Real;
  wiel,stg : Real;
  rowwiel,h,col,incr,w,v,i,j,row_f,col_f,ns,lc: Integer;
  harr: array [0..4] of Integer;
  colwiel: array[0..50]of Integer;

Begin

stg := StrToFloat(Edit1.Text);
{Number of steps/fractions of Control Input Value counting from stage 2}
ns  := Round(1+int((StrToFloat(Edit3.text)-
StrToFloat(Edit6.Text))/StrToFloat(Edit4.Text)));
{number of the final row in the table of 'Subsequent Stages'}
row_f := ((Round(stg/2)-1)*(ns));
```

```
{number of columns in the table of 'Subsequent Stages'}
col_f := Round(stg);
{number of the last column in the 'Table of Values'}
lc := Round(StrToFloat(Edit1.Text));

x_2[0] := 0{ -xstep};
wiel := 0 ;
{finds value of Revenue in the last row in the table of 'Subsequent Stages'}
duzy_r[(a2+3),row_f] := StrToFloat(StringGrid3.Cells[(a2+3),row_f]);

{counts number of Control Input value fractions allocated to the last stage}
  for row := 1 to row_f do
    begin
   {Evaluates fractions of Control Input Value allocated to the last stage
   based on the sum of 'Step of Input'}
    x_2[row] := x_2[row-1] + xstep;
   {Initial fraction of Control Input Value - 'Initial investment' value}
    x_2[1] := StrToFloat(Edit6.Text);

       {if the sum of all 'Step of Inputs' is greater than
       the Control Input Value then start calculations from
       the 'Initial Investment Value' }
       if x_2[row] > StrToFloat(Edit3.Text) then
         begin
            x_2[row] := StrToFloat(Edit6.Text);
         end;
    end;

incr := 0;

{Starting the optimisation from the last stage}
  begin
Memo1.Clear;
Memo1.lines.add('Optimised Stage'+FloatToStr(stg/2));

{looking for optimal values in the last stage, stringrid3}
      For h := 1 to a2 do     {}
        begin
   {check position of the last row - max values}
        duzy_c[h,row_f] := StrToFloat(StringGrid3.Cells[h+1,row_f]);
   {if the 'Revenue' values of the last row are equal to the most optimal
   value of that row then find the 'Revenues'}
          if (duzy_c[h,row_f] = duzy_r[a2+3,row_f])  then
            begin
            incr := incr + 1;
```

```
            h_stage[h] := h;
        {
        calculates 'Investment' left after choosing the most optimal 'Revenues'
         of the last stage - the difference between the total nvestment of
         the last row's 'Revenue'(equivalent to the total value of Control
         Input Value)and the value of proposed 'Investment' related to the chosen
         'Revenue'}
{}        x_p[1,h] := x_2[row_f]- StrToFloat(StringGrid1.Cells[(lc-1),h]);

         {if the above difference is equal to the accumulated 'Investments'
         (resource requirements) of the next stage then continue optimisation}
                for row := 1 to ns do
                  begin
                  x_2[row] := x_2[row-1] + xstep;
                  x_2[1] := StrToFloat(Edit6.Text);
                     if x_2[row] > StrToFloat(Edit3.Text) then
                     begin
                     x_2[row] := StrToFloat(Edit6.Text);
                     end;
                     if (x_p[1,h] = x_2[row]) then
                     begin
                     Memo1.lines.add('Prop'+IntToStr(h)+'|'+'Res left'+
                     FloatToStr(x_p[1,h]));
                     end;
                  end;
              end
      {if the 'Revenue' values of the last row are not equal to the most optimal
     value of that row then no feasible regions are found - stop the optimisation}

             Else if duzy_c[h,row_f]<>duzy_r[a2+3,row_f] then
               begin
               x_p[1,h] := -999;
               h_stage[h] := -999;
               end;
            x_p_stage[1,h] := x_p[1,h];


         end;
    end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
 {The logic of the buttons' appearance/disappearance}
 OpenLink.Visible := false;
 Button1.visible := false;
 Button2.visible := false;
 Button3.visible := false;
```

```
  Button4.visible := true;
  Button5.visible := true;
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{End of procedure}
End;


procedure TForm1.Button5Click(Sender: TObject);
{THE EVALUATION OF THE 'CRITICAL PATH' CONTINUES FOR THE
SUBSEQUENT STAGE}
 {FOR THE STAGE JUST AFTER THE LAST ONE TO THE FIRST STAGE}


label
st3,st4,st5,st6,st7;

type
string10 = string[10];
string_s = ^string10;
list = array[1..5] of string_s;

var
sendvc,sendvr,sendp: list;
h,h_4,row,row_4,slef,slef1,stgl,stgh,pl1,inv,stage,lc,k,r,clm,op,nc,pth,ns : integer;
rl,rm: integer;
dum, col, stg: Real;
grd1,grd2: single;

x_1: array [0..50] of Real;
sle,sle1,option: string;
ClickedOk : Boolean;

res,re: Real;
poke,pok,po: String;
DDECli : TDDEClientConv;
Sptr: array[0..200] of Char;

Begin
 {initialisation of variables continue}
pth := 0;
r := 1;
row_4 := 1;
k := 0;
ns  := Round(1+int((StrToFloat(Edit3.text)-
StrToFloat(Edit6.Text))/StrToFloat(Edit4.Text)));
```

```
st5:

{lc is a number of columns}
lc := Round(StrToFloat(Edit1.Text));
stg := StrToFloat(Edit1.Text);
{stgh is a number indicating higher row (i.e x_2[row]) in stage }
stgh := ((Round(stg/2)-1)*(a1));
{stgl is a numer indicating lower row (i.e x_2[row]) in stage}
stgl := stgh-a1;
pl1 := 0;
h_4 := 1;
inv := -1;
x_1[0] := 0{-xstep};

st3:

{loops stgl and stgh where a1 is a total number of rows}
stgl := stgl - (a1);
stgh := stgh - (a1);
pl1 := pl1 - 2;
{gets total number of columns}
col := (StrToFloat(edit1.text));
{calculates number of stages}
stage := round(col/2)+1;
{st6 marks a loop when chosen optimum path number = to previous
stage number}
st6:

{This initialises input box}
ClickedOk := InputQuery('Input Box','Prompt', sle);
{When OK on input box is pressed}
    if ClickedOk then
      begin
 {Convert string in input box to integer}
       slef := StrToInt(sle);
 {loop all possibilities of columns (i.e. 1 to h) to check if chosen path number
= to previous number}
          for h := 1 to a2 do
            begin

  {check if chosen optimum path number = to previous number after
  all calls of stage 3}
               if (slef <> h_stage[h])then
                 if (h = a2)then
```

```
                goto st6;
  {check if chosen optimum path number = to previous number at any call}
                if slef = h_stage[h] then
                 begin
  {integers from 0 to n to number headings of the 'Critical Path' table}
                  inv := inv + 1;

  {headings of the 'Critical Path' table from the last stage to stage 2}

  StringGrid4.Cells[(h_4+inv),row_4] := StringGrid1.Cells[(lc-1)+pl1+2,slef];
  StringGrid4.Cells[(h_4+inv+1),row_4] := StringGrid1.Cells[lc+pl1+2,slef];
  StringGrid4.Cells[(h_4+inv),0] := 'St'+IntToStr(stage-h_4)+'/Inv';
  StringGrid4.Cells[(h_4+inv+1),0] := 'St'+IntToStr(stage-h_4)+'/Rev';

  {shows consecutive proposals from last stage till stage 2}
  StringGrid4.Cells[(h_4+lc),0] := 'St'+IntToStr(stage-h_4)+'/Opt';
  StringGrid4.Cells[(h_4 +lc),row_4] := StringGrid1.Cells[0,slef];

  {follow all procedures with correctly chosen column}
                  break;
                  {end if slef}
                  end;
            {end for h=1 to a2}
            end;

{integers from 1 to n to count number of optimised 'Revenues' and related 'Investments'}
       k := k + 1;
{total number of columns of the 'Critical Path' table}
       clm := round(StrToFloat(Edit1.text));

{procedure to send the optimised values of 'Revenues' and respective 'Investments
to Excel file}
        DDECli := DDEClientItem1.DDEConv;
        begin
           if DDECli  <> Nil then
             begin
             New( sendp[k]);
             New( sendvr[k]);
             New( sendvc[k]);

 {Names of proposals}
 sendp[k]^ := 'R'+IntToStr(18+r)+'C'+IntToStr((13+stage)-k);
 {Revenues}
 sendvr[k]^ := 'R'+IntToStr(18+r)+'C'+IntToStr((lc+3)-(2+inv+k));
 {Investments}
```

```
  sendvc[k]^ := 'R'+IntToStr(18+r)+'C'+IntToStr((lc+3)-(1+inv+k));
  TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(8))).DDEItem :=
sendp[k]^;
  TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(9))).DDEItem :=
sendvr[k]^;
  TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(10))).DDEItem :=
sendvc[k]^;

  {poking value of Critical Path to Excel}
  res := StrToFloat(StringGrid4.Cells[inv+k,r]);
  poke := FloatToStr(res);
  re := StrToFloat(StringGrid4.Cells[(inv+k+1),r]);
  pok := FloatToStr(re);

  {shows proposals for critical path}
  po :=  StringGrid4.Cells[(h_4 +lc),row_4];
  DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(8))).DDEItem, StrPCopy(sptr,po));
  DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(9))).DDEItem, StrPCopy(sptr,poke));
  DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(10))).DDEItem, StrPCopy(sptr,pok));

 {shows proposal for maximum 5 stages}
 {Dispose( sendp[k]);
 Dispose( sendvr[k]);
 Dispose( sendvc[k]);    }

              {end begin DDECLIENT}
              end;
         {end begin before if DDECLIENT}
          end;

      {end if Click not OK}
      end
     {if Click not OK}
     else
        begin

        goto st4;
        {end of else and click OK}
        end;

{sorts e.g. 6 values('Investments') from given set e.g. 1-6 6-12 and
assigns x_2[row] again}
```

```
      for row := 1 to stgh do
        begin
        x_2[row] := x_2[row-1] + xstep;
        x_2[1] := StrToFloat(Edit6.Text);
            if x_2[row] > StrToFloat(Edit3.text) then
              begin
              {start from the initial Investment value again}
              x_2[row] :=StrToFloat(Edit6.Text);
              end;
        {end for row=1 to stgh}
        end;

{Looking for optimal values in stage1, stringrid2}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
{{{{}

{loop all possibilities of columns (i.e. 1 to h) to check if chosen path
number = to previous number}
    for row := 1 to (a1) do

      begin
      {calculate accumulated Control Input Value fractions alocated for stage 1}
      x_1[row] := x_1[row-1] +xstep;
      {Initial value of the accumulated fractions - 'Step of Input' value}
      x_1[1] := StrToFloat(Edit6.Text);
      end;

    {if in range of stage1 then}
    if stgh < (a1) then
     begin
     {selecting all optimal values of 'Revenues' and respective 'Investments' in Stage 1 }
        for row := 1 to (a1) do
          if (x_p[1,slef] = x_1[row]) then
           begin
              for h := 1 to a2 do
                begin
                   if StrToFloat(StringGrid2.Cells[h+1,row]) =
StrToFloat(StringGrid2.Cells[a2+3,row]) then
                     begin

 {displaying the optimised values of 'Revenues and 'Investments in the
'Critical Path' table}
 StringGrid4.Cells[(h_4+inv)+2,0] := 'St1/Inv1';
 StringGrid4.Cells[(h_4+inv+1)+2,0] := 'St1/Rev1';
 StringGrid4.Cells[(h_4+inv)+2,row_4] := StringGrid1.Cells[(lc-1)+pl1,h];
```

```
  StringGrid4.Cells[(h_4+inv+1)+2,row_4] := StringGrid1.Cells[lc+pl1,h];
  {shows last proposal }
  StringGrid4.Cells[(h_4+lc)+1,0] := 'St1'+'/Opt';
  StringGrid4.Cells[(h_4 +lc)+1,row_4] := StringGrid1.Cells[0,h];

  {integers from 0 to n to number headings of the 'Critical Path' table}
                    inv := inv + 1;
                    {end if StrToFloat}
                    end;
                    {end for h}
                end;
          {end if (x_p[1,slef] = x_1[row])}
           end;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{integers from 1 to n to count number of optimised 'Revenues' and related
'Investments'}
     k := k + 1;
     DDECli := DDEClientItem1.DDEConv;
      begin
         if DDECli  <> Nil then
           begin
           New( sendp[k]);
           New( sendvr[k]);
           New( sendvc[k]);
           {Names of proposals}
           sendp[k]^ := 'R'+IntToStr(18+r)+'C'+IntToStr((13+stage)-k);
           {Revenues}
           sendvr[k]^ := 'R'+IntToStr(18+r)+'C'+(IntToStr((lc+3)-(2+inv+k)));
           {Investments}
           sendvc[k]^ := 'R'+IntToStr(18+r)+'C'+IntToStr((lc+3)-(1+inv+k));
           {Sending values of 'Revenues' and 'Investments' to Excel files}
   TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(8))).DDEItem :=
sendp[k]^;
   TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(9))).DDEItem :=
sendvr[k]^;
   TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(10))).DDEItem :=
sendvc[k]^;

{Poking value of Critical Path to Excel}
   res := StrToFloat(StringGrid4.Cells[inv+k,r]);
   poke := FloatToStr(res);
   re := StrToFloat(StringGrid4.Cells[(inv+k+1),r]);
   pok := FloatToStr(re);
```

```
   po :=  StringGrid4.Cells[(h_4 +lc)+1,row_4];

   DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(9))).DDEItem, StrPCopy(sptr,poke));
   DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(10))).DDEItem, StrPCopy(sptr,pok));
   DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
IntToStr(8))).DDEItem, StrPCopy(sptr,po));

{Dispose( sendp[k]);
 Dispose( sendvr[k]);
 Dispose( sendvc[k]);   }

         {end if DDECLIENT}
          end
     {end begin before DDECLIENT}
      end;

 {if all stages are optimised proceed to choice "to click or not to click"}
     ClickedOk := InputQuery('Input Box','Do You Want to Continue?', option);
     if ClickedOk then
       begin
          if option = 'y' then
           begin
 { initiation to count number of optimised 'Revenues' and related 'Investments'}}
           k := 0;
 {integers from 1 to n}
           r := r +1;
{Number of selected optimal 'Critical Paths'}
           pth := pth + 1;
 {if number of proposals is greater than 5 stop optimisation and go to st4}
              If pth = 5 then
                begin
                Memo1.Lines.Add('Maximum number of paths used - 5' );
                goto st4;
                end;

                Memo1.Clear;

{Loking for optimal values of 'Revenues' in the stages between, after the last
and before the first stage}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
{{{}
                Memo1.lines.add('Optimised Stage'+IntToStr(Stage-1));
              for h := 1 to a2 do
```

