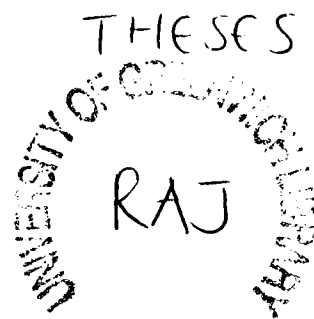


THE DEVELOPMENT OF A STRUCTURED METHODOLOGY FOR THE CONSTRUCTION AND INTEGRITY CONTROL OF SPREADSHEET MODELS

KAMALASEN RAJALINGHAM

A thesis submitted in partial fulfilment of the
requirements of the University of Greenwich
for the degree of Doctor of Philosophy

December 2002



Abstract

Numerous studies and reported cases have established the seriousness of the frequency and impact of user-generated spreadsheet errors. This thesis presents a structured methodology for spreadsheet model development, which enables improved integrity control of the models. The proposed methodology has the potential to ensure consistency in the development process and produce more comprehensible, reliable and maintainable models, which can reduce the occurrence of user-generated errors.

An insight into the nature and properties of spreadsheet errors is essential for the development of a methodology for controlling the integrity of spreadsheet models. An important by-product of the research is the development of a comprehensive classification or taxonomy of the different types of user-generated spreadsheet errors based on a rational taxonomic scheme.

Research on the phenomenon of spreadsheet errors has revealed the need to adopt a software engineering based methodology as a framework for spreadsheet development in practical situations. The proposed methodology represents a new approach to the provision of a structured, software engineering based discipline for the development of spreadsheet models.

It is established in this thesis that software engineering principles can in fact be applied to the process of spreadsheet model building to help improve the quality of the models. The methodology uses Jackson structures to produce the logical design of the spreadsheet model. This is followed by a technique to derive the physical model, which is then implemented as a spreadsheet. The methodology's potential for improving the quality of spreadsheet models is demonstrated.

In order to evaluate the effectiveness of the proposed framework, the various features of the proposed structured methodology are tested on a range of spreadsheet models through a series of experiments. The results of the tests provide adequate evidence of the methodology's potential to reduce the occurrence of user-generated errors and enhance the comprehensibility of the models.

Acknowledgements

I would like to express my heartfelt thanks and appreciation, first and foremost, to the members of my supervisory team in the School of Computing and Mathematical Sciences of the University of Greenwich (England). They are Professor Brian Knight, Mr David Chadwick and Dr Dilwyn Edwards.

I am also grateful to the following individuals for their support, guidance, advice and co-operation in the research programme:

- Mr Mike Shallcross, former Principal Consultant, Financial Modelling Department, KPMG Management Consulting (London)
- Mr Chris Conlong, Principal Consultant, Financial Modelling Department, KPMG Management Consulting (London)
- Mr David Colver, Joint Chief Executive, Operis Group plc (London)
- Dr Nadarajah Ramesh, Senior Lecturer (Statistics), School of Computing and Mathematical Sciences, University of Greenwich (England)
- Mr Patrick Lees, Chair of the Information Systems Department, Cavendish School of Computer Science, University of Westminster (London)
- Dr Dusko Dincov, Lecturer, School of Computing and Mathematical Sciences, University of Greenwich (England)
- Mr Chris Rodger, Principal Consultant, Business Dynamics Consultancy Group, PricewaterhouseCoopers (London)
- Dr Anthony Berglas (Developer of *Spreadsheet Detective*), Southern Cross Software Qld (Australia)
- Mr Ray Butler, Chief Technical Adviser to the Joint Head of Computer Audit Services, HM Customs & Excise (United Kingdom) & Chairman of the North England ISACA Chapter
- Professor Raymond Panko, Professor of Business Administration, University of Hawai'i (Honolulu)

I would also like to thank my parents for their constant support and encouragement.

Keywords

Spreadsheet, Spreadsheet Model, Error, Methodology, Taxonomy, Software Engineering, Modularisation, Jackson Structured Programming (JSP), Integrity Control, Structured Methods, Structured Spreadsheet Development

TABLE OF CONTENTS

ABSTRACT	1
ACKNOWLEDGEMENTS	2
KEYWORDS	3
TABLE OF CONTENTS	4
CHAPTER 1	
INTRODUCTION	8
1.1 OVERVIEW	8
1.2 CONTRIBUTIONS OF THE THESIS	9
1.3 OUTLINE OF THE THESIS	10
1.4 TIME-LINE OF DEVELOPMENTS	11
1.5 RESEARCH APPROACHES	14
CHAPTER 2	
BACKGROUND	15
2.1 INTRODUCTION	15
2.2 AN INSIGHT INTO SPREADSHEETS	15
2.3 THE PHENOMENON OF SPREADSHEET ERRORS	16
2.4 NEED FOR A DISCIPLINED AND STRUCTURED APPROACH	21
2.5 SUMMARY	23
CHAPTER 3	
ANALYSIS AND CLASSIFICATION OF SPREADSHEET ERRORS	24
3.1 INTRODUCTION	24
3.2 THE CONCEPT OF <i>TAXONOMY</i> OR CLASSIFICATION	24
3.3 RATIONALE FOR THE CLASSIFICATION OF THE ERRORS	25
3.4 DERIVATION OF THE TAXONOMIC SCHEME	25
3.5 THE CLASSIFICATION OF ERRORS	27
3.6 SUMMARY	50

CHAPTER 4

PAST WORK AND EXISTING DEVELOPMENTS 52

4.1 INTRODUCTION 52

4.2 EXISTING TOOLS AND TECHNIQUES 52

4.3 SPREADSHEET DEVELOPMENT LIFE CYCLE
AND METHODOLOGY 62

4.4 SUMMARY 70

CHAPTER 5

PRELIMINARY INVESTIGATION AND DEVELOPMENTS 72

5.1 INTRODUCTION 72

5.2 ANALYSIS OF SPREADSHEET STRUCTURE 72

5.3 INITIAL APPROACHES EXPLORED AND DEVELOPED 75

5.4 SUMMARY 82

CHAPTER 6

**SOFTWARE ENGINEERING PRINCIPLES
AND JACKSON STRUCTURES 83**

6.1 INTRODUCTION 83

6.2 SOFTWARE ENGINEERING PRINCIPLES 83

6.3 RATIONALE FOR SELECTION OF JACKSON STRUCTURES 85

6.4 CONCEPTS AND NOTATIONS OF JACKSON STRUCTURES 86

6.5 JACKSON STRUCTURE RULES 90

6.6 OTHER PRINCIPLES AND TECHNIQUES 91

6.7 SUMMARY 93

CHAPTER 7

PROPOSED STRUCTURED METHODOLOGY 94

7.1 INTRODUCTION 94

7.2 DEVELOPMENT AND SYNTHESIS OF THE
PROPOSED METHODOLOGY 94

7.3 THE PROPOSED STRUCTURED METHODOLOGY 103

7.4 APPLICATION OF THE PROPOSED METHODOLOGY 116

7.5 POTENTIAL FOR QUALITY IMPROVEMENT 120

7.6 SUMMARY 122

CHAPTER 8

EVALUATION STRATEGY AND EXPERIMENTS 124

8.1 INTRODUCTION 124

8.2 THE EVALUATION STRATEGIES 126

8.3 THE EXPERIMENTS UNDERTAKEN 127

8.4 SUMMARY 139

CHAPTER 9

ANALYSIS OF RESULTS 142

9.1 INTRODUCTION 142

9.2 RESULTS OF EXPERIMENTS 142

9.3 SUMMARY 157

CHAPTER 10

CONCLUSIONS AND FUTURE WORK 158

10.1 CONCLUSIONS 158

10.2 FUTURE WORK 160

REFERENCES161

APPENDIX A

FREQUENCY OF SPREADSHEET ERRORS (Panko-96,98,00) 173

APPENDIX B

**DESCRIPTIONS OF CORPORATE/WORKGROUP
POLICIES (Panko-96) 176**

APPENDIX C

EXAMPLES OF THE PROPOSED METHODOLOGY 178

APPENDIX D

SPREADSHEET MODELS USED IN THE EXPERIMENTS 206

APPENDIX E

NORMALITY TEST RESULTS (GRAPHS) 231

APPENDIX F

RAW DATA FROM THE EXPERIMENTS 240

CHAPTER 1

INTRODUCTION

1.1 Overview

Over the years, spreadsheet users in business and academia have been completely taken aback by the appalling rates of user-generated errors occurring in spreadsheet models. Many publications have also described the adverse effect this phenomenon has had on businesses and other organisations. As a result, many groups of users and individuals from the commercial and non-commercial sectors have endeavoured to find solutions to the problem.

Despite all the efforts, the problem has been prevalent. The initial efforts to establish the magnitude of the problem of spreadsheet errors were based on measuring error rates and studying the impact of the errors on organisations. All the experiments and studies that were carried out proved beyond any doubt that this was indeed a very critical problem (Brown-87, Davies-87, Panko-96, Freeman-96, Ward-97) which had to be addressed urgently. However, there was very limited material available on specific types of spreadsheet errors. Therefore, far more extensive research had to be undertaken to identify, analyse and classify specific types of spreadsheet errors.

The focus of authors on the subject of spreadsheet model integrity subsequently turned towards ascertaining the cause of and reasons for the occurrence of user-generated spreadsheet errors. Many who carefully analysed the problem were able to conclude that the principal cause of these errors was the absence of standards for or a structured approach to designing and developing spreadsheet models (Ronen-89, Cragg-93, Isakowitz-95, Hall-96). Such standards and structured methods have however been adopted in other areas of software and systems development such as programming and database development. Authors responded to this discovery by recommending the adoption of software engineering principles and traditional programming techniques in the process of spreadsheet model building (Benham-93, Isakowitz-95, Panko-96, Davis-96, Kavanagh-97). However, none came up with a comprehensive methodology based on this requirement.

In general, two distinct approaches can be adopted to address the problem of user-generated spreadsheet errors. The first approach involves developing tools and methods to help identify errors in spreadsheet models so that they can be corrected. In a desperate pursuit for immediate solutions, this is the approach taken by most organisations at present, as a large number of existing spreadsheet models are already infested with errors. The second approach concentrates on preventing the errors from occurring in the first place. In order to achieve this, an effective methodology for controlling the integrity of spreadsheet models has to be developed and applied.

The principal objective of this research programme is to examine the possibility of developing a structured methodology for the quality or integrity control of spreadsheet models. This involves an investigation into the application of software engineering principles and techniques to the process of spreadsheet design and development.

1.2 Contributions of the Thesis

Basic Questions Posed at the Outset of this Work

These are the questions which the work described in this thesis has been directed towards:

Primary Question

- *Can a structured methodology be developed for the integrity control of spreadsheet models? Can such a framework for quality control of spreadsheet models reduce the occurrence of user-generated errors?*

Secondary Questions

- *Can a classification of the different types of user-generated spreadsheet errors be developed based on a rational taxonomic scheme?*
- *What framework for spreadsheet model development is most likely to be optimum in a practical situation?*
- *How effective is the framework?*
- *Can software engineering principles be applied to the process of spreadsheet model building to help improve the quality of the models?*

Contributions

Several contributions are made in this thesis. Firstly, a diverse collection of user-generated spreadsheet errors have been collected, analysed and categorised according to a rational taxonomic scheme. The provision of this comprehensive classification of the different types of spreadsheet errors is based on an analysis of the properties of user-generated errors. The errors are accumulated from numerous spreadsheet models. The spreadsheet error taxonomy is described in detail and supported by appropriate examples.

Secondly, structured techniques and principles have been proposed as the basis of a novel spreadsheet development methodology. The proposed structured methodology fundamentally adopts a software engineering approach and is based on established principles of structured analysis, design and development. It presents a systematic algorithm, consisting of a set of coherent stages addressing the analysis, design and development of spreadsheet models.

The main principle underpinning the proposed methodology has been derived from data structure diagrams akin to those proposed by Jackson in Jackson Structured Programming (Jackson-75, Ingevaldsson-86, Burgess-87). The methodology provides models in a structured form, allowing indentation and all its concomitant advantages in terms of comprehension and maintenance. It therefore enables improved integrity control of spreadsheet models, facilitating understanding and interpretation of the models in a standard and unambiguous manner. As a result of the structure and strict

discipline introduced in the process of spreadsheet building, the occurrence of user-generated errors can be reduced.

In addition to that, the methodology has been tested on a wide spectrum of spreadsheet errors for its effect on quality. The various features of the proposed methodology are also tested on a wide range of spreadsheet models and users in academia. The results have demonstrated that the methodology is indeed effective in producing spreadsheet models that are more comprehensible and less prone to user-generated errors. In conclusion, the research has contributed significantly to the provision of additional knowledge and novel methods to the area of integrity control of spreadsheet models.

1.3 Outline of the Thesis

Chapter 1 of this dissertation provides an overview, and outlines the contributions made in this research programme. The chapter also includes a development story, time-line and chronology of publications produced. The last section gives details of the research approaches considered and adopted.

Chapter 2 presents an insight into spreadsheets and describes the problem of user-generated spreadsheet errors in terms of their frequency and impact. It also distinctly establishes the magnitude of the phenomenon of spreadsheet errors. At the end of the chapter, the importance of applying software engineering and structured methods to spreadsheet development is discussed.

Chapter 3 presents a framework for classifying user-generated spreadsheet errors based on a rational taxonomic scheme. The spreadsheet error taxonomy is produced by analysing the nature and characteristics of the different types and categories of errors. The various types of spreadsheet errors are described and appropriate examples are given.

Chapter 4 presents a review of a spectrum of existing tools and techniques for controlling the integrity of spreadsheet models. An analysis of the effectiveness and limitations of these techniques and methods is also carried out. Various life cycles and methodologies proposed for the development of spreadsheet models are also critically explored.

Chapter 5 presents findings of a preliminary investigation carried out into various methods and approaches that are deemed to have some potential in improving the quality of spreadsheet models. The core of this chapter is presented in the third section, *Section 5.3*. The second section concerns an analysis of spreadsheet structure.

Chapter 6 conveys an insight into related software engineering concepts and principles, especially *Jackson Structures*. This is in view of the fact that the main techniques and principles of the proposed structured methodology are derived from these methods and techniques. The discussion primarily covers the rationale for the selection of Jackson structural forms and the concepts, notations and rules of Jackson structures. This is followed by a discussion of the other relevant software engineering principles and their application to spreadsheets.

Chapter 7 presents the proposed structured methodology for the development and integrity control of spreadsheet models. It focuses on the synthesis of a framework or methodology based on the established software engineering principles and structured techniques described in *Chapter 6*. The various stages of the methodology are described in detail and supported by suitable examples. The methodology's potential for enhancing the quality of spreadsheet models is also addressed.

Chapter 8 begins by putting forth a plan for the evaluation of the proposed methodology based on experimental trials. The evaluation strategies underpinning the experiments are also discussed. The actual experiments conducted are subsequently described in detail. The experiments are aimed at testing the various features of the proposed structured methodology. The series of experiments involve a range of spreadsheet models used in educational institutions and industry. The elements of the methodology are tested on diverse groups of students.

Chapter 9 presents a detailed analysis of the results of the experiments conducted. The experiments are described in *Chapter 8*.

Chapter 10 presents the conclusions drawn based on the results of the experiments and from the doctoral research programme as a whole. The principal contributions of the research project are presented. The degree to which the objectives of the research have been achieved is also established. At the end of the chapter, future work to be undertaken is proposed.

1.4 Time-line of Developments

At the outset of the programme, the primary and secondary research questions to be answered were determined and specified (*Chapter 1*).

The research began with an investigation of the evolution and functions of spreadsheets, and the problem of user-generated spreadsheet errors in terms of their frequency and impact. The views and recommendations of authors and researchers on the subject were considered to identify possible causes and potential approaches to solving the problem (*Chapter 2*).

Having established the frequency and impact of spreadsheet errors, efforts were concentrated on two sets of activities that were undertaken in parallel. These activities were as follows:

- The examination and classification of specific types of user-generated spreadsheet errors based on a rational taxonomic scheme (*Chapter 3*). This was carried out so that the effects of improvements in methodology could be studied with regard to error types.
- A review of existing tools and techniques for controlling the integrity of spreadsheet models and the different life cycles and methodologies proposed for their development (*Chapter 4*).

Upon completion of these activities, an investigation was carried out into various methods and approaches that were deemed capable of improving the quality of

spreadsheet models. The development of these initial methods and approaches was preceded by an analysis of spreadsheet structure (*Chapter 5*).

The next activity in the research programme was an elaborate examination of relevant software engineering methods and structured techniques, and their potential application to the design and development of spreadsheet models. The principal method focused upon was the use of *Jackson Structures* due to its capacity to model data dependencies, relative simplicity and likely acceptance in the spreadsheet community (*Chapter 6*).

Based on the software engineering methods and techniques investigated, a comprehensive structured methodology for the construction and integrity control of spreadsheet models was developed. Various spreadsheet models were used to assess the quality and effectiveness of the methodology (*Chapter 7*).

After the development of the proposed structured methodology, a plan was created for the evaluation of the methodology based on experimental trials. Various factors such as evaluation strategies, subjects, test models and other constraints were carefully taken into account. Following the development of the plan, the experiments were carried out accordingly (*Chapter 8*).

Various techniques and methods were subsequently employed to meticulously analyse the results of the experiments. Appropriate conclusions were drawn based on the results (*Chapter 9*).

Finally, overall conclusions were drawn based on the entire research programme, and appropriate recommendations were made pertaining to future work that can be undertaken (*Chapter 10*).

The following is a chronology of publications produced during the course of the research programme:

- **Rajalingham, K.** and Chadwick, D. (1998) "Integrity control of spreadsheets: organisation & tools". In: Jajodia, S., List, W., McGregor, G.W. and Strous, L. (eds) (1998) *Integrity and internal control in information systems*. Massachusetts: Kluwer Academic Publishers, pp. 147-168.
- **Rajalingham, K.**, Chadwick, D., Knight, B. and Edwards, D. (1999) "An approach to improving the quality of spreadsheet models". In: Hawkins, C., King, G., Ross, M. and Staples, G. (eds) (1999) *Software quality management VII – managing quality*. Great Britain: British Computer Society, pp. 117-131.
- Chadwick, D., **Rajalingham, K.**, Knight, B. and Edwards, D. (1999) "A methodology for spreadsheet development based on data structure", *CMS Press*, 99/IM/50.