```
                begin
 {let the resource (Revenue) value left from the previous stage be
 the resource level used in the current stage}
                x_p[1,h] := x_p_stage[1,h];
                  if x_p[1,h] <> -999 then
                   begin
 {h_stage must be initialised to be properly called by ClickOk}
                   h_stage[h] :=h;

{loop used to find the values of 'Revenues' which resource requirement
('Investment') is equal to the allocated 'Investment'}

                      for rl := 1 to ns do
                        begin
                        x_2[rl] := x_2[rl-1] + xstep;
                        x_2[1] := StrToFloat(Edit6.Text);
                          if x_2[rl] > StrToFloat(Edit3.Text) then
                           begin
                           x_2[rl] := StrToFloat(Edit6.Text);
                           end;
                          if (x_p[1,h] = x_2[rl]) then
                           begin
                           Memo1.lines.add('Prop'+IntToStr(h)+'|'
                           +'Res left'+
                           FloatToStr(x_p[1,h]));
                           end
                        {end for row 1 to ns}
                        end ;
                     {end if x_p <> dummy}
                     end;
                  {end for h=1toa2}
                  end;


        row_4 := row_4+1;
{if the optimal values of 'Revenues in the stages have been selected then
go back to Click button with h_stage initialised to stage 1}
           goto st5;
          {end if option y}
          end
          else
          goto st4;
      {end if Click OK}
      end
    else
```

```
  Memo1.Clear;
  goto st4;
 {end if stgh<a1}
  end;

{After optimisation of all stages if the user pressed 'Y' in the 'Input Box'
the new critical path is optimised and the whole procedure is started all over
again}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
{{{}

{Loop which starts the optimisation of the last stage again}
  for row := stgl+1 to stgh do
    begin
      if x_p[1,slef] = x_2[row] then
       begin
       Memo1.Clear;
       Memo1.lines.add('Optimised Stage'+IntToStr(stage-1-h_4));
          for h := 1 to a2 do
            begin
            grd1 := StrToFloat(StringGrid3.Cells[h+1,row]);
            grd2 := StrToFloat(StringGrid3.Cells[a2+3,row]);
               if grd1 = grd2  then
                 begin
                 x_p[1,h] := x_2[row]- StrToFloat(StringGrid1.Cells[(lc-1)+pl1,h]);

{initialise columns for optimum path}
                 h_stage[h] := h;

{loop used to find the values of 'Revenues' which has a resource requirement
('Investment') equal to the allocated 'Investment'}
                    for rm := 1 to ns do
                     begin
                     x_2[rm] := x_2[rm-1] + xstep;
                     x_2[1] := StrToFloat(Edit6.Text);
                        if x_2[rm] > StrToFloat(Edit3.Text) then
                          begin
                          x_2[rm] := StrToFloat(Edit6.Text);
                          end;
                        if (x_p[1,h] = x_2[rm]) then
                          begin
                          Memo1.lines.add('Prop'+IntToStr(h)+'|'
                          +'Res. left'+FloatToStr(x_p[1,h]));
                          end;
                     end;
```

```
                    {{{{{{{{{{{{{{{{{{{{{{{{{}
                  {end if grid1=grid2}
                  end
                 {dummy columns not wanted}
                 else
                    begin
                    h_stage[h] := -999;
                    end;
               {end for h=1to a2}
               end
          {end if xp[1,slef]=x2[row]}
          end;
     {end for row =stgl to stgh}
     end;

{when the whole last stage has been optimised, go to the beginning of the
button5 procedure to optimise the subsequent stages }
 h_4 := h_4 +1;
 goto st3;

 st4:
{End of the procedure}
End;


procedure TForm1.RadioButton1Click(Sender: TObject);
{ACTIVATES MAXIMISATION}
begin
Optm := 1;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
{ACTIVATES MINIMISATION}
begin
Optm := 2;
end;

procedure TForm1.Button6Click(Sender: TObject);
{THIS PROCEDURE CLEARS DATA IN THE 'CRITICAL PATH'(in Main.xls or
Second1.xls)
 TABLE AND 'TABLE OF VALUES' (in the Dynamic Programming module on Delphi)
SO NEW
OPTIMISED VALUES OF 'REVENUE' AND 'INVESTMENT' CAN BE DISPLAYED
}
var
```

```
clearr,clearc: integer;
Sptr: array[0..200] of Char;
begin


for clearc := 1 to 100 do
for clearr := 1 to 100 do
   begin
   StringGrid2.Cells[clearc,clearr]:= '';
   StringGrid3.Cells[clearc,clearr]:= '';
   StringGrid4.Cells[clearc,clearr]:= '';
   end;

{Erasing values of 'Investments' and 'Revenues'}

DDEClientItem11.DDEItem := 'R19C2:R28C18';
DDEClientConv1.PokeData(DDEClientItem11.DDEItem,StrPCopy(sptr,''));


{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{logic of the buttons appearance/disappearance}
 OpenLink.Visible := False;
 Button1.Visible := False;
 Button2.Visible := False;
 Button3.Visible := True;
 Button4.Visible := False;
 Button5.Visible := False;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;


procedure TForm1.Button7Click(Sender: TObject);
{CLOSES THE DYNAMIC PROGRAMMING MODULE}
begin
close;
end;

end.
```

APPENDIX C (continued)

Monte Carlo simulation module

```
unit Msim;

{This module was designed to perform Monte Carlo simulation for the group
of sample values which are stored on the Excel spreadsheet}

{VARIABLES USED FOR THIS MODULE}
{
i;i_n,col     - amount of columns/ amount of transitions of a state
z,z_2,row     - amount of rows/ number of samples of transition states
RN,RNV        - number of trials used for the simulation process
index         - probabilities of a sample occurrence
index_n,Num   - size of the range of sample occurrence
n_nn,nn_1     - amount of random numbers generated
count         - count the amount of frequencies of occurrence for samples
P             - probabilities of sample occurrence
t             - generates random numbers
ave           - cumulative averages
evde          - cumulative standard deviations
Table,v       - values of samples of transition rates
mid1, mid1_p  - mid points
big, small    - respectively biggest and smallest values of transition rates
nv            - cumulative step values;
step          - step value of the representative values of the range of samples




}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, DBGrids, StdCtrls, DB, DBTables, DdeMan, Spin,
  ExtCtrls, ToCtrl;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
```

```
    TRIAL: TLabel;
    states: TLabel;
    Edit3: TEdit;
    DdeClientConv1: TDdeClientConv;
    DdeClientItem1: TDdeClientItem;
    DdeClientItem2: TDdeClientItem;
    SpinEdit1: TSpinEdit;
    Edit2: TEdit;
    SAMPLES: TLabel;
    DdeClientConv2: TDdeClientConv;
    DdeClientItem3: TDdeClientItem;
    Edit4: TEdit;
    DdeClientItem4: TDdeClientItem;
    DdeClientItem5: TDdeClientItem;
    StringGrid3: TStringGrid;
    Panel1: TPanel;
    Label1: TLabel;
    Panel2: TPanel;
    Panel3: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    StringGrid4: TStringGrid;
    DdeClientItem25: TDdeClientItem;
    Label4: TLabel;
    DdeClientItem6: TDdeClientItem;
    SpinEdit2: TSpinEdit;
    Edit5: TEdit;
    DdeClientItem7: TDdeClientItem;
    Label5: TLabel;
    Button2: TButton;
    Edit6: TEdit;
    DdeClientItem8: TDdeClientItem;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Edit6Change(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
type
```

```
string11 = string[11];
string_s = ^string11;
list_s = array[1..50] of string_s;

var
  Form1: TForm1;
   row,col,path_1 :integer;
  Sending : list_s;

implementation

{$R *.DFM}

PROCEDURE TForm1.FormCreate(SENDER:TObject);

{The whole procedure links Delphi module(Msim unit) with Excel spreadsheets}
Var
sptr: array[0..200] of Char;

 BEGIN
Form1.Width := 648;
Form1.Height := 484;
Form1.Left := 0;
Form1.Top := 0;
{This code links Excel final output file (FGR.xls) with Delphi (Msim unit)}

DDEClientConv2.SetLink('Excel','C:\sysopt\excelfil\FGR.xls');
DDEClientItem3.DDEItem := 'R4C2';
Edit4.text := DDEClientItem3.text;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

DDEClientConv1.PokeData(DDEClientItem7.DDEItem,StrPCopy(sptr,'1'));
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{[}

{'If' statement connect this Delphi module with currently chosen Spreadsheet
(considered subsystem)}

If Edit4.text = 'Main  ' then
begin

DDEClientConv1.SetLink('Excel', 'C:\sysopt\excelfil\main.xls');

end
else if Edit4.text = 'Second_1  ' then
```

```
begin

 DDEClientConv1.SetLink('Excel', 'C:\sysopt\excelfil\second_1.xls');

end;

{Transfers values of numbers used as the maximum limits for the
"For" loop statements}

{maximum number of rows/transition rate samples in excel file}
DDEClientItem1.DDEItem := 'R70C2';
{maximum number of columns/transition rates of stages in excel file}
DDEClientItem2.DDEItem := 'R71C2';
{maximum number of critical paths at the considered subsystem}
DDEClientItem6.DDEItem := 'R6C6';

DDEClientItem7.DDEItem := 'R69C3';

Edit3.text := DDEClientItem1.text;
Edit2.text := DDEClientItem2.text;
Edit5.text := DDEClientItem6.text;
DDEClientItem7.DDEConv := DDEClientConv1;

Spinedit2.MinValue := 1;
SpinEdit2.MaxValue := Round(StrToFloat(Edit5.text));
If (SpinEdit2.MaxValue-Spinedit2.MinValue) = 0 then
Spinedit2.enabled := false;

path_1 := StrToInt(SpinEdit2.Text);
{Control Input Value}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{[[}
DDEClientItem8.DDEConv := DDEClientConv1;
DDEClientItem8.DDEItem := 'R'+ IntToStr(121+path_1)+'C2';

Edit6.Text := DDEClientItem8.text;
{[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[}

 {Heading of the fist table - Table of states transfer rates}

  StringGrid4.Cells[1,0]:='R12'; StringGrid4.Cells[2,0]:='R13';
  StringGrid4.Cells[3,0]:='R14'; StringGrid4.Cells[4,0]:='R15';
  StringGrid4.Cells[5,0]:='R21'; StringGrid4.Cells[6,0]:='R23';
  StringGrid4.Cells[7,0]:='R24'; StringGrid4.Cells[8,0]:='R25';
  StringGrid4.Cells[9,0]:='R31'; StringGrid4.Cells[10,0]:='R32';
  StringGrid4.Cells[11,0]:='R34'; StringGrid4.Cells[12,0]:='R35';
```

```
  StringGrid4.Cells[13,0]:='R41'; StringGrid4.Cells[14,0]:='R42';
  StringGrid4.Cells[15,0]:='R43'; StringGrid4.Cells[16,0]:='R45';
  StringGrid4.Cells[17,0]:='R51'; StringGrid4.Cells[18,0]:='R52';
  StringGrid4.Cells[19,0]:='R53'; StringGrid4.Cells[20,0]:='R54';

END;


procedure TForm1.Button1Click(Sender: TObject);
label endi;
type
string10 = string[10];
string_ptr = ^string10;
list = array[1..10,1..10] of string_ptr;


var
z,i,i_n,index,index_n,Num,Z_2,nn_n,nn_1,nn_3,Row, Col: integer;

st,RN : string;
t : double;
count,k,incr,RNV,pathn: integer;

P, ave: array [0..25, 0..10] of single;
evde : array [0..10, 0..5] of single;

v,nv,Table, mid1, mid1_p: array [0..25,0..10]of single;
big,small, Step : array [0..25] of single;


 res, re: Real;
 poke, pok: String;
 DDECli : TDDEClientConv;
 Sptr: array[0..200] of Char;

 Pick : list;

begin

Row := round(StrToFloat(Edit3.text));
Col := round(StrToFloat(Edit2.text));

 {maximum number of trials}
 RN := SPINEDIT1.TEXT;
 RNV := Round(StrToFloat(RN));
```

```
{number of stages/states}
for i := 1 to col do
 begin
 big[i] := -1;
 small[i] := 100000;

 {number of samples/values of transition rates}
 for z := 1 to row  do
    begin
    st := IntToStr(z);
    Edit1.Text := st;    {z value}


 {Picking values of transition rates from spreadsheet}
 {Transitions rates comes from the currently considered subsystem
 and are regarded as input values for the simulation}
 New(Pick[i,z]);
 Pick[i,z]^:= 'R'+IntTostr(78+z)+'C'+IntToStr(1+i);

 DDEClientItem4.DDEConv := DDEClientConv1;
 DDEClientItem4.DDeItem := Pick[i,z]^;
 StringGrid4.Cells[i,z]:= DDEClientItem4.Text;

 Dispose(Pick[i,z]);

 {calculation of maximum and minimum values of each considered transition
 for each state of a subsystem}
  v[i,z] := StrToFloat(StringGrid4.Cells[i,z]);
   if v[i,z] > big[i] then
     begin
     big[i] := v[i,z];
     end;
   if v[i,z] < small[i] then
     begin
     small[i] := v[i,z];
     end;
  end;
     StringGrid4.Cells[0,z+2]:= 'biggest value ';
     StringGrid4.Cells[0,z+3]:= 'smallest value';
     StringGrid4.Cells[i,z+3]:= FloatToStr(small[i]);
   {2}StringGrid4.Cells[i,z+2]:= FloatToStr(big[i]);
end;

{Check the size of a sample (number of samples/transition
```

```
rates for every transition }
For i := 1 to col do
  BEGIN
   nv[i,0]:= 0;
   mid1[i,0]:= 0;
   for Z_2 := 1 to row do
      begin
   {3}  Table[i,Z_2]:= StrToFloat(StringGrid4.Cells[i,Z_2]);
      end;

If z <= 20 then
  Begin
   Num := Round(0.5*z);
  End                    {STEP VALUE}
   Else
  Begin
   Num:= 20
  End;

   Step[i] := (big[i]/Num);

   {Calculation of frequencies of occurrences for
   given sample values/transition rates}
   For index := 1 to Num do
    begin
     count := 0;
     k := 0;
     P[i,0] := 0;

     {Calculate cumulative steps}
     nv[i,index] := nv[i,index-1]+ Step[i];
     {Use cumulative steps and calculates cumulative midpoints
     The midpoints are needed to calculate representative ranges of
     sample values occurrence}
     mid1[i,index] := (nv[i,index]-nv[i,index-1])/2+ nv[i,index-1]; {MID POINTS }

     for Z_2 := 1 to row do
        begin
        If (Round(100000*Table[i,Z_2])<= Round(100000*nv[i,index]))and
        (Round(100000*Table[i,Z_2])> Round(100000*nv[i,index-1])) Then
           begin
    {counts frequences of occurance of transition rates}
              count := count + 1;

           end;
```

```
        end;