- Chadwick, D., **Rajalingham, K.**, Knight, B. and Edwards, D. (1999) “An approach to the teaching of spreadsheets using software engineering concepts”, *Proceedings of the Fourth International Conference on Software Process Improvement, Research, Education and Training, INSPIRE'99, 9-11 September 1999, Crete, Greece*. Great Britain: British Computer Society, pp. 261-273.
- **Rajalingham, K.**, Chadwick, D., Knight, B. and Edwards, D. (1999) “Efficient methods for checking integrity: an integrated spreadsheet engineering methodology (ISEM)”. In: van Biene-Hershey, M.E. and Strous, L. (eds) (1999) *Integrity and internal control in information systems – strategic views on the need for control*. Massachusetts: Kluwer Academic Publishers, pp. 41-58.
- **Rajalingham, K.**, Chadwick, D., Knight, B. and Edwards, D. (2000) “Quality control in spreadsheets: a software engineering-based approach to spreadsheet development”. In: Sprague, R.H., Jr. (ed.) (2000) *Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences 2000 – abstracts and CD-ROM of full papers*. California: IEEE Computer Society.
- Chadwick, D., Knight, B. and **Rajalingham, K.** (2000) “Quality control in spreadsheets: a visual approach using color codings to reduce errors in formulae”, *Software Quality Journal*, 9(2), pp. 133-143.
- Knight, B., Chadwick, D. and **Rajalingham, K.** (2000) “A structured methodology for spreadsheet modelling”. In: Chadwick, D. (ed.) (2000) *EuSpRIG 2000 Symposium proceedings - spreadsheet risks, audit and development methods*. London: University of Greenwich, pp. 43-50.
- **Rajalingham, K.**, Chadwick, D. and Knight, B. (2000) “Classification of spreadsheet errors”, *British Computer Society (BCS) Computer Audit Specialist Group (CASG) Journal*, 10(4), pp. 5-10.
- **Rajalingham, K.**, Chadwick, D. and Knight, B. (2001) “An evaluation of the quality of a structured spreadsheet development methodology”. In: Chadwick, D. and Strous, L. (eds) (2001) *Controlling the subversive spreadsheet – risks, audit and development methods*. The Netherlands: EuSpRIG, pp. 39-59.
- **Rajalingham, K.**, Chadwick, D. and Knight, B. (2002) “Efficient methods for checking integrity: a structured spreadsheet engineering methodology”, *Informatica: An International Journal of Computing and Informatics*, 26(1).

1.5 Research Approaches

The research methodology adopted for this work may be summarised by the following steps:

1. Obtain an understanding of the problem domain.
2. Make a comprehensive study of what had already been done by others.
3. Synthesise possible high-level solutions.
4. Select the most promising high level solution.
5. Elaborate the chosen solution.
6. Test the efficacy of the solution.

To obtain an understanding of the problem domain, several resources were used. These were published literature, interviews with modellers and attendances at a spreadsheet modelling training course and spreadsheet conferences. Among the main conferences were the *IFIP TC11 WG11.5 Working Conferences on Integrity and Internal Control in Information Systems*, *International Conferences on Software Quality Management* organised by the *British Computer Society*, *International Conferences on Software Process Improvement, Research, Education and Training (INSPIRE)*, the *Hawaii International Conferences on System Sciences* and the *Annual European Spreadsheet Risks Interest Group (EuSpRIG) Spreadsheet Symposiums*.

In order to gain an insight into what had already been done on the subject of integrity control of spreadsheet models and spreadsheet development, a thorough review of existing literature was deemed to be the most appropriate approach. The material reviewed included books, journal papers, conference proceedings and articles in other publications. Interviews and face-to-face meetings were considered very important in a research of this nature. Engaging in such interviews and meetings could provide a direct insight into the various aspects of the research, especially the phenomenon of spreadsheet errors and existing tools, techniques and methods used to control the integrity of spreadsheet models. In order to effectively benefit from the use of this research method, the people to be interviewed were carefully chosen. Interviews, meetings and discussions were subsequently held with researchers on the subject, spreadsheet users in academia and industry, people involved in the auditing of spreadsheet models, facilitators of training in spreadsheet modelling and developers of tools for spreadsheet auditing and quality control.

High-level solutions were generated by examining all existing software engineering methodologies, and examining their applicability to the current problem. Pros and Cons of each methodology were presented to the supervisory team, and a favoured candidate emerged (*Jackson Structures*).

The selected methodology was elaborated in logical mode and tried out on some standard business models. These were obtained from standard texts, and from industrial users.

The testing for efficacy was carried out on real users who were students attending courses where the researcher was lecturing. These users were used for trials, in view of the need for statistical significance. This was preceded and guided by research into relevant past experiments.

CHAPTER 2

BACKGROUND

2.1 Introduction

The primary and secondary research questions to be answered by this doctoral research programme were identified and specified in *Chapter 1*. In order to begin addressing these research questions, an investigation was undertaken into the evolution and functions of spreadsheets, and the problem of user-generated spreadsheet errors in terms of their frequency and impact. The views and recommendations of authors and researchers on the subject were subsequently explored to identify possible causes and potential approaches to solving the problem.

This chapter begins by presenting the results of the investigation into the evolution, functions and benefits of spreadsheets. This is followed by a discussion of the phenomenon of user-generated spreadsheet errors. The different aspects of the problem addressed are the trends in spreadsheet errors, the frequency of the errors, and their real-life impact and consequences. The views and recommendations of authors and researchers on the subject are subsequently presented. This also involves a discussion of the need to adopt software engineering and structured methods in spreadsheet development.

2.2 An Insight into Spreadsheets

Prior to the investigation of spreadsheet errors, it is appropriate to gain a basic understanding of spreadsheets as well as their evolution in recent years. Spreadsheet programs attained widespread use since the development of the first electronic spreadsheet package, *VisiCalc*, in 1979 (Brown-87). After the creation of *VisiCalc*, *Lotus 1-2-3* was built for the IBM PC, followed by *Microsoft Excel*, which is presently used on the Windows platform (Butler-97).

The spreadsheet provides a large matrix of rows and columns. Each column is assigned unique letters while each row is identified by a distinct number. Users organise parameters, variables, formulae and components of the spreadsheet model within this framework (Nardi-90). The intersection of a row and column defines a cell. A cell can contain a numeric constant, label or formula. According to Ronen et al (Ronen-89), the tremendous power of spreadsheets is attributable to its ability to relate cells with formulae.

The underlying formula of a cell is not readily visible to the user. It is only the numeric result of the calculation defined by the formula, which is displayed (Brown-87). Formulae perform calculations on absolute values and references to other cells, represented by the corresponding cell addresses. Users can model problems in a spreadsheet and easily automate the calculation of large complex systems using cell formulae (Igarashi-98). When *Lotus 1-2-3* was developed in 1983, macros were added. Creeth (Creeth-85) defines a *macro* as a single computer instruction that stands

for a sequence of operations. Macros further enhanced the functionality of spreadsheets.

Igarashi et al (Igarashi-98) state that spreadsheets are one of the most successful applications making use of visual language techniques, and have the capacity to display and manipulate complex information in tabular form. With the advent of spreadsheets, end-users in business could more easily computerise laborious and time-consuming custom calculations that were needed for a wide range of commercial activities (Butler-97). Bodily (Bodily-86) believes that an important function of the electronic spreadsheet is its ability to support *what-if* analyses of all kinds.

Olsen and Nilsen (Olson-87-88) have described three major advantages offered by spreadsheets. First, the spreadsheet can be easily edited. Second, the values of certain cells can be automatically calculated from the contents of other cells by using formulae. The third advantage is the ability to *copy* a formula from one cell to another while keeping constant the relative location of cells that are referenced.

Spreadsheet based systems are an important part of end-user computing (Cragg-92). They are used for a wide variety of applications. Ronen et al (Ronen-89) believe that the most frequent use of spreadsheets is for decision support and personal productivity. They, however, also state that many spreadsheet applications can in fact be regarded as mainstream information systems applications.

It is important to have an understanding of the different roles taken on by people involved in a spreadsheet project. The number of people needed to carry out a particular role is mainly dependent on the size and complexity of the spreadsheet model. Read and Batson (Read-99) define various roles in spreadsheet model development and use. The *model sponsor* is the person who requests that the model be built and ensures that the required resources are available. Agreement of the objectives of the model is the responsibility of the model sponsor. The *model developer* translates the sponsor's requirements into the actual spreadsheet model. The model that has been built will have at least one *user*. The sponsor and developer of the model may also be its users. The *reviewer* is the person who tests the spreadsheet (Read-99).

2.3 The Phenomenon of Spreadsheet Errors

Numerous publications have recently demonstrated the seriousness of user-generated spreadsheet errors and their adverse consequences or potential impact on businesses. There is substantial anecdotal evidence suggesting that end-user developed spreadsheets can be considered unreliable, inflexible, unmaintainable, and unmanageable (Benham-93). According to Ray Butler of HM Customs and Excise (United Kingdom), even in a domain such as indirect taxation, which involves relatively simple calculations and well-documented calculation rules, spreadsheet models are prone to errors, despite relatively high domain knowledge by developers (Chadwick-00b).

According to Ronen et al (Ronen-89), spreadsheet packages have extended computing to vast numbers of individuals. They argue that for many users, the spreadsheet

program represents their first experience with programming and documentation. Ronen et al (Ronen-89) state that in general, these users have not been trained in systems analysis and tend to overlook the concerns of the professional systems analyst in designing a system. The practitioner literature has discussed a number of problems with spreadsheet construction (Ronen-89).

The phenomenon and magnitude of spreadsheet errors can be viewed from three different perspectives (Rajalingham-99). They are as follows:

- frequency of the errors
- impact and real-life consequences of spreadsheet errors
- types and classes of specific errors

The first two aspects of the problem of spreadsheet errors are discussed in this chapter while the third is analysed and presented in the next chapter, *Chapter 3: Analysis and Classification of Spreadsheet Errors*.

2.3.1 Overview of Trends in Spreadsheet Errors

There is more than sufficient evidence from various reliable sources that the problem of spreadsheet errors has been experienced for decades. This also appears to be the situation today despite the advent of various tools and techniques for controlling the integrity of spreadsheet models.

In 1998, research carried out by Pricewaterhouse Coopers revealed that there was a trend of increasing spreadsheet model size and complexity (Whittaker-99). Whittaker argues that the trend towards larger model size and complexity is clear, and there is every possibility that this trend will continue in future. A conclusion that can be drawn from this statement is that the frequency of spreadsheet errors is steadily increasing as in general, the number of errors is proportionate to the spreadsheet model size and complexity.

Another factor that influences the escalating frequency of spreadsheet errors is the speed and simplicity of building spreadsheet models. Howitt (Howitt-85) believes that spreadsheets create the opportunity to make more mistakes and multiply them rapidly due to the speed and simplicity of spreadsheet application development. This indicates that with the increasing use of spreadsheets over the years, users have been making more errors and quickly multiplying them.

There is extensive material clearly indicating an increasing use of spreadsheets, resulting in a proportional increase in the frequency of spreadsheet errors. According to Carlsson (Carlsson-89), in business, spreadsheet programs have become one of the most frequently purchased and used personal computer programs. Isakowitz et al (Isakowitz-95) state that there has been increasing sophistication and power of commercial spreadsheet packages. They believe that spreadsheet programs have transformed the concept of end-user computing, creating a new computational paradigm that offers a unique combination of ease of use and unprecedented modelling power. This has encouraged the widespread use of spreadsheets in business

and resulted in spreadsheet programs becoming the most popular decision support tool in modern business (Isakowitz-95).

Ray Butler of HM Customs and Excise, United Kingdom (Butler-97) states that spreadsheets are among the most dangerous and error-prone development platforms. The figures on the frequency of user-generated spreadsheet errors are truly astounding and indicate a high probability of imminent disaster scenarios around the world (Chadwick-00b). An important conclusion that can be drawn on the trends in spreadsheet errors is that with the profound increase in the production and use of spreadsheet models over the years, the frequency and impact of the errors have also steadily increased.

2.3.2 Frequency of User-generated Spreadsheet Errors

There have been various publications containing information on the frequency of spreadsheet errors. Despite the widespread use of spreadsheets, there has been extensive anecdotal and experimental evidence that electronic spreadsheets are highly susceptible to user-generated errors (Brown-87). Although electronic spreadsheets are immensely beneficial to accountants and financial analysts, they may have a disastrous impact on critical business decisions (Hayen-89). After a thorough review of relevant literature, various cases have been selected and presented in this section to demonstrate the appalling frequency of user-generated spreadsheet errors.

Based on the results of an experiment, Brown and Gould (Brown-87) concluded that even a substantial percentage of spreadsheets created by experienced spreadsheet users contained one or more errors. According to Freeman (Freeman-96), Coopers and Lybrand (London), reported that over 90% of all spreadsheets they had examined, with more than 150 rows, contained at least one significant formula error. This is an extremely high figure and if the errors had gone undetected, they could have had a devastating effect on the business.

An article in *New Scientist* (Ward-97) has reported that a decade's worth of research findings of Professor Raymond Panko at the University of Hawaii revealed that spreadsheets had a dangerously high rate of errors. It appears that on average, 30% of spreadsheets contain errors, many of which are serious. According to Professor Panko, the problem is that spreadsheets demand a level of accuracy that people find difficult to manage.

A financial model review by KPMG Management Consulting, London (KPMG-97) stated that in 95% of the financial models audited, at least 5 errors had been found. The review also revealed alarming statistics concerning defects and flaws in the spreadsheet development process, addressing the project management, technical and analysis aspects. An audit of spreadsheets from over 21 major UK banking and financial organisations revealed that 92% of the spreadsheets dealing with tax issues had significant errors while 75% had significant accounting errors (KPMG-98b).

An excellent compilation of studies on the frequency of spreadsheet errors has been produced by Panko and Halverson (Panko-96,98,00). The findings are presented in *Appendix A*.

There is also substantial anecdotal evidence from the commercial sector, of the high frequency of user-generated errors in spreadsheet models. A selection of the relevant cases is presented below in chronological order of publication.

- Creeth (Creeth-85) has stated that according to industry experts, one out of every three spreadsheet printouts contains errors.
- An article from *Personal Computing* (Ditlea-87) reported that a Houston consultant with Price Waterhouse had found 128 errors in 4 spreadsheet models that had already been in use for months.
- Estimates from the trade press on the number of spreadsheets that contain errors range from 20 to 40 percent (Brown-87).
- According to Davies & Ikin (Davies-87), out of 19 worksheets (from 10 different firms) audited, 4 (21%) had serious errors, while 13 were considered to have inadequate documentation, and 10 did not use cell protection.
- Roberts (Roberts-88) found one or more errors in 80% of spreadsheet models audited.
- In an inspection of 20 operational models of 10 firms, errors were found in at least 25% of the models. Apart from that, other problems were also found (Cragg-93).
- In an Australian mining firm, an audit found that 30% of the spreadsheets audited had been corrupted because cell protection had not been used, and users typed numbers into formula cells (Dent-95).

2.3.3 Impact and Consequences of User-generated Errors

Spreadsheet errors can be devastating because the data is often the foundation on which many organisations make critical decisions (Freeman-96). It is important to examine the adverse consequences of the problem of spreadsheet errors in real life. This enables a distinct comprehension of the magnitude of the problem and an assessment of the seriousness of the situation.

The information presented in this section has been obtained from numerous publications. It must however be noted that these are based only on reported cases. It is believed that there are many other similar cases that have not been brought to public attention due to fear that it might adversely affect the reputation of the organisation involved.

There are publications from more than a decade ago with clear indications that user-generated spreadsheet errors have caused serious disruption of business. Although these cases are not based on formal research, they do show that spreadsheet errors were considered important enough to be reported in the general business and computing press.

A subset of significant reported cases is provided below in chronological order of publication.

- According to an article in *Business Week* (Business Week-84), a Midwestern firm's estimated taxes had been \$5,000 off due to an incorrect formula for assessing salvage value in the spreadsheet.
- The article (Business Week-84) has also stated that in the forecast for a new product, the forecast sales was \$8 million over. Fortunately, it was detected in time to prevent any serious damage.
- In another case (Business Week-84), a person ordered 30,000 units at \$4 each, but the plan had changed and the company only needed 1500. Quite a lot of money was therefore tied up in excess stock.
- Two spreadsheets with 15,000 cells were used to project the market for CAD equipment. The numbers were rounded off to whole dollars and even the inflation multiplier, which should have been 1.06 was rounded off to 1. Consequently, the market was underestimated by \$36 million (Business Week-84).
- A Dallas-based oil and gas company fired several executives for spreadsheet model oversights that cost the company millions of dollars (Freeman-86).
- Work by Ditlea (Ditlea-87) published in *Personal Computing*, offer several cases showing the adverse impact of spreadsheet errors on businesses. The controller of James A. Cummings, Inc., a Florida construction company, was putting together a Symphony spreadsheet model to bid on a \$3 million office complex. His formula to calculate the bid did not include a figure of \$254,000 for overhead costs that he had later inserted at the top of a column of figures. This entry fell outside the range of numbers to be added by the @SUM (=SUM in MS Excel) function in his formula. The undetected error resulted in a loss for the company when the bid was won (Simkin-87).
- In another case (Simkin-87), a consultant called Larry Nipon found an error that would have cost \$1.5 million had it gone unchecked. The error was actually identified by *Cambridge Spreadsheet Analyst*, a spreadsheet auditing program.
- Davies and Ikin (Davies-87) have found that out of 19 operational models audited from 10 different firms, 4 (21%) had serious errors, including a \$7 million error in interdivisional transfers, different exchange rates for Australian dollars in the same time period, and a negative balance for stock on hand. The effect errors like these can have on the company is simply unimaginable.
- According to Woodbury G G (Woodbury-89), in a North Carolina election, results of the election were about to be incorrectly posted. Mr Woodbury, using a calculator, detected an inconsistency. Examination found an incorrect cross-tabulation in the spreadsheet being used to post the results.

The following are three of the more recently reported cases:

- Dhebar (Dhebar-93) reported that a firm called Fortune 500 used discounted cash flows to evaluate investment proposals and an important figure was not updated for 8 years. The formula and discount rate had apparently been established long ago, were never documented and made by a person who had left the company. Although the prime rate rose from 8% to over 20% between 1973 and 1981, the spreadsheet was kept at 8%. This is potentially detrimental to the business.
- At Fidelity, a spreadsheet was used to report distributions for various funds. For the huge Magellan fund, a \$4.32 per share capital gains distributions was forecast in November, and investors were notified. However in December the company announced there would be no distribution. A clerical worker put the wrong sign in front of a \$1.2 billion ledger entry. This "created" a \$2.3 billion gain in place of the real \$0.1 billion loss. This may have affected buyers, some of whom may have sold to avoid the distribution and missed a price rise, others of whom may have waited to buy to avoid the distribution and also missed the price rise (Savitz-94).
- According to an article in *New Scientist* (Ward-97), a study by the Computer Audit Unit of HM Customs and Excise (UK) found that as a result of errors, spreadsheets were out by amounts ranging from a few hundred pounds to millions of pounds. These errors were made by people when filling in computer spreadsheets used by companies to keep track of their cash.