      {calculates probabilities of occurrence for transition rates}
      P[i,index] := P[i,index-1] + count/z;

      end;

   End;

{Generation of random numbers  }
 randomize;

For i_n := 1 to col do
 begin
 For nn_n := 1 to RNV do
  begin

  {value assigned to evoke randomization only once}
  t:=Random;
  For index_n := 1 to Num do
   begin
   {Checks if the biggest and the smallest numbers have the same value
   If this statement is true then all probabilities of occurrence
    are the same}
     if small[i_n]= big[i_n] then
         begin
      mid1_p[i_n,nn_n] := big[i_n];

         end
   {If the above statement is false then different ranges of transition
   rates occurrence are generated}
   else If (t <= P[i_n,index_n]) and (t > P[i_n,index_n-1]) then
    begin

     mid1_p[i_n,nn_n] := mid1[i_n,index_n];

    end;

   end;


 end;

{Calculation of cumulating  averages (p754, H.A.Taha[7] ) and
their respective standard deviations}
```

```
ave[i_n,0] := 0;

For nn_1 := 1 to RNV do
 begin

   {Calculates cumulative mid points for calculations
   of average values of transition rates}
   ave[i_n,nn_1] := ave[i_n,nn_1-1] + mid1_p[i_n,nn_1];
   If ave[i_n,nn_1]<= 1e-10 then
    ave[i_n,nn_1]:= 0;

end;

 {Calculates standard deviation of the last 5 values}
 evde[i_n,RNV-6] := 0;
 For nn_3 := (RNV-5) to RNV do
  begin
   evde[i_n,nn_3] := abs(ave[i_n,nn_3]/nn_3-ave[i_n,RNV]/RNV)+evde[i_n,nn_3-1];

   If evde[i_n,nn_3]<= 1e-7 then
    evde[i_n,nn_3]:= 0;


  end ;

{Heading of the second table- Table of simulated transition rates and
their respective standard deviations}
StringGrid3.Cells[0,1]:=' Average';
StringGrid3.Cells[0,2]:=' STD DEV';
StringGrid3.Cells[1,0]:='R12'; StringGrid3.Cells[2,0]:='R13';
StringGrid3.Cells[3,0]:='R14'; StringGrid3.Cells[4,0]:='R15';
StringGrid3.Cells[5,0]:='R21'; StringGrid3.Cells[6,0]:='R23';
StringGrid3.Cells[7,0]:='R24'; StringGrid3.Cells[8,0]:='R25';
StringGrid3.Cells[9,0]:='R31'; StringGrid3.Cells[10,0]:='R32';
StringGrid3.Cells[11,0]:='R34'; StringGrid3.Cells[12,0]:='R35';
StringGrid3.Cells[13,0]:='R41'; StringGrid3.Cells[14,0]:='R42';
StringGrid3.Cells[15,0]:='R43'; StringGrid3.Cells[16,0]:='R45';
StringGrid3.Cells[17,0]:='R51'; StringGrid3.Cells[18,0]:='R52';
StringGrid3.Cells[19,0]:='R53'; StringGrid3.Cells[20,0]:='R54';

 New(Sending[i_n]);
{Output for the averages of transition rates}
  StringGrid3.Cells[i_n,1]:= FloatToStr(ave[i_n,nn_1]/nn_1);
```

```
{Output for the standard deviations}
 StringGrid3.Cells[i_n,2]:= FloatToStr(evde[i_n,nn_3]/(RNV-1));


 {Sends the output values(averages and standard deviations) to spreadsheet
 of an Excel file}

 {counting amount of server components }

pathn := StrToInt(SpinEdit2.text); {path number of considered subsystem}
Sending[i_n]^ := 'R'+IntToStr(106+pathn) + 'C'+ IntToStr(1+i_n);

{Poking value of critical Path to excel}
res := StrToFloat(StringGrid3.Cells[(i_n),1]);
poke := FloatToStr(res);

DDEClientItem5.DDEConv := DDEClientConv1;
DDEClientItem5.DDEItem := Sending[(i_n)]^;
DDEClientConv1.PokeData(DDEClientItem5.DDEItem, StrPCopy(sptr,poke));
Dispose(Sending[i_n]);

end{i_n};

end;


procedure TForm1.SpinEdit2Change(Sender: TObject);
var
 sptr: array[0..200] of Char;
 path_1 :integer;
begin

DDEClientConv1.PokeData(DDEClientItem7.DDEItem,StrPCopy(sptr,SpinEdit2.Text));

path_1 := StrToInt(SpinEdit2.Text);
{Control Input Value}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{[[}
DDEClientItem8.DDEConv := DDEClientConv1;
DDEClientItem8.DDEItem := 'R'+ IntToStr(121+path_1)+'C2';

Edit6.Text := DDEClientItem8.text;

{[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[}
```

```
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Close;
end;

procedure TForm1.Edit6Change(Sender: TObject);
var
sptr: array[0..200] of Char;
begin

DDEClientConv1.PokeData(DDEClientItem8.DDEItem,StrPCopy(sptr,Edit6.Text));
end;

end.
```

APPENDIX C (continued)

Fuzzy Logic module

```
unit Fuzf;

{ Program copyright (c) 1996 by Slawomir Jaroslawski }
{ Project Name: Fuzzy Logic }


interface

uses
  WinTypes, WinProcs,SysUtils,
  Classes, Graphics,
  Controls, Printers, Forms, StdCtrls, Grids, DdeMan, Spin, ExtCtrls;

const
  fieldNames : array[1..3,1..3] of String =

(('R186C5','R187C5','R188C5'),('R186C6','R187C6','R188C6'),('R186C7','R187C7','R188
C7'));

type
  TForm1 = class(TForm)
```

```
DdeClientConv1: TDdeClientConv;
DdeClientItem1: TDdeClientItem;
DdeClientItem2: TDdeClientItem;
DdeClientItem3: TDdeClientItem;
DdeClientItem4: TDdeClientItem;
DdeClientItem5: TDdeClientItem;
DdeClientItem6: TDdeClientItem;
DdeClientItem7: TDdeClientItem;
DdeClientItem8: TDdeClientItem;
DdeClientItem9: TDdeClientItem;
StringGrid1: TStringGrid;
Edit2: TEdit;
Edit3: TEdit;

Button1: TButton;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Label2: TLabel;
Label3: TLabel;
Label5: TLabel;
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Edit7: TEdit;
Edit8: TEdit;
Edit9: TEdit;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
DdeClientItem10: TDdeClientItem;
DdeClientItem11: TDdeClientItem;
DdeClientItem12: TDdeClientItem;
DdeClientItem13: TDdeClientItem;
DdeClientItem14: TDdeClientItem;
DdeClientItem15: TDdeClientItem;
DdeClientItem16: TDdeClientItem;
DdeClientItem17: TDdeClientItem;
```

```
DdeClientItem18: TDdeClientItem;
Button5: TButton;
Label16: TLabel;
Button6: TButton;
Memo1: TMemo;
Edit1: TEdit;
DdeClientItem19: TDdeClientItem;
DdeClientItem20: TDdeClientItem;
DdeClientItem21: TDdeClientItem;
DdeClientItem22: TDdeClientItem;
DdeClientItem23: TDdeClientItem;
DdeClientItem24: TDdeClientItem;
DdeClientItem25: TDdeClientItem;
DdeClientItem26: TDdeClientItem;
DdeClientItem27: TDdeClientItem;
ListBox1: TListBox;
ListBox2: TListBox;
Edit10: TEdit;
DdeClientItem28: TDdeClientItem;
DdeClientItem29: TDdeClientItem;
DdeClientConv2: TDdeClientConv;
SpinEdit1: TSpinEdit;
DdeClientItem30: TDdeClientItem;
DdeClientItem31: TDdeClientItem;
DdeClientItem32: TDdeClientItem;
DdeClientItem33: TDdeClientItem;
DdeClientItem34: TDdeClientItem;
DdeClientItem35: TDdeClientItem;
DdeClientItem36: TDdeClientItem;
DdeClientItem37: TDdeClientItem;
Label1: TLabel;
Label4: TLabel;
Panel1: TPanel;
Button7: TButton;
DdeClientConv3: TDdeClientConv;


procedure FormPaint(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
```

```
    procedure Button6Click(Sender: TObject);

    procedure ListBox1Click(Sender: TObject);
    procedure ListBox2Click(Sender: TObject);

    procedure SpinEdit1Change(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure Edit5Change(Sender: TObject);
    procedure Edit6Change(Sender: TObject);
    procedure Edit7Change(Sender: TObject);
    procedure Edit8Change(Sender: TObject);


  end;

var
  Form1: TForm1;
  Form2: TForm1;
  G_1: ARRAY[0..4,0..4] OF REAL;
  Col,Row,ya1,ya2,yb1,yb2,st,i:integer;
  ua1,ua2,ub1,ub2,la1,la2,la3,la4,lb1,lb2,lb3,lb4,x:real;
  res_1,res_2,res_3,res_4,res_5,res_6,res_7,res_8,res_9,spa,pka,xa,spb,pkb,xb,po:real;
  poke_1,poke_2,poke_3,poke_4,poke_5,poke_6,poke_7,poke_8,poke_9,pok: string;
  Sptr,spt: array[0..200] of Char;
  DDeCli : TDDEClientConv;

implementation

{$R *.DFM}

 procedure TForm1.FormCreate(Sender: TObject);

{a1:real;}

begin

DDEClientConv2.SetLink('Excel','C:\sysopt\excelfil\FGR.xls');
DDEClientItem35.DDEItem := 'R4C2';
Edit10.text := DDEClientItem35.text;

Form1.Width := 648;
```

```
Form1.Height := 484;
Form1.Left := 0;
Form1.Top := 0;


{{{{{{{{{{{{{{{{{{{{{Connections with Excel{{{{{{{{{{{{{{{{{{{{
If  Edit10.Text = 'Main  ' Then
begin
DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\main.xls');
end
Else if Edit10.Text = 'Second_1  ' Then
begin
DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\second_1.xls');
end;

DDECli := DDEClientItem1.DDEConv;

{option number equivalent to number of critical path}
DDEClientItem34.DDEItem := 'R196C1';
{maximum number of paths}
DDEClientItem36.DDEItem := 'R6C6';
{amount of states}
DDEClientItem37.DDEItem := 'R5C6';
st := Round(StrToFloat(DDEClientItem37.Text));

SpinEdit1.MinValue := 1;
SpinEdit1.MaxValue := Round(StrToFloat(DDEClientItem36.Text));
If (SpinEdit1.MaxValue-SpinEdit1.MinValue) = 0 then
  SpinEdit1.enabled := False;

DDECli.PokeData(DDEClientItem34.DDEItem, StrPCopy(sptr,'1'));

DDEClientItem10.DDEItem := 'R194C5'; {target A}
DDEClientItem11.DDEItem := 'R195C5'; {space A}
DDEClientItem12.DDEItem := 'R192C5';   {input A}

DDEClientItem13.DDEItem := 'R194C6';   {target B}
DDEClientItem14.DDEItem := 'R195C6';   {space B}
DDEClientItem15.DDEItem := 'R192C6';   {input B}

DDEClientItem16.DDEItem := 'R194C7';   {target output}
DDEClientItem17.DDEItem := 'R195C7';   {space output}
DDEClientItem18.DDEItem := 'none';    {output}
```

```
 Form1.Height := 488;
 Form1.Width := 648;

 DDEClientItem19.DDEItem := 'R196C5';
 DDEClientItem20.DDEItem := 'R197C5';
 DDEClientItem21.DDEItem := 'R198C5';
 DDEClientItem22.DDEItem := 'R199C5';
 DDEClientItem23.DDEItem := 'R200C5';


 DDEClientItem24.DDEItem := 'R196C6';
 DDEClientItem25.DDEItem := 'R197C6';
 DDEClientItem26.DDEItem := 'R198C6';
 DDEClientItem27.DDEItem := 'R199C6';
 DDEClientItem28.DDEItem := 'R200C6';

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{Headings of the Fuzzy Rule{{{{{{{{{{{{{{{{{{{{{{{{}
 for col := 1 to 3 do
 Begin

 StringGrid1.Cells[1,0] := 'LOW/A';
 StringGrid1.Cells[2,0] := 'MEDIUM/A' ;
 StringGrid1.Cells[3,0] := 'HIGH/A';
 StringGrid1.Cells[0,1] := 'LOW/B';
 StringGrid1.Cells[0,2] := 'MEDIUM/B' ;
 StringGrid1.Cells[0,3] := 'HIGH/B';

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{Transfer of Fuzzy Rule setup from the Excel spreadsheet{{{{{{}
 for Row := 1 to 3 do

 begin
   TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(Col))).DDEConv :=
DDEClientConv1;
   TDDEClientItem(FindComponent('DDEClientItem' + IntToStr(Col))).DDEItem :=
FieldNames[Col,Row];
   StringGrid1.Cells[Col,Row]
:=(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(Col))).Text);
     end;
 END ;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
```

```
{{{{{Logic of the buttons appearance and disappearance{{{{{{{{{{{}
Memo1.visible := false;
Button2.visible := true;
Button3.visible := false;
Button1.visible := false;
Button5.visible := false;
Button4.visible := false;
Button6.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

end;

procedure TForm1.FormPaint(Sender: TObject);
var
 R: TRect;
 i,x,y,Col,Row: integer;
 Color: LongInt;

begin
{{{{{{{{{{{{Target, space and the Input values for Input A, B and
the Output{{{{{{}

Edit1.text := DDEClientItem10.text;
Edit2.text := DDEClientItem11.text;
Edit3.text := DDEClientItem12.text;

Edit4.text := DDEClientItem13.text;
Edit5.text := DDEClientItem14.text;
Edit6.text := DDEClientItem15.text;

Edit7.text := DDEClientItem16.text;
Edit8.text := DDEClientItem17.text;
Edit9.text := DDEClientItem18.text;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{Initial graphics for three sets of triangles{{{{{{{{{{{{{}
R :=getclientRect;
x := R.right;
y := R.bottom;
canvas.pen.color := RGB(255,255,255);
canvas.moveto(x-0,y-70);
canvas.lineto(x-400,y-70);
canvas.lineto(x-400,y-150);
canvas.pen.color := RGB(128,25,178);
canvas.moveto(x-400,y-70);
```