These reports demonstrate that the occurrence of user-generated spreadsheet errors is indeed a critical problem for businesses and requires immediate attention. If this situation prevails, organisations will, inevitably, be suffering great financial losses as a result of incorrect decisions made based on their erroneous and unreliable spreadsheet models.

2.4 Need for a Disciplined and Structured Approach

Spreadsheet models are increasingly being used in decision-making within organisations (Cragg-93). However, much past research and published reports have firmly established that there is no unified approach to spreadsheet development in industry. Spreadsheet development can, in many ways, be compared to the days of main-line software development before the advances due to structured programming, analysis and design. Isakowitz et al (Isakowitz-95) state that in spite of the increasing sophistication and power of commercial spreadsheet packages, there is still a lack of a formal theory or methodology to support the development and maintenance of spreadsheet models.

Findings from research carried out over several years have revealed the need for a new approach or discipline for spreadsheet development. This is evident from the constant call for a new structured approach, in many recent publications. Studies have also discovered a general lack of policies on spreadsheet development. A collection of these studies have been organised and presented by Panko and Halverson (Panko-96). This can be found in *Appendix B*. Hall (Hall-96) argues that with the high probability

of the occurrence of spreadsheet errors, there is an obvious need for some formalised control policy in the spreadsheet development process.

Panko and Halverson (Panko-96) state that surveys of spreadsheet development have revealed that strict development disciplines have not been followed in spreadsheet development, as they are in conventional programming. They also indicate that the process of building spreadsheet models has been largely informal and emphasise on the need to adopt programming disciplines in order to deal with complex spreadsheets. Panko and Halverson also point out the fact that there is an obvious need to adopt traditional programming disciplines due to the similarity between spreadsheet and programming errors.

Spreadsheet applications are more vulnerable to poor design and errors compared to conventional programs, as many spreadsheet users have not been trained in systems analysis and software engineering (Davis-96). Benham et al (Benham-93) propose the adoption of the techniques of structured analysis, design and programming to spreadsheets, in order to enhance the quality of the applications. Their reason is that such structured techniques were developed to address the shortcomings of early data processing systems. Howitt (Howitt-85) states that users' failure to employ a consistent and thorough design methodology is due to the speed and simplicity of spreadsheet model development.

According to the publications by Davies and Ikin (Davies-87) and Cragg and King (Cragg-93), spreadsheet development, in most cases, has been found to be very informal and lacking in the use of important development disciplines. David Finch, Head of Internal Audit at Superdrug plc (United Kingdom) believes that there is often inadequate control and standardisation in the process of spreadsheet development by end-users in different departments (Chadwick-00b). Creeth (Creeth-85) has called for quality control over the use of spreadsheet models.

An investigation carried out into the spreadsheet practices in ten firms revealed that spreadsheet models were usually built in an informal, iterative manner, by people with very little training (Cragg-93). This created an awareness of the need for increased training as well as setting and enforcing organisational spreadsheet standards (Cragg-93). According to Ray Butler (Butler-97), the problem with spreadsheet building is that users do not regard spreadsheet models as computer programs requiring specification, testing and documentation. Ray Butler believes that a reduction in the risk of errors can be achieved by using a more formalised development and testing methodology for spreadsheet applications (Chadwick-00b).

Hendry and Green (Hendry-94) have pointed out that the great disadvantage of spreadsheets is that it is so easy. They suggest that instead of creating the whole spreadsheet first and then checking for errors, errors ought to be checked for at various stages of the development process. They believe that this would enable the detection and correction of errors without missing many. This strategy of stage-by-stage component testing is a software engineering-based technique.

Ronen et al (Ronen-89) express concern over the lack of formal analysis or documentation in spreadsheet development. They state that a structured approach to spreadsheet design can help reduce the occurrence and seriousness of problems with

spreadsheets. According to Isakowitz et al (Isakowitz-95), a decrease in spreadsheet errors can be achieved by adopting principles of structured methods from software/system engineering. Kavanagh (Kavanagh-97) states that end-users are putting their companies at risk by building spreadsheets without realising that this demands the discipline of traditional programming.

Based on these studies and published reports, we can arrive at the firm conclusion that the application of structured methods and the adoption of a disciplined approach based on programming (or software engineering) principles in spreadsheet development, is indeed imperative. This research programme investigates the possibility of applying such methods in order to effectively address the phenomenon of user-generated spreadsheet errors and enhance the integrity of spreadsheet models.

2.5 Summary

Spreadsheet programs have been in widespread use from the development of the first electronic spreadsheet package, *VisiCalc*, in 1979, to the current windows-based *Microsoft Excel*. The different roles in spreadsheet modelling include the *model sponsor*, the *model developer*, the *user* and the *reviewer*. There are three important perspectives to the phenomenon of spreadsheet errors. They are the *frequency of the errors*, the *real-life consequences of spreadsheet errors* and the *types and classes of specific errors*.

An important conclusion that can be drawn on the trends in spreadsheet errors is that with the profound increase in the production and use of spreadsheet models over the years, the frequency and impact of the errors have also steadily increased. Users have been making more errors and quickly multiplying them.

It is evident from numerous publications that the frequency of user-generated spreadsheet errors is indeed appallingly high. These publications contain extensive anecdotal and experimental evidence of the vulnerability of spreadsheet models, from both business and academia sources. This is further corroborated by research findings.

Numerous studies and audits have been carried out on the impact of user-generated spreadsheet errors. These along with a huge collection of reported cases over two decades, have clearly revealed the extent of damage that has been caused to businesses, as well as potential future disruption.

Researchers on the problem of spreadsheet errors and authors of numerous relevant publications have relentlessly stressed on the need for a new approach to spreadsheet modelling. They have constantly recommended a structured and disciplined approach to spreadsheet model development based on software engineering methods and techniques.

CHAPTER 3

ANALYSIS AND CLASSIFICATION OF SPREADSHEET ERRORS

3.1 Introduction

The previous chapter, *Chapter 2*, presented an insight into spreadsheets and described the problem of user-generated spreadsheet errors in terms of their frequency and impact. The importance of applying software engineering and structured methods to spreadsheet development was also discussed. As mentioned in *Chapter 2*, the phenomenon and magnitude of spreadsheet errors can be viewed from three distinct perspectives. The first two perspectives were addressed in *Chapter 2* by investigating the frequency and real-life consequences of spreadsheet errors. This chapter focuses on the third perspective, the types and classes of specific errors.

A thorough review of literature concerning spreadsheet development and the relevant integrity issues, has revealed a significant deficiency. Very little research has been devoted to the study and examination of specific errors that occur in spreadsheet models. Therefore, an analysis of specific types of errors has been conducted as a precursor to the development of strategies and solutions to deal with the problem effectively.

There are numerous types of user-generated spreadsheet errors, with different characteristics and attributes. As such, an essential and integral part of the analysis of specific types of spreadsheet errors would be to develop a classification of these errors. This chapter presents a more comprehensive classification or taxonomy of spreadsheet errors than ever presented or published before, following a meticulous analysis of specific types of user-generated spreadsheet errors from a wide variety of sources. The classification is based on a rational taxonomic scheme and is supported by a selection of generic and specific examples. The spreadsheet error taxonomy is produced by analysing the nature and characteristics of the different types and categories of specific errors. Earlier versions of the taxonomy have been published (Rajalingham-98, 99, 99a, 00, 00a, 00b). The classification facilitates more effective comprehension of the different types of spreadsheet errors.

3.2 The Concept of *Taxonomy* or Classification

In a broad sense, taxonomy is the science of classification, though more strictly, it refers to the classification of living and extinct organisms. The term is derived from the Greek *taxis* ("arrangement") and *nomos* ("law"). It is important to note, however, that there is no special theory which lies behind modern taxonomic methods. In attempting to define *taxonomy* within the context of spreadsheet errors, it would be appropriate to investigate the definition of this term in other fields of study. In biology, *taxonomy* refers to the establishment of a hierarchical system of categories on the basis of presumed natural relationships among organisms. The goal of classifying is to place an organism into an already existing group or to create a new group for it,

based on its resemblances to and differences from known forms. To this end, a hierarchy of categories is recognised (Britannica.com-99-00).

Based on the definitions borrowed from other disciplines, we can extend the concept of taxonomy to the classification of spreadsheet errors. For our purposes, the *spreadsheet error taxonomy* can be defined as a hierarchical system of classes of spreadsheet errors on the basis of common characteristics and relationships.

3.3 Rationale for the Classification of Spreadsheet Errors

There are various reasons for developing a classification of spreadsheet errors. The most important purpose of creating a taxonomy is that it is a methodical approach to problem analysis. The analysis of the different types of errors based on this approach is likely to improve comprehensive testing of a spreadsheet development methodology. The development of a taxonomy of spreadsheet errors also forces us to gain a deeper understanding of the characteristics of an error as well as the nature of its occurrence. A comparison can also be made with other related errors belonging to the same category.

An insight into the features and nature of an error is of paramount importance, in order to prevent the occurrence of the error or develop a method of detecting its presence. The classification of spreadsheet errors would inevitably involve an identification of similar characteristics and properties between certain errors. This can be used as a basis for developing similar approaches to address spreadsheet errors within the same category or taxonomic group. Knowledge of the characteristics of an error also enables analysis of its potential impact and frequency. It is highly probable that other errors in the same category would have the same degree of seriousness.