```
canvas.lineto(x-300,y-140);
canvas.lineto(x-200,y-70);
canvas.lineto(x-100,y-140);
canvas.lineto(x-0,y-70);
canvas.moveto(x-300,y-70);
canvas.lineto(x-200,y-140);
canvas.lineto(x-100,y-70);

canvas.pen.color := RGB(255,255,255);
canvas.moveto(x-0,y-220);
canvas.lineto(x-400,y-220);
canvas.lineto(x-400,y-300);
canvas.pen.color := RGB(128,25,178);
canvas.moveto(x-400,y-290);
canvas.lineto(x-300,y-290);
canvas.lineto(x-200,y-220);
canvas.lineto(x-100,y-290);
canvas.lineto(x-0,y-290);
canvas.moveto(x-300,y-220);
canvas.lineto(x-200,y-290);
canvas.lineto(x-100,y-220);

canvas.pen.color := RGB(255,255,255);
canvas.moveto(x-0,y-370);
canvas.lineto(x-400,y-370);
canvas.lineto(x-400,y-450);
canvas.pen.color := RGB(128,25,178);
canvas.moveto(x-400,y-440);
canvas.lineto(x-300,y-440);
canvas.lineto(x-200,y-370);
canvas.lineto(x-100,y-440);
canvas.lineto(x-0,y-440);
canvas.moveto(x-300,y-370);
canvas.lineto(x-200,y-440);
canvas.lineto(x-100,y-370);

end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}


procedure TForm1.Button2Click(Sender: TObject);
var
i : integer;
begin
{Input A-includes poking, displaying and transfer of Input A values}
```

```
Memo1.Visible := false;
for i := 1 to st do
begin
TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(18+i))).DDEConv:=
DDEClientConv1;
ListBox1.Items.Add(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(18+i))
).text);
end;

 begin
 if DDECli  <> Nil then
  begin
   res_1 := StrToFloat(Edit1.text);
   poke_1 := FloatToStr(res_1);
   res_2 := StrToFloat(Edit2.text);
   poke_2 := FloatToStr(res_2);
   res_3 := StrToFloat(Edit3.text);
   poke_3 := FloatToStr(res_3);

   DDECli.PokeData(DDEClientItem10.DDEItem, StrPCopy(sptr,poke_1));
   DDECli.PokeData(DDEClientItem11.DDEItem, StrPCopy(sptr,poke_2));
   DDECli.PokeData(DDEClientItem12.DDEItem, StrPCopy(sptr,poke_3));
 end
  else
   begin
    Edit1.Text := DDEClientItem10.text ;
    Edit2.Text := DDEClientItem11.text ;
   end;
  end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{Logic of the buttons appearance and disappearance{{{{{{{{{{{{}
Memo1.Clear;
Button2.visible := false;
Button3.visible := true;
Button1.visible := false;
Button5.visible := false;
Button4.visible := false;
Button6.visible := false;
Memo1.lines.add('Fuzzy logic results');

end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

procedure TForm1.Button3Click(Sender: TObject);
```

```
var
i : integer;
begin
{Input B-includes poking, displaying and transfer of Input B values}
for i := 1 to st do
begin
DDEClientItem(FindComponent('DDEClientItem'+IntToStr(23+i))).DDEConv:=
DDEClientConv1;
ListBox2.Items.Add(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(23+i))
).text);
end;

begin
 if DDECli <> Nil then
  begin
  res_4 := StrToFloat(Edit4.text);
  poke_4 := FloatToStr(res_4);
  res_5 := StrToFloat(Edit5.text);
  poke_5 := FloatToStr(res_5);
  res_6 := StrToFloat(Edit6.text);
  poke_6 := FloatToStr(res_6);

  DDECli.PokeData(DDEClientItem13.DDEItem, StrPCopy(sptr,poke_4));
  DDECli.PokeData(DDEClientItem14.DDEItem, StrPCopy(sptr,poke_5));
  DDECli.PokeData(DDEClientItem15.DDEItem, StrPCopy(sptr,poke_6));

  DDECli.PokeData(DDEClientItem16.DDEItem, StrPCopy(sptr,Edit7.text));
  DDECli.PokeData(DDEClientItem17.DDEItem, StrPCopy(sptr,Edit8.text));
  end
 else
  begin
   Edit4.Text := DDEClientItem13.text ;
   Edit5.Text := DDEClientItem14.text ;
   Edit6.Text := DDEClientItem15.text ;
  end;
end;

Button2.visible := false;
Button3.visible := false;
Button1.visible := true;
Button5.visible := false;
Button4.visible := false;
Button6.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
k: string;
i,j:integer;
xg:real;

R: TRect;
xc,yc: integer;
Color: LongInt;
Newrecta,Newrectb,Newreca,Newrecb: TRect;

begin
R :=getclientRect;
xc := R.right;
yc := R.bottom;
{{{{{{{{{{{{{{{{{{{Gradients of the Input triangles{{{{{{{{{{{{{}
pka := StrToFloat(Edit1.Text);
spa := StrToFloat(Edit2.Text);
xa := StrToFloat(Edit3.Text);

pkb := StrToFloat(Edit4.Text);
spb := StrToFloat(Edit5.Text);
xb := StrToFloat(Edit6.Text);

la1 := ((pka+spa)-xa)/((pka+spa)-pka);
la2 := (xa-pka)/((pka+spa)-pka);
la3 := (xa-(pka+spa))/((pka+2*spa)-(pka+spa));
la4 := ((pka+2*spa)-xa)/((pka+2*spa)-(pka+spa));
lb1 := ((pkb+spb)-xb)/((pkb+spb)-pkb);
lb2 := (xb-pkb)/((pkb+spb)-pkb);
lb3 := (xb-(pkb+spb))/((pkb+2*spb)-(pkb+spb));
lb4 := ((pkb+2*spb)-xb)/((pkb+2*spb)-(pkb+spb));
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{{{{{{{{{{{Clearing {{{{{{{{{{{{{{{{{{{{{{{{{}
Newreca := Rect(xc-400,yc-400,xc,yc-370);
Newrecb := Rect(xc-400,yc-250,xc,yc-220);
Canvas.fillrect(newreca);
Canvas.fillrect(newrecb);
Canvas.TextOut(xc-300,yc-220,'                                    ');
Canvas.TextOut(xc-200,yc-220,'                                    ');
Canvas.TextOut(xc-300,yc-220,FloattoStr(pkb));
Canvas.TextOut(xc-200,yc-220,FloattoStr(pkb+spb));
Canvas.TextOut(xc-100,yc-220,FloattoStr(pkb+2*spb));
```

```
Canvas.TextOut(xc-300,yc-370,'                    ');
Canvas.TextOut(xc-200,yc-370,'                    ');
Canvas.TextOut(xc-300,yc-370,FloattoStr(pka));
Canvas.TextOut(xc-200,yc-370,FloattoStr(pka+spa));
Canvas.TextOut(xc-100,yc-370,FloattoStr(pka+2*spa));
Newrecta := Rect(xc-400,yc-450,xc,yc-370);
Newrectb := Rect(xc-400,yc-290,xc,yc-220);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{Conditions for Input A and B - include redrawing of
graphs and graphical representation of the Fuzzy Rule}
  if (xa<pka+spa) and (xa>=pka) then
    begin
    ua1 := la1;
    ua2 := la2;
    ya1 := round(70*((100-(100/spa)*(xa-pka))/(100)));
    ya2 := round(70*((100/spa)*(xa-pka))/100);
    canvas.brush.color := clsilver;
    canvas.fillrect(newrecta);

    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-370);
    canvas.lineto(xc-400,yc-370);
    canvas.lineto(xc-400,yc-450);
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-440);
    canvas.lineto(xc-300,yc-440);
    canvas.lineto(xc-200,yc-370);
    canvas.lineto(xc-100,yc-440);
    canvas.lineto(xc-0,yc-440);
    canvas.moveto(xc-300,yc-370);
    canvas.lineto(xc-200,yc-440);
    canvas.lineto(xc-100,yc-370);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-370-(ya1));
    canvas.lineto(xc-300+round((100/(spa))*(xa-pka)),yc-370-(ya1));

    canvas.pen.color := RGB(125,0,125);
    canvas.moveto(xc-400,yc-370-(ya2));
    canvas.lineto(xc-300+round((100/(spa))*(xa-pka)),yc-370-(ya2));

    end;

  if (xa<=pka+2*spa) and (xa>=pka+spa) then
```

```
    begin
    ua1 := la4;
    ua2 := la3;
    ya1 := round(70*((100-(100/spa)*(xa-(pka+spa)))/(100)));
    ya2 := round(70*((100/spa)*(xa-(pka+spa)))/100);

    canvas.brush.color := clsilver;
    canvas.fillrect(newrecta);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-370);
    canvas.lineto(xc-400,yc-370);
    canvas.lineto(xc-400,yc-450);
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-440);
    canvas.lineto(xc-300,yc-440);
    canvas.lineto(xc-200,yc-370);
    canvas.lineto(xc-100,yc-440);
    canvas.lineto(xc-0,yc-440);
    canvas.moveto(xc-300,yc-370);
    canvas.lineto(xc-200,yc-440);
    canvas.lineto(xc-100,yc-370);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-370-(ya1));
    canvas.lineto(xc-200+round((100/(spa))*(xa-(pka+spa))),yc-370-(ya1));

    canvas.pen.color := RGB(125,0,125);
    canvas.moveto(xc-400,yc-370-(ya2));
    canvas.lineto(xc-200+round((100/(spa))*(xa-(pka+spa))),yc-370-(ya2));

    end;

if (xb<pkb+spb) and (xb>=pkb) then
    begin
    ub1 := lb1;
    ub2 := lb2;
    yb1 := round(70*((100-(100/spb)*(xb-pkb))/(100)));
    yb2 := round(70*((100/spb)*(xb-pkb))/100);

    canvas.brush.color := clsilver;
    canvas.fillrect(newrectb);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-220);
    canvas.lineto(xc-400,yc-220);
    canvas.lineto(xc-400,yc-290);
```

```
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-290);
    canvas.lineto(xc-300,yc-290);
    canvas.lineto(xc-200,yc-220);
    canvas.lineto(xc-100,yc-290);
    canvas.lineto(xc-0,yc-290);
    canvas.moveto(xc-300,yc-220);
    canvas.lineto(xc-200,yc-290);
    canvas.lineto(xc-100,yc-220);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-220-(yb1));
    canvas.lineto(xc-300+round((100/(spb))*(xb-pkb)),yc-220-(yb1));

    canvas.pen.color := RGB(125,0,125);
    canvas.moveto(xc-400,yc-220-(yb2));
    canvas.lineto(xc-300+round((100/(spb))*(xb-pkb)),yc-220-(yb2));

    end;

if (xb<=pkb+2*spb) and (xb>=pkb+spb) then
    begin
    ub1 := lb4;
    ub2 := lb3;
    yb1 := round(70*((100-(100/spb)*(xb-(pkb+spb)))/(100)));
    yb2 := round(70*((100/spb)*(xb-(pkb+spb)))/100);

    canvas.brush.color := clsilver;
    canvas.fillrect(newrectb);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-220);
    canvas.lineto(xc-400,yc-220);
    canvas.lineto(xc-400,yc-290);
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-290);
    canvas.lineto(xc-300,yc-290);
    canvas.lineto(xc-200,yc-220);
    canvas.lineto(xc-100,yc-290);
    canvas.lineto(xc-0,yc-290);
    canvas.moveto(xc-300,yc-220);
    canvas.lineto(xc-200,yc-290);
    canvas.lineto(xc-100,yc-220);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-220-(yb1));
```

```
    canvas.lineto(xc-200+round((100/(spb))*(xb-(pkb+spb))),yc-220-(yb1));

    canvas.pen.color := RGB(125,0,125);
    canvas.moveto(xc-400,yc-220-(yb2));
    canvas.lineto(xc-200+round((100/(spb))*(xb-(pkb+spb))),yc-220-(yb2));

    end;

  if(xa<pka) then
    begin
    ua1 := 2;
    ua2 := -333;
    ya1 := 70;
    ya2 := 0;

    canvas.brush.color := clsilver;
    canvas.fillrect(newrecta);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-370);
    canvas.lineto(xc-400,yc-370);
    canvas.lineto(xc-400,yc-440);
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-440);
    canvas.lineto(xc-300,yc-440);
    canvas.lineto(xc-200,yc-370);
    canvas.lineto(xc-100,yc-440);
    canvas.lineto(xc-0,yc-440);
    canvas.moveto(xc-300,yc-370);
    canvas.lineto(xc-200,yc-440);
    canvas.lineto(xc-100,yc-370);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-370-(ya1));
    canvas.lineto(xc-400+round((100/pka)*xa),yc-370-(ya1));

    end;

  if(xb<pkb) then
    begin
    ub1 := 2;
    ub2 := -2222;
    yb1 := 70;
    yb2 := 0;

    canvas.brush.color := clsilver;
```

```
    canvas.fillrect(newrectb);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-220);
    canvas.lineto(xc-400,yc-220);
    canvas.lineto(xc-400,yc-290);

    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-290);
    canvas.lineto(xc-300,yc-290);
    canvas.lineto(xc-200,yc-220);
    canvas.lineto(xc-100,yc-290);
    canvas.lineto(xc-0,yc-290);
    canvas.moveto(xc-300,yc-220);
    canvas.lineto(xc-200,yc-290);
    canvas.lineto(xc-100,yc-220);

    canvas.pen.color := RGB(255,255,125);
    canvas.moveto(xc-400,yc-220-(yb1));
    canvas.lineto(xc-400+round((100/pkb)*xb),yc-220-(yb1));

    end;

if(xa>pka+2*spa)then
    begin
    ua1 := -1110;
    ua2 := 2;
    ya1 := 0;
    ya2 := 70;

    canvas.brush.color := clsilver;
    canvas.fillrect(newrecta);
    canvas.pen.color := RGB(255,255,255);
    canvas.moveto(xc-0,yc-370);
    canvas.lineto(xc-400,yc-370);
    canvas.lineto(xc-400,yc-440);
    canvas.pen.color := RGB(128,37,78);
    canvas.moveto(xc-400,yc-440);
    canvas.lineto(xc-300,yc-440);
    canvas.lineto(xc-200,yc-370);
    canvas.lineto(xc-100,yc-440);
    canvas.lineto(xc-0,yc-440);
    canvas.moveto(xc-300,yc-370);
    canvas.lineto(xc-200,yc-440);
    canvas.lineto(xc-100,yc-370);
```

```
        canvas.pen.color := RGB(255,255,125);
        canvas.moveto(xc-400,yc-370-(ya2));
        canvas.lineto(xc-100+round((100/spa)*(xa-(pka+2*spa))),yc-370-(ya2));

        end;

     if(xb>pkb+2*spb) then
        begin
        ub1 := -10000;
        ub2 := 2;
        yb1 := 0;
        yb2 := 70;

        canvas.brush.color := clsilver;
        canvas.fillrect(newrectb);
        canvas.pen.color := RGB(255,255,255);
        canvas.moveto(xc-0,yc-220);
        canvas.lineto(xc-400,yc-220);
        canvas.lineto(xc-400,yc-290);

        canvas.pen.color := RGB(128,37,78);
        canvas.moveto(xc-400,yc-290);
        canvas.lineto(xc-300,yc-290);
        canvas.lineto(xc-200,yc-220);
        canvas.lineto(xc-100,yc-290);
        canvas.lineto(xc-0,yc-290);
        canvas.moveto(xc-300,yc-220);
        canvas.lineto(xc-200,yc-290);
        canvas.lineto(xc-100,yc-220);

        canvas.pen.color := RGB(255,255,125);
        canvas.moveto(xc-400,yc-220-(yb2));
        canvas.lineto(xc-100+round((100/spb)*(xb-(pkb+2*spb))),yc-220-(yb2));
        end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{Logic of the buttons appearance and disappearance{{{{{{{{{{{{}
Button2.visible := false;
Button3.visible := false;
Button1.visible := false;
Button5.visible := true;
Button4.visible := true;
Button6.visible := true;
end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
```

```
procedure TForm1.Button4Click(Sender: TObject);
label Higha1,Higha2,Highb1,Highb2,Mediuma1,Mediuma2,Mediumb1,Mediumb2;
label Lowa1,Lowa2,Lowb1,Lowb2,koniec,level2;
var
num:real;
  R: TRect;
  xc,yc:integer;
  k1,k2,h1,h2,hm,spc,pkc:real;
  Color: LongInt;
  Newrectc: TRect;
begin

begin
R :=getclientRect;
xc := R.right;
yc := R.bottom;
Newrectc := Rect(xc-400,yc-140,xc,yc-70);
k1 := 0;
k2 := 0;

{{{{{{{{{{{{{{{{{Peak and space in output triangles{{{{{{{{{{{{{}
pkc := StrToFloat(Edit7.Text);
spc := StrToFloat(Edit8.Text);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{Clearing and recreation of the bottom drawing - output function{}
Canvas.TextOut(xc-300,yc-70,'                                    ');
Canvas.TextOut(xc-100,yc-70,'                                    ');
Canvas.TextOut(xc-300,yc-70,FloattoStr(pkc));
Canvas.TextOut(xc-200,yc-70,FloattoStr(pkc+spc));
Canvas.TextOut(xc-100,yc-70,FloattoStr(pkc+2*spc));

canvas.brush.color := clsilver;
canvas.fillrect(newrectc);
canvas.pen.color := RGB(255,255,255);
canvas.moveto(xc-0,yc-70);
canvas.lineto(xc-400,yc-70);
canvas.lineto(xc-400,yc-150);
canvas.pen.color := RGB(128,25,178);
canvas.moveto(xc-400,yc-70);
canvas.lineto(xc-300,yc-140);
canvas.lineto(xc-200,yc-70);
canvas.lineto(xc-100,yc-140);
canvas.lineto(xc-0,yc-70);
canvas.moveto(xc-300,yc-70);
```

```
canvas.lineto(xc-200,yc-140);
canvas.lineto(xc-100,yc-70);

canvas.pen.color := RGB(128,128,25);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{Process of defuzzification of all allowed fuzzy rules{{{}
    if(ua1 = 2) and (ub1 = 2) then
     begin
     num := StrToFloat(StringGrid1.Cells[1,1]);
     k1:=1;
     k2:=1;
     if(num = 1) then
      begin
      canvas.moveto(xc-299,yc-140);
      canvas.lineto(xc-301,yc-140);
      h1:=pkc;
      h2:=pkc;
      end;
      if(num = 2) then
       begin
       canvas.moveto(xc-199,yc-140);
       canvas.lineto(xc-201,yc-140);
       h1:=pkc+spc;
       h2:=pkc+spc;
       end;
      if(num = 3) then
       begin
       canvas.moveto(xc-99,yc-140);
       canvas.lineto(xc-101,yc-140);
       h1:=pkc+2*spc;
       h2:=pkc+2*spc;
       end;
     end;

    if(ua1 = 2) and (ub2 = 2) then
     begin
     num := StrToFloat(StringGrid1.Cells[1,3]);
     k1:=1;
     k2:=1;
     if(num = 1) then
      begin
      canvas.moveto(xc-299,yc-140);
      canvas.lineto(xc-301,yc-140);
      h1:=pkc;
```

```
  h2:=pkc;
  end;
  if(num = 2) then
   begin
   canvas.moveto(xc-199,yc-140);
   canvas.lineto(xc-201,yc-140);
   h1:=pkc+spc;
   h2:=pkc+spc;
   end;
  if(num = 3) then
   begin
   canvas.moveto(xc-99,yc-140);
   canvas.lineto(xc-101,yc-140);
   h1:=pkc+2*spc;
   h2:=pkc+2*spc;
   end;
 end;

if(ua2 = 2) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 k1:=1;
 k2:=1;
 if(num = 1) then
  begin
  canvas.moveto(xc-299,yc-140);
  canvas.lineto(xc-301,yc-140);
  h1:=pkc;
  h2:=pkc;
  end;
  if(num = 2) then
   begin
   canvas.moveto(xc-199,yc-140);
   canvas.lineto(xc-201,yc-140);
   h1:=pkc+spc;
   h2:=pkc+spc;
   end;
  if(num = 3) then
   begin
   canvas.moveto(xc-99,yc-140);
   canvas.lineto(xc-101,yc-140);
   h1:=pkc+2*spc;
   h2:=pkc+2*spc;
   end;
 end;
```

```
if(ua2 = 2) and (ub2 = 2) then
 begin
 num :=StrToFloat(StringGrid1.Cells[3,3]);
 k1:=1;
 k2:=1;
 if(num = 1) then
  begin
  canvas.moveto(xc-299,yc-140);
  canvas.lineto(xc-301,yc-140);
  h1:=pkc;
  h2:=pkc;
  end;
  if(num = 2) then
   begin
   canvas.moveto(xc-199,yc-140);
   canvas.lineto(xc-201,yc-140);
   h1:=pkc+spc;
   h2:=pkc+spc;
   end;
  if(num = 3) then
   begin
   canvas.moveto(xc-99,yc-140);
   canvas.lineto(xc-101,yc-140);
   h1:=pkc+2*spc;
   h2:=pkc+2*spc;
   end;
 end;

if(ua1 = 2) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua1 = 2) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,2]);
 if(num = 1) then
  goto Lowb1;
```

```
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua1 = la1) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua1 = la1) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la1) and (ub1 = lb4) then
 begin
 num :=StrToFloat(StringGrid1.Cells[1,2]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
```

```
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la1) and (ub2 = 2) then
 begin
 num :=StrToFloat(StringGrid1.Cells[1,3]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua1 = la4) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua1 = la4) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;
```

```
if(ua1 = la4) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,2]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la4) and (ub2 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,3]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua2 = 2) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua2 = 2) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,2]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
```

```
 goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

level2:

 if(ua2 = la3) and (ub2 = lb3) then
  begin
  num := StrToFloat(StringGrid1.Cells[3,3]);
  if(num = 1) and (ub2<=ua2) then
   goto Lowb2;
  if(num = 1) and (ub2>=ua2) then
   goto Lowa2;
  if(num = 2) and (ub2<=ua2) then
   goto Mediumb2;
  if (num =2) and (ub2>=ua2) then
   goto Mediuma2;
  if(num = 3) and (ub2<=ua2) then
   goto Highb2;
  if(num = 3) and (ub2>=ua2) then
   goto Higha2;
 end;

 if(ua1 = 2) and (ub2 = lb2) then
  begin
  num := StrToFloat(StringGrid1.Cells[1,2]);
  if(num = 1) then
   goto Lowb2;
  if(num = 2) then
   goto Mediumb2;
  if(num = 3) then
   goto Highb2;
  end;

 if(ua1 = 2) and (ub2 = lb3) then
  begin
  num := StrToFloat(StringGrid1.Cells[1,3]);
  if(num = 1) then
   goto Lowb2;
  if(num = 2) then
   goto Mediumb2;
  if(num = 3) then
   goto Highb2;
  end;
```

```
if(ua2 = la2) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
 end;

if(ua2 = la2) and (ub2 = lb2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,2]);
 if(num = 1) and (ub2<=ua2) then
  goto Lowb2;
 if(num = 1) and (ub2>=ua2) then
  goto Lowa2;
 if(num = 2) and (ub2<=ua2) then
  goto Mediumb2;
 if (num =2) and (ub2>=ua2) then
  goto Mediuma2;
 if(num = 3) and (ub2<=ua2) then
  goto Highb2;
 if(num = 3) and (ub2>=ua2) then
  goto Higha2;
end;

if(ua2 = la2) and (ub2 = 1) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,3]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
 end;

if(ua2 = la2) and (ub2 = lb3) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,3]);
 if(num = 1) and (ub2<=ua2) then
  goto Lowb2;
```

```
 if(num = 1) and (ub2>=ua2) then
  goto Lowa2;
 if(num = 2) and (ub2<=ua2) then
  goto Mediumb2;
 if (num =2) and (ub2>=ua2) then
  goto Mediuma2;
 if(num = 3) and (ub2<=ua2) then
  goto Highb2;
 if(num = 3) and (ub2>=ua2) then
  goto Higha2;
end;

if(ua2 = 2) and (ub2 = lb3) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,2]);
 if(num = 1) then
  goto Lowb2;
 if(num = 2) then
  goto Mediumb2;
 if(num = 3) then
  goto Highb2;
 end;

if(ua2 = 2) and (ub2 = lb3) then
 begin
 num :=StrToFloat(StringGrid1.Cells[3,3]);
 if(num = 1) then
  goto Lowb2;
 if(num = 2) then
  goto Mediumb2;
 if(num = 3) then
  goto Highb2;
 end;

if(ua2 = la3) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
 end;
```

```
        if(ua2 = la3) and (ub2 = lb2) then
         begin
         num := StrToFloat(StringGrid1.Cells[3,2]);
         if(num = 1) and (ub2<=ua2) then
          goto Lowb2;
         if(num = 1) and (ub2>=ua2) then
          goto Lowa2;
         if(num = 2) and (ub2<=ua2) then
          goto Mediumb2;
         if (num =2) and (ub2>=ua2) then
          goto Mediuma2;
         if(num = 3) and (ub2<=ua2) then
          goto Highb2;
         if(num = 3) and (ub2>=ua2) then
          goto Higha2;
        end;

        if(ua2 = la3) and (ub2 = 2) then
         begin
         num := StrToFloat(StringGrid1.Cells[3,3]);
         if(num = 1) then
          goto Lowa2;
         if(num = 2) then
          goto Mediuma2;
         if(num = 3) then
          goto Higha2;
         end;

goto koniec;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{The graphical representation of the output function and the
numerical setup for the Height method{{{{}
Higha1:
canvas.moveto(1+round((yc-70-ya1-yc-0.7*xc+210)/(-0.7)),yc-70-(ya1));
canvas.lineto(round((yc-70-ya1-yc+0.7*xc+70)/0.7)-1,yc-70-(ya1));
k1:=ua1;
h1:=pkc+2*spc;
goto level2;

Higha2:
canvas.moveto(1+round((yc-70-ya2-yc-0.7*xc+210)/(-0.7)),yc-70-(ya2));
canvas.lineto(round((yc-70-ya2-yc+0.7*xc+70)/0.7)-1,yc-70-(ya2));
k2:=ua2;
h2:=pkc+2*spc;
```

```
goto koniec;

Highb1:
canvas.moveto(1+round((yc-70-yb1-yc-0.7*xc+210)/(-0.7)),yc-70-(yb1));
canvas.lineto(round((yc-70-yb1-yc+0.7*xc+70)/0.7)-1,yc-70-(yb1));
k1:=ub1;
h1:=pkc+2*spc;
goto level2;

Highb2:
canvas.moveto(1+round((yc-70-yb2-yc-0.7*xc+210)/(-0.7)),yc-70-(yb2));
canvas.lineto(round((yc-70-yb2-yc+0.7*xc+70)/0.7)-1,yc-70-(yb2));
k2:=ub2;
h2:=pkc+2*spc;
goto koniec;

Mediuma1:
canvas.moveto(1+round((yc-70-ya1-yc-0.7*xc+280)/(-0.7)),yc-70-(ya1));
canvas.lineto(round((yc-70-ya1-yc+0.7*xc-0)/0.7)-1,yc-70-(ya1));
k1:=ua1;
h1:=pkc+spc;
goto level2;

Mediuma2:
canvas.moveto(1+round((yc-70-ya2-yc-0.7*xc+280)/(-0.7)),yc-70-(ya2));
canvas.lineto(round((yc-70-ya2-yc+0.7*xc-0)/0.7)-1,yc-70-(ya2));
k2:=ua2;
h2:=pkc+spc;
goto koniec;

Mediumb1:
canvas.moveto(1+round((yc-70-yb1-yc-0.7*xc+280)/(-0.7)),yc-70-(yb1));
canvas.lineto(round((yc-70-yb1-yc+0.7*xc-0)/0.7)-1,yc-70-(yb1));
k1:=ub1;
h1:=pkc+spc;
goto level2;

Mediumb2:
canvas.moveto(1+round((yc-70-yb2-yc-0.7*xc+280)/(-0.7)),yc-70-(yb2));
canvas.lineto(round((yc-70-yb2-yc+0.7*xc-0)/0.7)-1,yc-70-(yb2));
k2:=ub2;
h2:=pkc+spc;
goto koniec;

Lowa1:
```

```
canvas.moveto(1+round((yc-70-ya1-yc-0.7*xc+350)/(-0.7)),yc-70-(ya1));
canvas.lineto(round((yc-70-ya1-yc+0.7*xc-70)/0.7)-1,yc-70-(ya1));
k1:=ua1;
h1:=pkc;
goto level2;

Lowa2:
canvas.moveto(1+round((yc-70-ya2-yc-0.7*xc+350)/(-0.7)),yc-70-(ya2));
canvas.lineto(round((yc-70-ya2-yc+0.7*xc-70)/0.7)-1,yc-70-(ya2));
k2:=ua2;
h2:=pkc;
goto koniec;

Lowb1:
canvas.moveto(1+round((yc-70-yb1-yc-0.7*xc+350)/(-0.7)),yc-70-(yb1));
canvas.lineto(round((yc-70-yb1-yc+0.7*xc-70)/0.7)-1,yc-70-(yb1));
k1:=ub1;
h1:=pkc;
goto level2;

Lowb2:
canvas.moveto(1+round((yc-70-yb2-yc-0.7*xc+350)/(-0.7)),yc-70-(yb2));
canvas.lineto(round((yc-70-yb2-yc+0.7*xc-70)/0.7)-1,yc-70-(yb2));
k2:=ub2;
h2:=pkc;
goto koniec;

koniec:
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{{{{The Height method{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
hm := (k1*h1+k2*h2)/(k1+k2);
Edit9.Text := FloatToStr(hm);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;


{{{{{Logic of the buttons appearance and disappearance{{{{{{{{{{{{{}
Button2.visible := true;
Button3.visible := false;
Button1.visible := false;
Button5.visible := false;
Button4.visible := true;
Button6.visible := false;
StringGrid1.Visible := False;
```