3.4 Derivation of the Taxonomic Scheme

This section discusses the factors and approaches that have been considered in the development of the taxonomy. As indicated earlier, there is no special theory which lies behind modern taxonomic methods. As such, an investigation had to be carried out into the taxonomic methods used in other fields. These methods of classification have been widely employed in the fields of *zoology* and *botany*.

Based on the principles of classification adopted in *zoology* and *botany* (Britannica.com-99-00), spreadsheet errors can be classified using a similar taxonomic scheme. The process of classification consists of the following steps:

- A specific type and example of a spreadsheet error is obtained.
- The error is compared with the known range of variation of spreadsheet errors.
- The error is correctly identified if it has been described, or a description showing similarities to and differences from known categories, is prepared. If the error is of a new type, it is assigned to a new category or class.

- The best position for the error is determined in the existing classification. This may also involve determining what revision the classification requires as a consequence of the new discovery.
- Available evidence is used to further suggest and describe the nature of the error, its possible cause(s) and other characteristics. In order to do carry out these tasks, there has to be in place a recognised system of ranks in classifying, recognised rules, and a procedure for verification, irrespective of the group being examined.

It is clear from the last step that the process of spreadsheet error classification requires a recognised system of ranks, rules and a verification procedure. An investigation of a taxonomic method that addresses these requirements revealed that there are two possible approaches to structuring the ranks within a taxonomy. Its system of ranks can either be based on a *binary* approach or a *bushy* approach.

The bushy approach was initially adopted and assessed. This method involves a top-down approach that produces a hierarchical taxonomy. A category at any level or rank can be divided into two or more classes. These categories are created as a result of studying the nature and characteristics of the errors. An example of an earlier version of the proposed taxonomy using the bushy approach is shown in *Figure 3.1* (Rajalingham-99, 99a, 00, Chadwick-99).

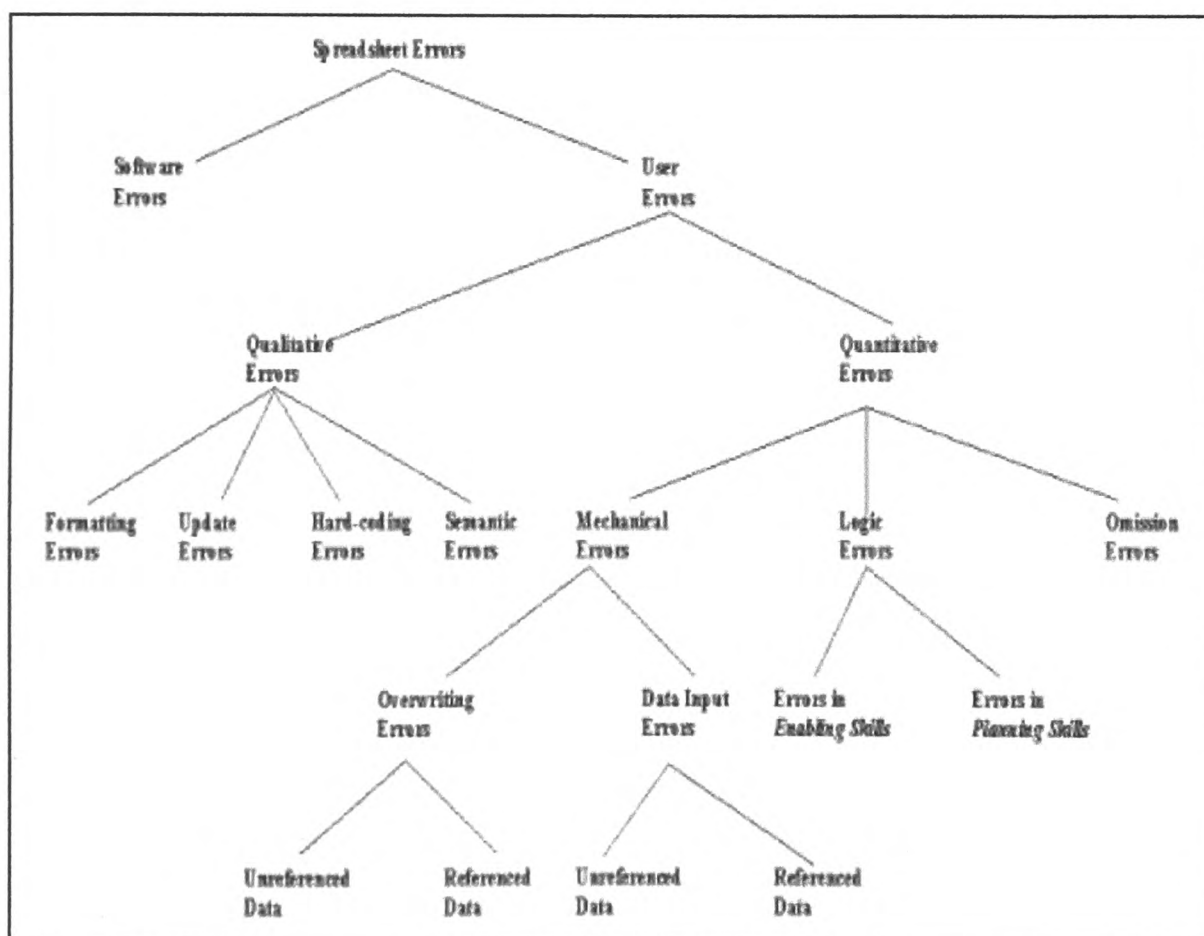


Figure 3.1: Taxonomy Using a *Bushy* Approach

The *bushy* taxonomic structure shown in *Figure 3.1* was found to have certain limitations. It was difficult to navigate down the taxonomic tree to assign a specific error to a class. With certain errors, it was also possible to place them in two or more different classes. This could potentially lead to an ambiguous interpretation of an error.

In order to address these limitations, the alternative *binary* approach was considered and assessed. Like the bushy structure, this method is also based on a top-down approach, resulting in a hierarchical taxonomy. However, at each stage of the taxonomy, the binary approach uses *dichotomies* or divisions into two mutually exclusive (*non-overlapping*) groups, to classify the errors. This eliminates the possibility of positioning the same type of error in different parts of the taxonomy and causing an overlap of the different categories of spreadsheet errors.

This feature of the binary approach enables a far more straight-forward way of assigning a specific error to a taxonomic class. A simple IF-THEN-ELSE rule or constraint can be used to navigate down the taxonomy tree and position errors in appropriate classes. This is demonstrated in the next section. Furthermore, as a rule, at each stage where a dichotomy is produced, only a single factor, representing a distinct aspect of the error is used. This reduces ambiguity of class definition at each rank.

To this end, the following aspects of a particular type of spreadsheet error are analysed:

- (i) Manifestation of the error
- (ii) Cause of the error
- (iii) The role of the person responsible for the error
- (iv) The cognitive state of the person responsible for the error
- (v) The stage of the spreadsheet building life cycle where the error occurs
- (vi) The relevant view of the spreadsheet model system

In view of the advantages of the *binary* method compared to the *bushy* method, the binary approach has been adopted as the basis of a rational taxonomic scheme for classifying spreadsheet errors. The taxonomic scheme also involves the conventional process of classification (as used in *zoology* and *botany*) and an analysis of the nature, properties and characteristics of spreadsheet errors.

3.5 The Classification of Spreadsheet Errors

An important point to be clarified at this stage is that the classification is confined to only *user-generated spreadsheet errors*, as opposed to system or software-generated errors. The issue of detecting or correcting flaws in the spreadsheet software is beyond the scope of this research. User-generated errors can be defined as errors (or potential errors) produced or caused by the developer(s) or end-users of the spreadsheet model and can therefore be controlled or prevented by them.

Whenever, the term *error* is used in this thesis, it should be noted that it has a broader definition encompassing both *actual errors* and *potential errors*. The errors include *flaws*, *slips* and *mistakes*. Slips are errors that occur when the intention to act fits the intended goal but the action is not carried out according to plan. Mistakes, on the other hand, are errors that occur when an action is carried out as intended but the action itself is not appropriate to the task (Chadwick-97).

It is also appropriate to state at this juncture that in the process of classifying certain specific errors, assumptions had to be made about the precise cause of the errors,

where this is not clearly indicated by the source. It is possible for the same error to be assigned to a different category, should the actual cause not match the assumed cause.

Figure 3.2 displays a comprehensive classification of user-generated spreadsheet errors. At the highest level, spreadsheet errors can be divided into two major categories, namely *qualitative errors* and *quantitative errors*. The classification factor used at this stage is the manifestation of the error. Panko and Halverson (Panko-96) have also broadly classified spreadsheet errors as being either quantitative or qualitative.

By examining the manifestation of a specific type of spreadsheet error, it can be clearly determined whether it is quantitative or qualitative, but not both. Any error or flaw which is not quantitative has to be qualitative. Therefore, spreadsheet errors can be divided into two non-overlapping categories of *quantitative* and *qualitative* errors. This can be expressed in a form identical to a structured program.