```
        )

end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

procedure TForm1.Button5Click(Sender: TObject);
 begin
{{{{{{{{{{{{{{{{{{Display of the Fuzzy Logic Rule{{{{{{{{{{{{{{{{{{{}
  StringGrid1.Visible := True;
  StringGrid1.Height := 129;
  StringGrid1.Width := 329;
  StringGrid1.Left := 168;
  StringGrid1.Top := 320;
 end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}


procedure TForm1.Button6Click(Sender: TObject);
{{{{Display of the Output values of the Fuzzy Logic optimisation.
This section resembles Button4 (Close Rule/Defuzzify) because
it was computationally impossible to display the graphical
results and to activate the numerical values at the same time{{{{{{{}

label Higha1,Higha2,Highb1,Highb2,Mediuma1,Mediuma2,Mediumb1,Mediumb2;
label Lowa1,Lowa2,Lowb1,Lowb2,koniec,level2;
var
i : integer;
hm,x_a,x_b,k1,k2,num,h1,h2,pkc,spc :real;
begin

{{{{{{{{{{{{{{{{{{{{{{{Peak and space for Input A triangles{{{{{{{{{{}
pka := StrToFloat(Edit1.Text);
spa := StrToFloat(Edit2.Text);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

Memo1.Visible := true;
{{{{{{{{{{{{{{{{{{{{{{{Peak and space for Input B triangles{{{{{{{{{{}
pkb := StrToFloat(Edit4.Text);
spb := StrToFloat(Edit5.Text);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

for i := 1 to st do
begin
{{{{{{{{{{{{{{{{{{{{{{{{{{{Inport of Input A and B from Excel{{{{{{{{}
TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(18+i))).DDEConv:=
DDEClientconv1;
```

```
TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(23+i))).DDEConv:=
DDEClientconv1;
x_a :=
StrToFloat(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(18+i))).Text);
x_b :=
StrToFloat(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(23+i))).Text);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{{{{{{{{{{{Gradients of Input A and B triangles{{{{{{}
la1 := ((pka+spa)-xa)/((pka+spa)-pka);
la2 := (x_a-pka)/((pka+spa)-pka);
la3 := (x_a-(pka+spa))/((pka+2*spa)-(pka+spa));
la4 := ((pka+2*spa)-x_a)/((pka+2*spa)-(pka+spa));
lb1 := ((pkb+spb)-x_b)/((pkb+spb)-pkb);
lb2 := (x_b-pkb)/((pkb+spb)-pkb);
lb3 := (x_b-(pkb+spb))/((pkb+2*spb)-(pkb+spb));
lb4 := ((pkb+2*spb)-x_b)/((pkb+2*spb)-(pkb+spb));
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{{{{{{{Membership Function of Input A and B{{{{{{{{{{}
  if (x_a<pka+spa) and (x_a>=pka) then
    begin
    ua1 := la1;
    ua2 := la2;
    ya1 := round(70*((100-(100/spa)*round(x_a-pka))/(100)));
    ya2 := round(70*((100/spa)*round(x_a-pka))/100);
    end;

  if (x_a<=pka+2*spa) and (x_a>=pka+spa) then
    begin
    ua1 := la4;
    ua2 := la3;
    ya1 := round(70*((100-(100/spa)*round(x_a-(pka+spa)))/(100)));
    ya2 := round(70*((100/spa)*round(x_a-(pka+spa)))/100);
    end;

  if (x_b<pkb+spb) and (x_b>=pkb) then
    begin
    ub1 := lb1;
    ub2 := lb2;
    yb1 := round(70*((100-(100/spb)*round(x_b-pkb))/(100)));
    yb2 := round(70*((100/spb)*round(x_b-pkb))/100);
    end;

  if (x_b<=pkb+2*spb) and (x_b>=pkb+spb) then
```

```
      begin
      ub1 := lb4;
      ub2 := lb3;
      yb1 := round(70*((100-(100/spb)*round(x_b-(pkb+spb)))/(100)));
      yb2 := round(70*((100/spb)*round(x_b-(pkb+spb)))/100);
      end;

    if(x_a<pka) then
      begin
      ua1 := 2;
      ua2 := -333;
      ya1 := 70;
      ya2 := 0;
      end;

    if(x_b<pkb) then
      begin
      ub1 := 2;
      ub2 := -2222;
      yb1 := 70;
      yb2 := 0;
      end;

    if(x_a>pka+2*spa)then
      begin
      ua1 := -1110;
      ua2 := 2;
      ya1 := 0;
      ya2 := 70;
      end;

    if(x_b>pkb+2*spb) then
      begin
      ub1 := -10000;
      ub2 := 2;
      yb1 := 0;
      yb2 := 70;
      end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

k1 := 0;
k2 := 0;

{{{{{{{{{{{{{{{{{{Peak and space of the Output triangles{{{{{{{{{{}
pkc := StrToFloat(Edit7.Text);
```

```
spc := StrToFloat(Edit8.Text);
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{Process of defuzzification of all allowed fuzzy rules {{{{{{{}
     if(ua1 = 2) and (ub1 = 2) then
      begin
      num := StrToFloat(StringGrid1.Cells[1,1]);
      k1:=1;
      k2:=1;
      if(num = 1) then
       begin
       h1:=pkc;
       h2:=pkc;
       end;
       if(num = 2) then
        begin
        h1:=pkc+spc;
        h2:=pkc+spc;
        end;
       if(num = 3) then
        begin
        h1:=pkc+2*spc;
        h2:=pkc+2*spc;
        end;
      end;

     if(ua1 = 2) and (ub2 = 2) then
      begin
      num := StrToFloat(StringGrid1.Cells[1,3]);
      k1:=1;
      k2:=1;
      if(num = 1) then
       begin
       h1:=pkc;
       h2:=pkc;
       end;
      if(num = 2) then
       begin
       h1:=pkc+spc;
       h2:=pkc+spc;
       end;
      if(num = 3) then
       begin
       h1:=pkc+2*spc;
       h2:=pkc+2*spc;
```

```
 end;
end;

if(ua2 = 2) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 k1:=1;
 k2:=1;
 if(num = 1) then
  begin
  h1:=pkc;
  h2:=pkc;
  end;
 if(num = 2) then
  begin
  h1:=pkc+spc;
  h2:=pkc+spc;
  end;
 if(num = 3) then
  begin
  h1:=pkc+2*spc;
  h2:=pkc+2*spc;
  end;
 end;

if(ua2 = 2) and (ub2 = 2) then
 begin
 num :=StrToFloat(StringGrid1.Cells[3,3]);
 k1:=1;
 k2:=1;
 if(num = 1) then
  begin
  h1:=pkc;
  h2:=pkc;
  end;
 if(num = 2) then
  begin
  h1:=pkc+spc;
  h2:=pkc+spc;
   end;
 if(num = 3) then
  begin
  h1:=pkc+2*spc;
  h2:=pkc+2*spc;
  end;
```

```
 end;

if(ua1 = 2) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua1 = 2) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,2]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua1 = la1) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua1 = la1) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,1]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
```

```
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la1) and (ub1 = lb4) then
 begin
 num :=StrToFloat(StringGrid1.Cells[1,2]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la1) and (ub2 = 2) then
 begin
 num :=StrToFloat(StringGrid1.Cells[1,3]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;

if(ua1 = la4) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;
```

```
if(ua1 = la4) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la4) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,2]);
 if(num = 1) and (ub1<=ua1) then
  goto Lowb1;
 if(num = 1) and (ub1>=ua1) then
  goto Lowa1;
 if(num = 2) and (ub1<=ua1) then
  goto Mediumb1;
 if (num =2) and (ub1>=ua1) then
  goto Mediuma1;
 if(num = 3) and (ub1<=ua1) then
  goto Highb1;
 if(num = 3) and (ub1>=ua1) then
  goto Higha1;
end;

if(ua1 = la4) and (ub2 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,3]);
 if(num = 1) then
  goto Lowa1;
 if(num = 2) then
  goto Mediuma1;
 if(num = 3) then
  goto Higha1;
 end;
```

```
if(ua2 = 2) and (ub1 = lb1) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

if(ua2 = 2) and (ub1 = lb4) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,2]);
 if(num = 1) then
  goto Lowb1;
 if(num = 2) then
  goto Mediumb1;
 if(num = 3) then
  goto Highb1;
 end;

level2:

if(ua2 = la3) and (ub2 = lb3) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,3]);
 if(num = 1) and (ub2<=ua2) then
  goto Lowb2;
 if(num = 1) and (ub2>=ua2) then
  goto Lowa2;
 if(num = 2) and (ub2<=ua2) then
  goto Mediumb2;
 if (num =2) and (ub2>=ua2) then
  goto Mediuma2;
 if(num = 3) and (ub2<=ua2) then
  goto Highb2;
 if(num = 3) and (ub2>=ua2) then
  goto Higha2;
end;


if(ua1 = 2) and (ub2 = lb2) then
 begin
```

```
 num := StrToFloat(StringGrid1.Cells[1,2]);
 if(num = 1) then
  goto Lowb2;
 if(num = 2) then
  goto Mediumb2;
 if(num = 3) then
  goto Highb2;
 end;

if(ua1 = 2) and (ub2 = lb3) then
 begin
 num := StrToFloat(StringGrid1.Cells[1,3]);
 if(num = 1) then
  goto Lowb2;
 if(num = 2) then
  goto Mediumb2;
 if(num = 3) then
  goto Highb2;
 end;

if(ua2 = la2) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,1]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
 end;

if(ua2 = la2) and (ub2 = lb2) then
 begin
 num := StrToFloat(StringGrid1.Cells[2,2]);
 if(num = 1) and (ub2<=ua2) then
  goto Lowb2;
 if(num = 1) and (ub2>=ua2) then
  goto Lowa2;
 if(num = 2) and (ub2<=ua2) then
  goto Mediumb2;
 if (num =2) and (ub2>=ua2) then
  goto Mediuma2;
 if(num = 3) and (ub2<=ua2) then
  goto Highb2;
 if(num = 3) and (ub2>=ua2) then
```

```
  goto Higha2;
 end;

 if(ua2 = la2) and (ub2 = 1) then
  begin
  num := StrToFloat(StringGrid1.Cells[2,3]);
  if(num = 1) then
   goto Lowa2;
  if(num = 2) then
   goto Mediuma2;
  if(num = 3) then
   goto Higha2;
  end;

 if(ua2 = la2) and (ub2 = lb3) then
  begin
  num := StrToFloat(StringGrid1.Cells[2,3]);
  if(num = 1) and (ub2<=ua2) then
   goto Lowb2;
  if(num = 1) and (ub2>=ua2) then
   goto Lowa2;
  if(num = 2) and (ub2<=ua2) then
   goto Mediumb2;
  if (num =2) and (ub2>=ua2) then
   goto Mediuma2;
  if(num = 3) and (ub2<=ua2) then
   goto Highb2;
  if(num = 3) and (ub2>=ua2) then
   goto Higha2;
 end;

 if(ua2 = 2) and (ub2 = lb3) then
  begin
  num := StrToFloat(StringGrid1.Cells[3,2]);
  if(num = 1) then
   goto Lowb2;
  if(num = 2) then
   goto Mediumb2;
  if(num = 3) then
   goto Highb2;
  end;

 if(ua2 = 2) and (ub2 = lb3) then
  begin
  num :=StrToFloat(StringGrid1.Cells[3,3]);
```

```
 if(num = 1) then
  goto Lowb2;
 if(num = 2) then
  goto Mediumb2;
 if(num = 3) then
  goto Highb2;
 end;

if(ua2 = la3) and (ub1 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,1]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
 end;

if(ua2 = la3) and (ub2 = lb2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,2]);
 if(num = 1) and (ub2<=ua2) then
  goto Lowb2;
 if(num = 1) and (ub2>=ua2) then
  goto Lowa2;
 if(num = 2) and (ub2<=ua2) then
  goto Mediumb2;
 if (num =2) and (ub2>=ua2) then
  goto Mediuma2;
 if(num = 3) and (ub2<=ua2) then
  goto Highb2;
 if(num = 3) and (ub2>=ua2) then
  goto Higha2;
end;

if(ua2 = la3) and (ub2 = 2) then
 begin
 num := StrToFloat(StringGrid1.Cells[3,3]);
 if(num = 1) then
  goto Lowa2;
 if(num = 2) then
  goto Mediuma2;
 if(num = 3) then
  goto Higha2;
```

```
    end;

goto koniec;


Higha1:
k1:=ua1;
h1:=pkc+2*spc;
goto level2;

Higha2:
k2:=ua2;
h2:=pkc+2*spc;
goto koniec;

Highb1:
k1:=ub1;
h1:=pkc+2*spc;
goto level2;

Highb2:
k2:=ub2;
h2:=pkc+2*spc;
goto koniec;

Mediuma1:
k1:=ua1;
h1:=pkc+spc;
goto level2;

Mediuma2:
k2:=ua2;
h2:=pkc+spc;
goto koniec;

Mediumb1:
k1:=ub1;
h1:=pkc+spc;
goto level2;

Mediumb2:
k2:=ub2;
h2:=pkc+spc;
goto koniec;
```

```
Lowa1:
k1:=ua1;
h1:=pkc;
goto level2;

Lowa2:
k2:=ua2;
h2:=pkc;
goto koniec;

Lowb1:
k1:=ub1;
h1:=pkc;
goto level2;

Lowb2:
k2:=ub2;
h2:=pkc;
goto koniec;

koniec:
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{5 inputs (from Excel) to the Fuzzy Logic for each proposal{{{{{{}
 If StrToFloat(SpinEdit1.Text) = 1   then
  begin
  DDEClientItem29.DDEItem := 'R165C2';
  DDEClientItem30.DDEItem := 'R165C3';
  DDEClientItem31.DDEItem := 'R165C4';
  DDEClientItem32.DDEItem := 'R165C5';
  DDEClientItem33.DDEItem := 'R165C6';
    end;
  if StrToFloat(SpinEdit1.Text) = 2  then
  begin
  DDEClientItem29.DDEItem := 'R166C2';
  DDEClientItem30.DDEItem := 'R166C3';
  DDEClientItem31.DDEItem := 'R166C4';
  DDEClientItem32.DDEItem := 'R166C5';
  DDEClientItem33.DDEItem := 'R166C6';
  end
  Else if StrToFloat(SpinEdit1.Text) = 3  then
  begin
  DDEClientItem29.DDEItem := 'R167C2';
  DDEClientItem30.DDEItem := 'R167C3';
  DDEClientItem31.DDEItem := 'R167C4';
```

```
DDEClientItem32.DDEItem := 'R167C5';
DDEClientItem33.DDEItem := 'R167C6';
end
Else if StrToFloat(SpinEdit1.Text) = 4 then
begin
DDEClientItem29.DDEItem := 'R168C2';
DDEClientItem30.DDEItem := 'R168C3';
DDEClientItem31.DDEItem := 'R168C4';
DDEClientItem32.DDEItem := 'R168C5';
DDEClientItem33.DDEItem := 'R168C6';
end
Else if StrToFloat(SpinEdit1.Text) = 5 then
begin
DDEClientItem29.DDEItem := 'R169C2';
DDEClientItem30.DDEItem := 'R169C3';
DDEClientItem31.DDEItem := 'R169C4';
DDEClientItem32.DDEItem := 'R169C5';
DDEClientItem33.DDEItem := 'R169C6';
end
Else if StrToFloat(SpinEdit1.Text) = 6 then
begin
DDEClientItem29.DDEItem := 'R170C2';
DDEClientItem30.DDEItem := 'R170C3';
DDEClientItem31.DDEItem := 'R170C4';
DDEClientItem32.DDEItem := 'R170C5';
DDEClientItem33.DDEItem := 'R170C6';
end
 Else if StrToFloat(SpinEdit1.Text) = 7 then
begin
DDEClientItem29.DDEItem := 'R171C2';
DDEClientItem30.DDEItem := 'R171C3';
DDEClientItem31.DDEItem := 'R171C4';
DDEClientItem32.DDEItem := 'R171C5';
DDEClientItem33.DDEItem := 'R171C6';
end
  Else if StrToFloat(SpinEdit1.Text) = 8 then
begin
DDEClientItem29.DDEItem := 'R172C2';
DDEClientItem30.DDEItem := 'R172C3';
DDEClientItem31.DDEItem := 'R172C4';
DDEClientItem32.DDEItem := 'R172C5';
DDEClientItem33.DDEItem := 'R172C6';
end
   Else if StrToFloat(SpinEdit1.Text) = 9 then
begin
```

```
  DDEClientItem29.DDEItem := 'R173C2';
  DDEClientItem30.DDEItem := 'R173C3';
  DDEClientItem31.DDEItem := 'R173C4';
  DDEClientItem32.DDEItem := 'R173C5';
  DDEClientItem33.DDEItem := 'R173C6';
  end
   Else if StrToFloat(SpinEdit1.Text) = 10 then
  begin
  DDEClientItem29.DDEItem := 'R174C2';
  DDEClientItem30.DDEItem := 'R174C3';
  DDEClientItem31.DDEItem := 'R174C4';
  DDEClientItem32.DDEItem := 'R174C5';
  DDEClientItem33.DDEItem := 'R174C6';
  end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

{{{{{{{{{{{{{{{{{{{{{Display of the Output, Input A and Input B
values for each operational stage of the optimised system{{{{{{{{{{}
hm := (k1*h1+k2*h2)/(k1+k2);
memo1.lines.add('Stage'+ IntToStr(i));
memo1.lines.add(FloatToStr(hm)+' -Output');
memo1.lines.add(FloatToStr(x_a)+' -Input A');
memo1.lines.add(FloatToStr(x_b)+' -Input B');

po := hm ;
pok := FloatToStr(po);
DDECli.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+IntToStr(28+i)))
.DDEItem,StrPCopy(spt,pok));
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;

{{{{{Logic of the buttons appearance and disappearance{{{{{{{{{{{{{}
Button2.visible := true;
Button3.visible := false;
Button1.visible := false;
Button5.visible := false;
Button4.visible := false;
Button6.visible := false;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;


procedure TForm1.ListBox1Click(Sender: TObject);
begin
{{{{{{{{{{{{{{{{{{{{{{{{{{Selection box for Input A{{{{{{{{{{{{{{{{{{{{}
```

```
 Edit3.text := ListBox1.Items.Strings[ListBox1.ItemIndex];
 res_3 := StrToFloat(Edit3.text);
 poke_3 := FloatToStr(res_3);
 DDECli.PokeData(DDEClientItem12.DDEItem, StrPCopy(sptr,poke_3));
 ListBox1.clear;
 Button2.visible:=true;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;


procedure TForm1.ListBox2Click(Sender: TObject);
begin
{{{{{{{{{{{{{{{{{{{{{{{{Selection box for Input B{{{{{{{{{{{{{{{{{{{{}
 Edit6.text := ListBox2.Items.Strings[ListBox2.ItemIndex];
 res_6 := StrToFloat(Edit6.text);
 poke_6 := FloatToStr(res_6);
 DDECli.PokeData(DDEClientItem15.DDEItem, StrPCopy(sptr,poke_6));
 ListBox2.clear;
 Button3.visible:=true;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;


procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
{{{{{{{{{{{{{{{{{Number of required simulations of Fuzzy Logic{{{{{}
 res_9 := StrToFloat(SpinEdit1.text);
 poke_9 := FloatToStr(res_9);
 DDECli.PokeData(DDEClientItem34.DDEItem, StrPCopy(sptr,poke_9));
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
Close;
end;

{{Transfer of Input A, Input B, all targets and all spaces to Excel}
procedure TForm1.Edit1Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem10.DDEItem, StrPCopy(sptr,Edit1.text));
end;

procedure TForm1.Edit2Change(Sender: TObject);
begin
```

```
DDECli.PokeData(DDEClientItem11.DDEItem, StrPCopy(sptr,Edit2.text));
end;

procedure TForm1.Edit3Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem12.DDEItem, StrPCopy(sptr,Edit3.text));
end;

procedure TForm1.Edit4Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem13.DDEItem, StrPCopy(sptr,Edit4.text));
end;

procedure TForm1.Edit5Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem14.DDEItem, StrPCopy(sptr,Edit5.text));
end;

procedure TForm1.Edit6Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem15.DDEItem, StrPCopy(sptr,Edit6.text));
end;

procedure TForm1.Edit7Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem16.DDEItem, StrPCopy(sptr,Edit7.text));
end;

procedure TForm1.Edit8Change(Sender: TObject);
begin
DDECli.PokeData(DDEClientItem17.DDEItem, StrPCopy(sptr,Edit8.text));
end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
end.
```

APPENDIX C : Co-ordination programme

```
unit Sysopt;
{
This module coordinates the optimisation procedure (i.e. connects the user with
 Dynamic Programming, Monte Carlo simulation and Fuzzy Logic modules) and displays
 the final values of effectiveness of the main and secondary subsystem.
 It also exports these value of effectiveness to FGR.xls file
}
```

```
{
General characteristic variables of the programme

G - ratios of the Effectiveness to Control Input Value Factor
z - number of rows in the 'Sub Main' table
y - number of columns in the 'Sub Second' table
Sending_S - ratios of the Effectiveness to Control Input Value Factor
        of the Secondary Subsystem
Sending_M - ratios of the Effectiveness to Control Input Value Factor
        of the Min Subsystem
i, i_f number of ratios of the Effectiveness to Control Input Value Factor
}
interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DdeMan, Buttons, StdCtrls, Grids, ToCtrl, Spin;

 type
  TForm1 = class(TForm)

    Button1: TButton;
    Edit1: TEdit;
    Label4: TLabel;
    Label3: TLabel;
    Label2: TLabel;
    Label1: TLabel;
    OleContainer1: TOleContainer;
    OleContainer2: TOleContainer;
    OleContainer3: TOleContainer;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    DdeClientItem1: TDdeClientItem;
    DdeClientConv1: TDdeClientConv;
    DdeClientConv2: TDdeClientConv;
    SpinEdit1: TSpinEdit;
    DdeClientItem21: TDdeClientItem;
    DdeClientItem22: TDdeClientItem;
    DdeClientConv3: TDdeClientConv;
    Label5: TLabel;
    DdeClientItem23: TDdeClientItem;
    DdeClientItem24: TDdeClientItem;
    DdeClientItem25: TDdeClientItem;
```

```
    Button2: TButton;
    DdeClientItem26: TDdeClientItem;
    DdeClientItem27: TDdeClientItem;
    DdeClientItem28: TDdeClientItem;
    DdeClientConv4: TDdeClientConv;
    DdeClientItem29: TDdeClientItem;
    DdeClientItem2: TDdeClientItem;
    DdeClientItem3: TDdeClientItem;
    DdeClientItem4: TDdeClientItem;
    Edit2: TEdit;
    Edit3: TEdit;
    Button3: TButton;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

  private


     { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  sptr : array[0..200] of Char;
  G : array[1..10,1..10] of String;


implementation

{$R *.DFM}

procedure TForm1.Button3Click(Sender: TObject);

Begin
{CLOSING THE PROGRAMME}
Close;

End;
```

```
procedure TForm1.Button2Click(Sender: TObject);
{THIS PROCEDURE ERASES ALL OPTIMISED VALUES OF CAPABILITIES,
DEPENDABILITIES
AND AVAILABILITIES}

var
z,y : integer;

Begin
{Erasing values of capabilities, dependabilities, availabilities from
main subsystem - gives space for new calculations of Ratio of Effectiveness
to Control Input Value Factor}

  DDEClientConv4.SetLink('excel','c:\sysopt\excelfil\Main.xls');
  DDEClientItem26.DDEItem := 'R19C2:R28C18';
  DDEClientConv4.PokeData(DDEClientItem23.DDEItem,StrPCopy(sptr,''));

  DDEClientItem27.DDEItem := 'R107C2:R116C11';
  DDEClientConv4.PokeData(DDEClientItem24.DDEItem,StrPCopy(sptr,''));

  DDEClientItem28.DDEItem := 'R165C2:R174C6';
  DDEClientConv4.PokeData(DDEClientItem25.DDEItem,StrPCopy(sptr,''));

{Clears ratios of Effectiveness to Control Input Value Factor of the Main Subsystem -
'Main Sub' table}
for z := 1 to 20 do
 for y := 0 to 20 do
  StringGrid1.Cells[y,z]:= '';

End;

procedure TForm1.FormCreate(Sender: TObject);
{THIS PROCEDURE OPENS LINK WITH FGR.XLS EXCEL FILE}

 Begin

 {opens the dialog with FGR.xls}
 DDEClientConv2.SetLink('excel','c:\sysopt\excelfil\FGR.xls');

 {dimensions of the main window}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
 Form1.Width := 648;
 Form1.Height := 484;
 Form1.Left :=0;
```

```
  Form1.top :=0;

 {Logic of buttons appearance and disappearance}
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
  Button1.Visible := false;
  OleContainer1.Visible := false;
  OleContainer2.Visible := false;
  OleContainer3.Visible := false;
  SpinEdit1.Visible := false;
  Label5.Visible := false;
 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

End;


procedure TForm1.BitBtn1Click(Sender: TObject);
{THIS PROCEDURE OPENS LINK WITH MAIN SUBSYSTEM (Main.xls)}


Begin

   Edit1.Text := 'Main';
   {Opens dialog with Main.xls}
   DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\main.xls');

  {poking name of currently considered subsystem for FGR.xls excel file to
  this module}
   DDEClientItem21.DDEItem := 'R4C2';
   DDEClientConv2.PokeData(DDEClientItem21.DDEItem,StrPCopy(sptr, Edit1.Text ));
  {Number of the ratios of Effectiveness to Control Input Value facto of the
  Main Subsystem}
   DDEClientItem3.DDEItem := 'R4C7';
   Edit2.text := DDEClientItem3.text;

  {Logic of buttons appearence and dissapearence}
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
  Button1.Visible := true;
  BitBtn1.Visible := false;
  BitBtn2.Visible := false;
  OleContainer1.Visible := true;
  OleContainer2.Visible := true;
  OleContainer3.Visible := true;
  {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

End;
```

```
procedure TForm1.BitBtn2Click(Sender: TObject);
{THIS PROCEDURE OPENS LINK WITH SECONDARY SUBSYSTEM
(Second.xls)}

begin
Edit1.Text := 'Second_1';
DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\second_1.xls');

 begin
{Number of critical paths of the main subsystem which indicate the number of
the entire secondary subsystem's optimisations }
   SpinEdit1.MinValue := 1;
   DDEClientItem29.DDEItem := 'R5C3';
   SpinEdit1.MaxValue := 1+Round(StrToFloat(DDEClientItem29.Text));
   If (SpinEdit1.MaxValue-SpinEdit1.MinValue)= 0 then
   SpinEdit1.Enabled := false;
 end;

 {Poking name of currently considered subsystem}
 DDEClientItem21.DDEItem := 'R4C2';
 DDEClientConv2.PokeData(DDEClientItem21.DDEItem,StrPCopy(sptr, Edit1.Text ));
 {Number of the ratios of Effectiveness to Control Input Value facto of the
 Secondary Subsystem}
 DDEClientItem4.DDEItem := 'R5C7';
 Edit3.text := DDEClientItem4.text;

 {Logic of buttons appearance and disappearance}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
 Button1.Visible := true;
 SpinEdit1.Visible := true;
 Label5.Visible := true;
 BitBtn1.Visible := false;
 BitBtn2.Visible := false;
 OleContainer1.Visible := true;
 OleContainer2.Visible := true;
OleContainer3.Visible := true;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

End;

procedure TForm1.Button1Click(Sender: TObject);
{IMPORTS VALUES OF EFFECTIVENESS FORM  EXCEL FILES (i.e. Main.xls or
Secod1.xls)
(the effectiveness value are displayed in StringGrid1 and StrinGrid2)
```

```
}
type

{use of pointer for naming cells in Excel{{{{{{{{{{{{{{{{{{{}
 string10 = string[5];
 Str = ^string10;
 send = array [1..10] of str;

{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}

var
 i, op, opt, i_f: integer  ;
  Sending_S,Sending_M : send;
  FieldNames : array[1..10] of string;

  {selection of the subsystem for optimisation}
Begin
     If  Edit1.Text = 'Main' Then
      begin
      DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\main.xls');
      i_f := round(StrToFloat(DDEClientItem3.text));
      end
     Else if Edit1.Text = 'Second_1' Then
      begin
      DDEClientConv1.SetLink('excel','c:\sysopt\excelfil\second_1.xls');
      i_f := round(StrToFloat(DDEClientItem4.text));
      op := Round(StrToFloat(SpinEdit1.Text));
      {indicates number of critical path from main subsystem}
      opt := i_f*(op-1);
      end;


 {{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
{Picks ratios of Effectiveness to Control Input Value Factors from
Cells A275 to M275}
     for i := 1 to i_f do
      begin
      FieldNames[i] := 'R275'+'C'+IntToStr(3+i);
{The exchange of data with the Main Subsystem}
         If  Edit1.Text = 'Main' Then
          begin
          New (Sending_M[i]);
          Sending_M[i]^:= 'R12'+'C'+IntToStr(2+i);
          DDEClientItem1.DDEConv := DDEClientConv1;
          DDEClientItem1.DDEItem := FieldNames[i];
```

```
            StringGrid1.Cells[1,i] :=DDEClientItem1.Text;
            G[1,i] := (StringGrid1.Cells[1,i]);
            {Headings of the Main Subsystem's table}
            StringGrid1.Cells[0,0]:= 'Number of Options';
            StringGrid1.Cells[0,i]:= 'MainSub'+IntToStr(i);