For all *user-generated* spreadsheet errors,

IF numerical error causing incorrect bottom-line value
THEN *quantitative* error
ELSE *NOT quantitative* error (i.e. *qualitative* error)

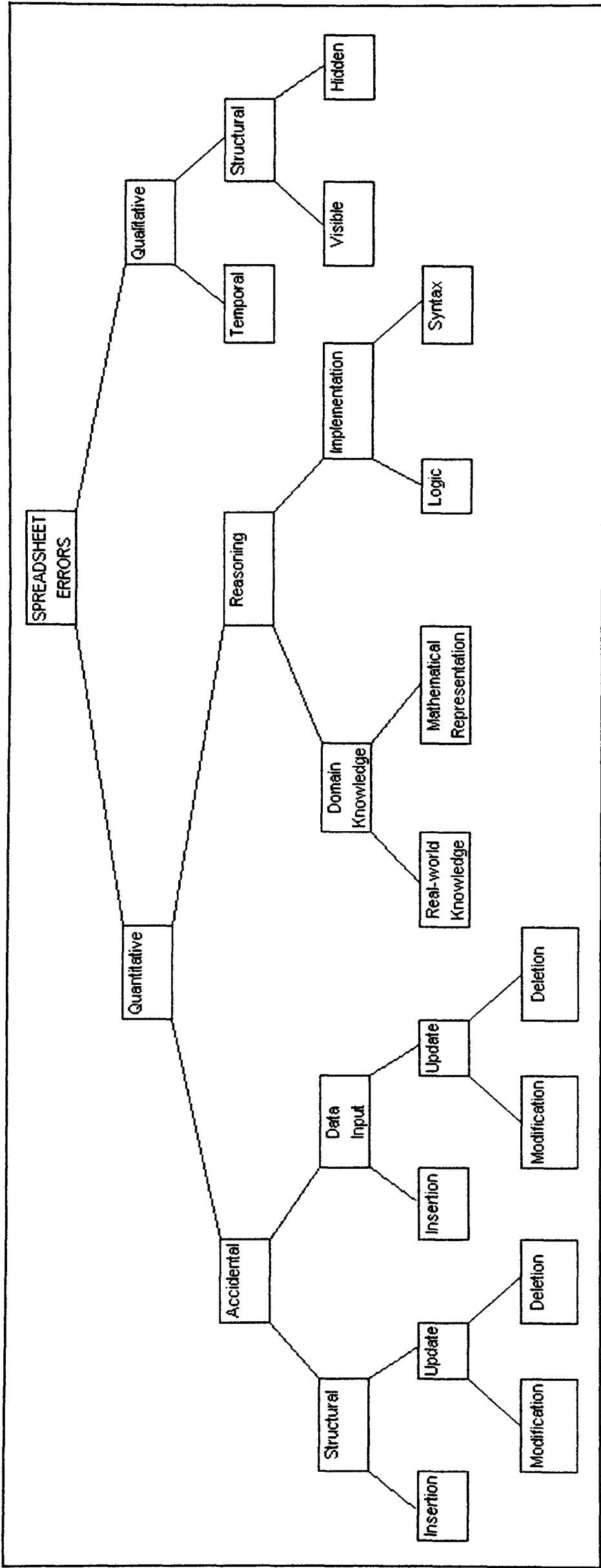


Figure 3.2: Taxonomy of User-generated Spreadsheet Errors

3.5.1 Quantitative Errors

Quantitative errors are numerical errors that lead to incorrect bottom-line values (Panko-96). They simply produce wrong data in the spreadsheet model. Based on an analysis of the cause of the error, a dichotomy of *accidental* and *reasoning errors* can be used to capture the different types of quantitative errors.

Any error or flaw which is quantitative and not accidental must have been produced as a result of a mistake in reasoning and can therefore be considered to be a reasoning error. Common sense would dictate that an error cannot be both accidental and caused by a mistake in reasoning. Therefore, quantitative errors can be divided into two non-overlapping categories of *accidental* and *reasoning errors*.

For all *quantitative errors*,

IF error is caused by negligence or carelessness
THEN ***accidental*** error
ELSE *NOT accidental* error (i.e. ***reasoning*** error)

It is important to state at this juncture that the dimension of fraud is not taken into account when developing the classification framework for quantitative errors. This is because any error can be deliberately produced with fraudulent or malicious intent and disguised as an accidental or reasoning error, unless of course the criminal motive is blatantly obvious as in this example:

A user rewrites a payroll equation as follows (Stang-87):

IF EMPLOYEEID = MINE
THEN PAYCHEQUEAMT = HOURS X RATE X 1.03
ELSE PAYCHEQUEAMT = HOURS X RATE.

1. *Accidental Errors*

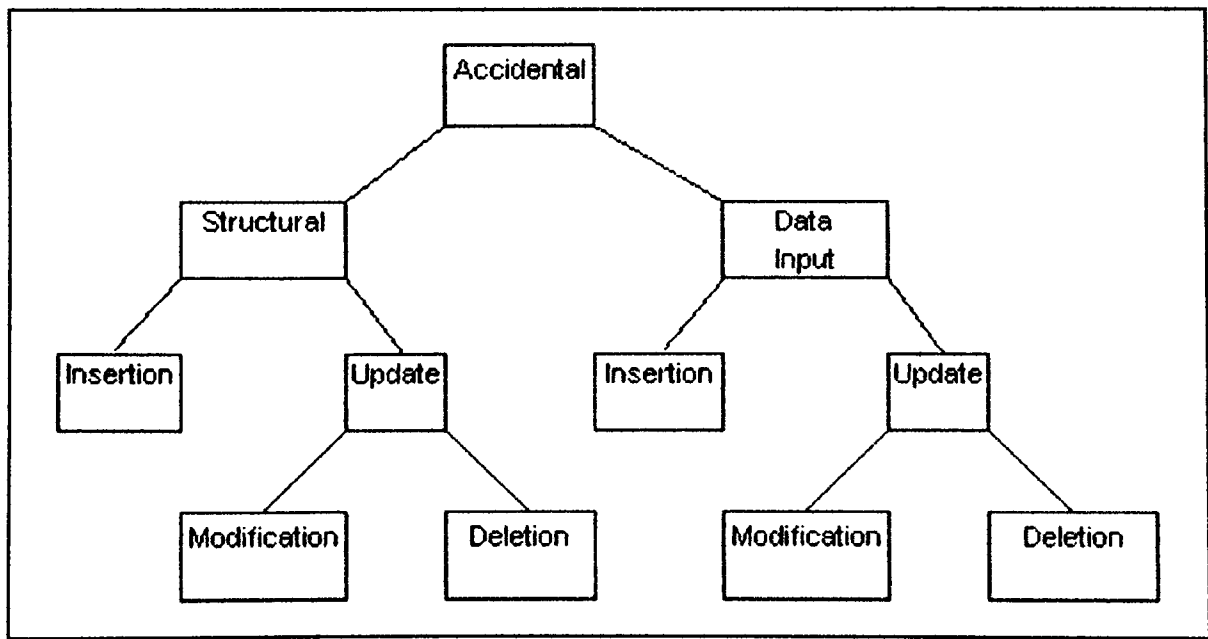


Figure 3.3: Accidental Errors

Accidental errors are mistakes and slips caused by negligence, such as typographical or pointing errors. Though quite frequently occurring, they have a high chance of being spotted and corrected immediately by the person committing the error. Some, however, do go undetected and could lead to incorrect values in the spreadsheet model. After a close examination of various types of accidental errors, it has been found that they can be further divided into two distinct categories. The taxonomic factor used to achieve this is the user role responsible for the error. As such, an accidental error can either be a *structural error* or a *data input error*.

Any user-generated error or flaw which is not produced by the model developer could only have been caused by the end-user(s) of the model. Errors caused by end-users are defined as *data input errors* as these errors occur when end-users insert, alter or remove data in the models. The structures or templates of these models would have already been constructed by the model developer. Based on this understanding of the two distinct user roles, accidental errors can be divided into two non-overlapping categories of *structural* and *data input* errors.

For all *accidental* errors,

IF error is caused by the model developer

THEN ***structural*** error

ELSE *NOT structural* error (i.e. ***data input***)

(a) **Structural Errors**

Structural errors are errors produced by the developer of the spreadsheet model. These errors are produced when creating or altering the structural or programmed component of the spreadsheet model. Therefore, these errors can again be segregated into two categories, namely, *insertion* and *update* errors. Though the structural component of a spreadsheet model consists of schema and editorial sub-components (described elaborately in *Chapter 5*), these errors primarily concern the schema of the model. As the editorial parts of the model are mainly textual and not referenced by any formulae, they do not produce numeric or bottom-line errors. As such they are not classed as quantitative errors. They are in fact qualitative errors as these editorial errors can degrade the quality of the model and distort its semantics.

Any developer-generated accidental error or flaw not produced while creating the structural aspects of the spreadsheet model must have occurred while they are being altered. This enables the disjointed division of accidental structural errors into *insertion* and *update* errors.

For all *structural* errors,

IF error is produced when creating the structural aspects of the spreadsheet model

THEN ***insertion*** error

ELSE *NOT insertion* error (i.e. ***update*** error)

(i) Insertion Errors

These errors occur while the developer is creating the structures of the spreadsheet model. The model at this stage would be prone to accidental errors such as typographical errors, pointing errors, duplication and omissions. As the activity is carried out by the model developer, the cells affected would usually be formula cells.