 {Exporting  ratios of Effectiveness to Control Input Value Factors from the
 'Main Sub' table of this module to FGR.xls Excel file}
            DDEClientItem2.DDEConv := DDEClientConv2;
            DDEClientItem2.DDEItem := Sending_M[i]^;

DDEClientConv2.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
            IntToStr(2))).DDEItem,StrPCopy(sptr,G[1,i]));
            Dispose(Sending_M[i]);
            end

 {The exchange of data between the Secondary Subsystem}
          Else if Edit1.Text = 'Second_1' Then
            begin
            New (Sending_S[i]);  {pointer }
            Sending_S[i]^:= 'R'+IntToStr(12+i)+'C'+IntToStr(2+op);
            {Exchanging values from excel}
            DDEClientItem1.DDEConv := DDEClientConv1;
            DDEClientItem1.DDEItem := FieldNames[i];
            StringGrid2.Cells[1,i] :=DDEClientItem1.Text;
            G[1,i] := (StringGrid2.Cells[1,i]);

            StringGrid2.Cells[1,0]:= 'MainSub'+ SpinEdit1.Text;
            StringGrid2.Cells[0,0]:= 'Number of Options';
            StringGrid2.Cells[0,i]:= 'SecondSub'+ IntToStr(i);

 {Poking values to final.xls}
            DDEClientItem2.DDEConv := DDEClientConv2;
            DDEClientItem2.DDEItem := Sending_S[i]^;

DDEClientConv2.PokeData(TDDEClientItem(FindComponent('DDEClientItem'+
            IntToStr(2))).DDEItem,StrPCopy(sptr,G[1,i]));
            Dispose(Sending_S[i]);
            end;
        end;
{Logic of buttons appearance and disappearance}
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{[}
  Button1.Visible := false;
  BitBtn1.Visible := false;
  OleContainer1.Visible := false;
```

```
  OleContainer2.Visible := false;
  OleContainer3.Visible := false;
  If Edit1.Text = 'Main' then
  begin
  BitBtn1.Visible := true;
  BitBtn2.Visible := true;
  end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{[}

End;

procedure TForm1.SpinEdit1Change(Sender: TObject);
{DISPLAYS NUMBER OF CONSIDERED MAIN SUBSYSTEM'S PATHS}

var
z,y : integer ;

Begin

   DDEClientConv3.SetLink('excel','c:\sysopt\excelfil\Second_1.xls');
   DDEClientItem22.DDEItem := 'R6C2';

DDEClientConv3.PokeData(DDEClientItem22.DDEItem,StrPCopy(sptr,SpinEdit1.Text))
;

 {Logic of buttons and windows appearance and disappearance }
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
   OleContainer1.Visible := True;
   OleContainer2.Visible := True;
   OleContainer3.Visible := True;
   Button1.Visible := True;

   If SpinEdit1.Text > DDEClientItem29.Text then
     begin
     BitBtn1.Visible := True;
     BitBtn2.Visible := True;
     OleContainer1.Visible := False;
     OleContainer2.Visible := False;
     OleContainer3.Visible := False;
     SpinEdit1.Text := '1';
     SpinEdit1.Visible := False;
     Label5.Visible := False;
     end;
{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{}
```

```
{Erasing old values of capabilities, dependebilities, availabilities from spreadsheet}
{ of the Secondary Subsystem to give space for new calculations}
  If Edit1.Text <> 'Main  ' Then
    begin
    DDEClientItem23.DDEItem := 'R19C2:R28C18';
    DDEClientConv1.PokeData(DDEClientItem23.DDEItem,StrPCopy(sptr,''));
    DDEClientItem24.DDEItem := 'R107C2:R116C11';
    DDEClientConv1.PokeData(DDEClientItem24.DDEItem,StrPCopy(sptr,''));
    DDEClientItem25.DDEItem := 'R165C2:R174C6';
    DDEClientConv1.PokeData(DDEClientItem25.DDEItem,StrPCopy(sptr,''));

{Clears 'Second Sub' table in this module}
       for z := 1 to 20 do
         for y := 0 to 20 do
           StringGrid2.Cells[y,z]:= '                ';
    end;

End;
{End of the Sysopt.pas module}
End.
End;
{End of the Sysopt.pas module}
End.
```

APPENDIX D : Results of Present Value calculations

CALCULATIONS OF INTERESTS USING PRESENT VALUE FACTOR

TOTAL REPAYMENTS FOR OPTION 1

NUMBER OF MONTHS REQUIRED TO FOR REPAYMENTS	PROPOSAL1 For investments *R1000			PROPOSAL2 For investments *R1000		
	20	30	25	20	30	25
1	20	15	25	20	30	25
2	10	15	12.5	12	20	19
3				8	14	9
4				4	8	5
5					4	5
6					4	

CALCULATED VALUES OF INTERESTS FOR OPTION1

	6%	8%	7%	6%	8%	7%
	STAGE1	STAGE2	STAGE3	STAGE1	STAGE2	STAGE3
1	($3.27)	($1.85)	($3.52)	($3.33)	($3.75)	($3.57)
2	($1.63)	($1.85)	($1.76)	($2.00)	($2.50)	($2.71)
3				($1.33)	($1.75)	($1.29)
4				($0.67)	($1.00)	($0.71)
5					($0.50)	($0.71)
6					($0.50)	
Total Value of Interest	($4.90)	($3.70)	($5.27)	($7.33)	($10.00)	($9.00)

TOTAL REPAYMENTS FOR OPTION 2

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	PROPOSAL1 For investments *R1000			PROPOSAL2 For investments *R1000		
	10	20	20	10	20	20
1	10	20	20	10	20	20
2		10	10	6	15	15
3					9	9
4					3	5
5					2	
6						

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	CALCULATED VALUES OF INTERESTS FOR OPTION2					
	6%	8%	7%	6%	8%	7%
	STAGE1	STAGE2	STAGE3	STAGE1	STAGE2	STAGE3
1	($1.43)	($2.47)	($2.81)	($1.63)	($2.50)	($2.86)
2		($1.23)	($1.41)	($0.98)	($1.87)	($2.14)
3					($1.12)	($1.29)
4					($0.37)	($0.71)
5					($0.25)	
6						
Total Value of Interest	($1.43)	($3.70)	($4.22)	($2.61)	($6.12)	($7.00)

TOTAL REPAYMENTS FOR OPTION 3

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	PROPOSAL1 For investments *R1000			PROPOSAL2 For investments *R1000		
	5	10	5	5	10	5
1	5	10	5	5	10	5
2				2	5	2
3						
4						
5						
6						

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	CALCULATED VALUES OF INTERESTS FOR OPTION3					
	7%	9%	10%	7%	9%	10%
	STAGE1	STAGE2	STAGE3	STAGE1	STAGE2	STAGE3
1	($0.63)	($1.00)	($0.45)	($0.70)	($1.10)	($0.50)
2				($0.28)	($0.55)	($0.20)
3						
4						
5						
6						
Total Value of Interest	($0.63)	($1.00)	($0.45)	($0.98)	($1.65)	($0.69)

TOTAL REPAYMENTS FOR OPTION 4

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	PROPOSAL1 For investments *R1000			PROPOSAL2 For investments *R1000		
	15	25	20	15	25	20
1	15	25	20	15	25	20
2	7.5	12.5	10	7	23	18
3				3	15	10
4					8	4
5					4	
6						

NUMBER OF MONTHS REQUIRED FOR REPAYMENTS	CALCULATED VALUES OF INTERESTS FOR OPTION4					
	6% STAGE1	8% STAGE2	7% STAGE3	6% STAGE1	8% STAGE2	7% STAGE3
1	($2.45)	($3.09)	($2.81)	($2.49)	($3.12)	($2.86)
2	($1.22)	($1.54)	($1.41)	($1.16)	($2.87)	($2.57)
3				($0.50)	($1.87)	($1.43)
4					($1.00)	($0.57)
5					($0.50)	
6						
Total Value of Interest	($3.67)	($4.63)	($4.22)	($4.15)	($9.37)	($7.43)

.PPENDIX E : Data used for the optimisation of the detergent manufacturing system

nput values for Dynamic Programming

1ain Subsystem

TABLE OF MAIN SUBSYSTEM'S PERFORMANCES

'roposal	STAGE1		STAGE2		STAGE3		STAGE4	
	*Investment 1 * R1000*	*Revenue 1 minutes*	*Investment 2 * R1000*	*Revenue 2 minutes*	*Investment3 * R1000*	*Revenue3 minutes*	*Revenue 4 * R1000*	*Investment 5*
1	20	10	30	20	25	30	0	
2	10	20	20	60	20	70	0	
3	5	40	10	120	5	150	0	
4	15	15	25	80	20	50	0	
5								

'econdary Subsystem

TABLE OF SECONDARY SUBSYSTEM'S PERFORMANCES

)ptions for dynamic -rogramming	Stage1		Stage2		Stage3		Stage4	
	Inve1 months	Rev1 *R1000	Inve 2 months	Rev2 *R1000	Inve 3 months	Rev3 *R1000	Inve 4 months	Rev4 *R1000
/Prop1	2	2.45	2	2.25	2	2.63		
/Prop2	5	1.37	6	1.56	5	1.7		
'/Prop1	1	1.33	2	2.32	2	2.11		
'/Prop2	2	1.04	5	1.12	4	1.85		
/Prop1	1	0.63	2	0.3	1	0.45		
/Prop2	2	0.39	1	1.65	2	0.24		
/Prop1	2	1.83	2	2.31	4	1.055		
/Prop2	3	1.28	5	1.674	2	3.71		
/Prop1								
/Prop2								

.PPENDIX E (continued)

nput Values for Monte Carlo Simulation

Iain Subsystem

Table of Proposed Transition Rates

)ependent ransition rates	R_14				R_24				R_34			
	Stage1				Stage2				Stage3			
roposal1	R_11	R_13	R_12	R_15	R_21	R_22	R_23	R_25	R_33	R_32	R_31	R_35
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
			0.0012				0.0015				0.0014	
	Stage1				Stage2				Stage3			
roposal2	R_11	R_13	R_12	R_15	R_21	R_22	R_23	R_25	R_33	R_32	R_31	R_35
			0.002				0.0022				0.0025	
			0.0019				0.0021				0.0026	
			0.0018				0.0019				0.0027	
			0.00195				0.0023				0.0026	
			0.002				0.0018				0.0024	
			0.00175				0.00185				0.0025	
			0.0018				0.0022				0.0026	
			0.00175				0.0023				0.0024	
			0.00195				0.002				0.0024	
			0.002				0.002				0.0023	
	Stage1				Stage2				Stage3			
roposal3	R_11	R_13	R_12	R_15	R_21	R_22	R_23	R_25	R_33	R_32	R_31	R_35
			0.008				0.0071				0.0093	
			0.0085				0.0073				0.0096 7	
			0.008				0.00722				0.0093 5	
			0.0088				0.00728				0.0094	
			0.00875				0.0072				0.0095 6	
			0.0088				0.0071				0.0095 8	

			0.0089				0.00715				0.00956	
			0.00885				0.00725				0.00939	
			0.0083				0.00716				0.00942	
			0.0082				0.0073				0.00954	
	Stage1				Stage2				Stage3			
Proposal4	R_11	R_13	R_12	R_15	R_21	R_22	R_23	R_25	R_33	R_32	R_31	R_35
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	
			0.001				0.0012				0.0012	

Secondary Subsystem

Dependent Transition rates		R_14				R_24				R_34		
Proposal1	Stage1				Stage2				Stage3			
	R_12	R_13	R_11	R_15	R_21	R_23	R_22	R_25	R_31	R_32	R_33	R_35
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
			0.0005				0.0004				0.00045	
	Stage1				Stage2				Stage3			
Proposal2	R_12	R_13	R_14	R_15	R_21	R_23	R_24	R_25	R_31	R_32	R_34	R_35
			0.000641				0.00069				0.000648	
			0.00062				0.000696				0.00065	
			0.00062				0.000697				0.00065	
			0.00061				0.0007				0.000645	
			0.000624				0.000691				0.000643	
			0.000634				0.000694				0.000643	
			0.00064				0.000694				0.000644	
			0.000626				0.000693				0.000648	
			0.000635				0.000699				0.00065	

			0.000637				0.000699				0.00065	
	Stage1				Stage2				Stage3			
Proposal3	R_12	R_13	R_14	R_15	R_21	R_23	R_24	R_25	R_31	R_32	R_34	R_35
			0.000941				0.00092				0.000948	
			0.001				0.00093				0.00095	
			0.00097				0.00091				0.000095	
			0.000971				0.0009				0.000945	
			0.000964				0.00091				0.000943	
			0.000954				0.00094				0.000943	
			0.00099				0.00094				0.000944	
			0.00092				0.00093				0.000948	
			0.000935				0.00099				0.00095	
			0.000937				0.00099				0.00095	
	Stage1				Stage2				Stage3			
Proposal4	R_12	R_13	R_14	R_15	R_21	R_23	R_24	R_25	R_31	R_32	R_34	R_35
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	
			0.00042				0.0004				0.00042	

APPENDIX E (continued)

nput values for Fuzzy Logic

Main Subsystem

TABLE OF PROPOSED INPUTS

NPUT A — INPUT DATA DEPENDENT ON DEPENDENT ON CAPABILITIES

	STAGE1	STAGE2	STAGE3	STAGE4	STAGE5
PROPOSAL1	5	5	5		
PROPOSAL2	3	4	3		
PROPOSAL3	1	2	1		
PROPOSAL4	4	4	4		
PROPOSAL5					

INPUT B — INPUT DATA DEPENDENT ON DEPENDENT ON CAPABILITIES

	STAGE1	STAGE2	STAGE3	STAGE4	STA
PROPOSAL1	8	9	9		
PROPOSAL2	3	6	5		
PROPOSAL3	1	3	1		
PROPOSAL4	7	7	7		
PROPOSAL5					

'econdary Subsystem

	PROPOSED INPUT VALUES				
	Subjective quality improvement INPUT DATA DEPENDENT ON CAPABILITIES' PROPOSALS INPUT A				
	STAGE1	STAGE2	STAGE3	STAGE4	STAGE5
PROPOSAL1	4	5	4		
PROPOSAL2	3	4	3		
PROPOSAL3	1	2	1		
PROPOSAL4	4	4	4		
PROPOSAL5					

	% introduction of new products INPUT DATA DEPENDENT ON CAPABILITIE PROPOSALS INPUT B				
	STAGE1	STAGE2	STAGE3	STAGE4	STA
PROPOSAL1	50	50	50		
PROPOSAL2	35	35	35		
PROPOSAL3	5	5	5		
PROPOSAL4	40	40	40		
PROPOSAL5					