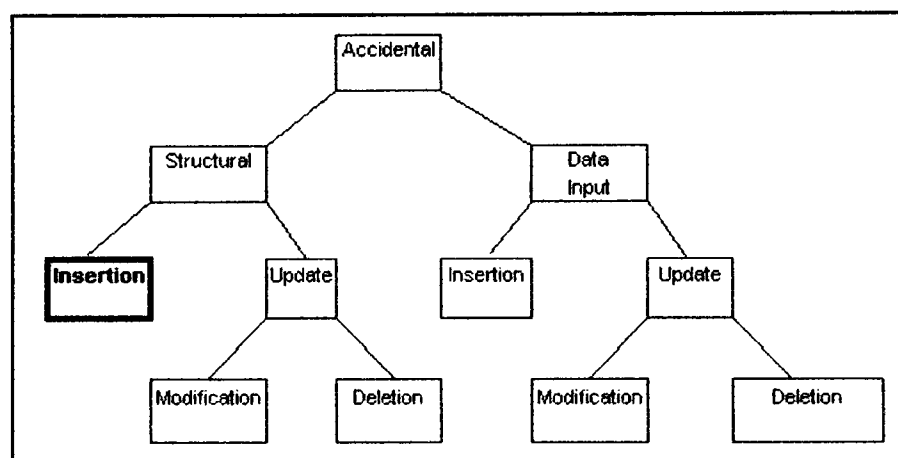


Figure 3.4: Insertion Errors (Structural)

Example 1: Omissions

Omissions are key factors or variables that are left out of the model (Cragg-93), that should be there. They often result from a misinterpretation of the situation. Human factors research has shown that omission errors are especially dangerous, because they have low detection rates (Panko-96). This is a problem which is at the heart of any modelling exercise. KPMG (KPMG-98) reported that references were made to worksheets that does not exist (Cragg-93).

Example 2: Pointing Errors

Pointing errors refer to errors caused by references being made to wrong cells or cells in the wrong location. The model developer types the wrong cell coordinates in composing the formula (Brown-87). As a result of carelessly entering incorrect cell or range addresses into formulae, the formulae themselves produce incorrect results. Pointing errors could therefore also manifest themselves in the form of references to blank cells and non-numeric cells or cause the presence of figures that are not used.

KPMG (London) found that in a client model, a formula incorrectly referred to a different range. Consequently, a *monthly average* was overstated by approximately *five minutes*, but did not affect *revenue* (KPMG-98).

A common example of erroneous formulae involves incorrect range specification in formulae. This occurred frequently with the use of functions that summed a range of entries in a row or column (Cragg-93). Based on *Figure 3.5* (Chadwick-97a), the correct formula in **E9** is **=SUM(E5:E8)** but the formula **=SUM(F5:F8)** is entered instead.

	A	B	C	D	E	F
1	Lazy Days Staff Budget Costs 1995-1996					
2		Staff	Basic	Overtime	Total	Average
3		Numbers	Wages £	Wages £	Wages £	Wage £
4						
5	Managers	1	17700	0		
6	Grade 1	3	45540	1400		
7	Grade 2	9	122340	2000		
8	Grade 3	12	102350	0		
9	Grand Totals	25	287930	3400		

Figure 3.5: Example to Illustrate Pointing Error

Example 3: Circular Reference

Circular reference can be caused by an accidental error by the model developer as a result of mistyping or a pointing error. With reference to *Figure 3.6*, an example of this error would be the entry of the formula **=SUM(D6:D10)** into cell **D10**, instead of **=SUM(D6:D9)**.

	C	D	E	F	G	H	
		Number of Staff	Day Wages £	Night Wages £	Total Wages £	Average Wage £	
Grade 1		1	17700.50	0.00	=SUM(E6:F6)	=G6/D6	5
Grade 2		3	45540.00	1400.55	=SUM(E7:F7)	=G7/D7	6
Grade 3		9	122340.00	2000.00	=SUM(E8:F8)	=G8/D8	7
Grade 4		12	102350.25	0.00	=SUM(E9:F9)	=G9/D9	8
Grand Total		=SUM(D6:D10)	=SUM(E6:E9)	=SUM(F6:F9)	=SUM(G6:G9)	=G10/D10	9
							10

circular reference

Figure 3.6: Circular Reference

(ii) Update Errors

These errors occur while the developer is altering the structural or programmed component of the spreadsheet model. The model at this stage would be prone to accidental errors such as typing errors, overwriting and deletion.

For all *structural update* errors,

IF If error is produced as a result of incorrectly changing the
 structural or programmed component of the model
THEN ***modification*** error
ELSE *NOT modification* error (i.e. ***deletion*** error)

Modification Errors

These errors occur as a consequence of incorrectly or inaccurately modifying the structural or programmed component of the spreadsheet model. The trade press indicates that modification of spreadsheets is a more error-prone task than the original creation of spreadsheets (Brown-87). As the activity is also carried out by the model developer, the cells affected would normally be formula cells.

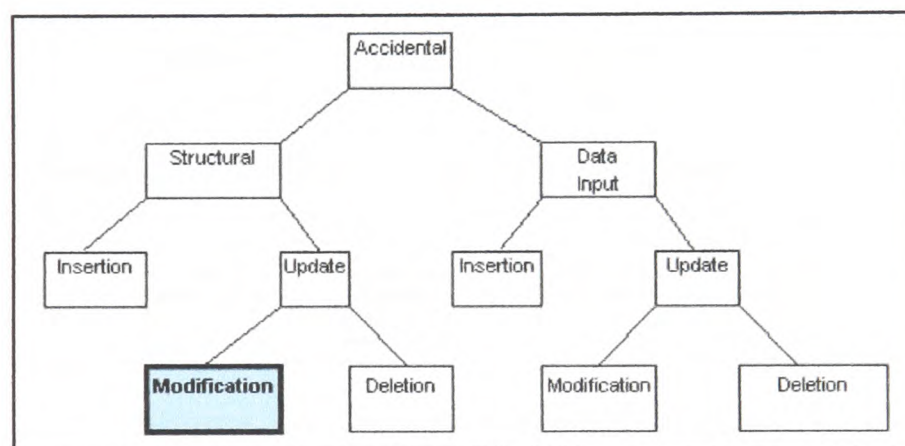


Figure 3.7: Modification Errors (Structural)

Example 1: Formulae Overwritten with Data

Data is incorrectly entered into a cell previously containing a formula, overwriting the formula and invalidating the model (Cragg-93). Hayen (Hayen-89) also points out this error, stating that when data are entered, they may be entered on top of formulae and wipe out the desired formula. This is a very common error made by the end-user during data entry. A simple solution to the problem would be to use cell protection.

Quoting a real-life example, in an Australian mining firm, an audit found that 30% of the spreadsheets audited had been corrupted because cell protection had not been used, and users typed numbers into formula cells (Dent-95).

Any user capable of making a typing mistake is also capable of entering data on top of an unprotected formula. The data may resemble the result of the formula during initial entry; later, when other data are changed, the formula won't contribute to the total (Stang-87).

According to Nixon (Nixon-01) these errors can be easily detected using many of the tools available in the market today. He states that the success in finding these errors is

largely due to the colour-coded overlays that could be applied to the worksheet, making these errors stand out.

Example 2: Formula Overwritten with an Incorrect Formula

Similar to the previous example. However, the correct formula is accidentally replaced with an erroneous formula. Unlike the previous example, this error is usually committed by the model developer.

Deletion Errors

These update errors, on the other hand, are produced as a result of deleting or erasing existing elements of the structural or programmed component of the spreadsheet model. These errors too are therefore caused by model developer.

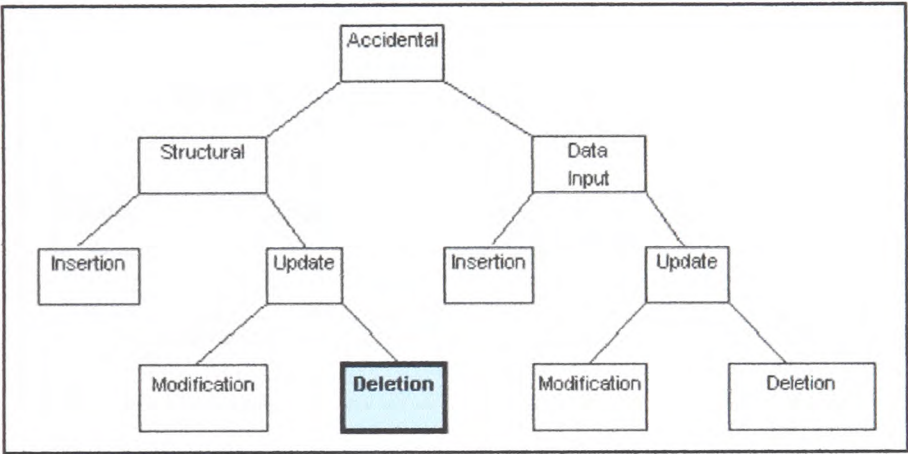


Figure 3.8: Deletion Errors (Structural)

Example 1: Deletion of a Formula

A correct formula required by the spreadsheet model is accidentally erased either by the model developer or during data entry. The main cause of this error is the failure to protect cells containing formulae.

(b) Data Input Errors

Data input errors are errors made by end-users who merely manipulate the spreadsheet model. They are caused by erroneous entry of data required by the model. These errors can occur while either entering new data or amending/updating existing data. This can be the basis upon which these errors are further categorised. The two subdivisions would be *insertion* and *update errors*.

Any data input error not produced while entering new data into the spreadsheet model could only have occurred while altering or amending existing data. Therefore, data input errors can be exclusively divided into two categories: *creation* and *alteration* errors.

For all *data input* errors,

IF error occurs when entering new data into the spreadsheet model
THEN ***insertion*** error
ELSE *NOT insertion* error (i.e. ***update*** error)

(i) Insertion Errors

These errors are produced while entering new data into the model. Typically these would take the form of typographical errors or omissions committed by the data entry users.

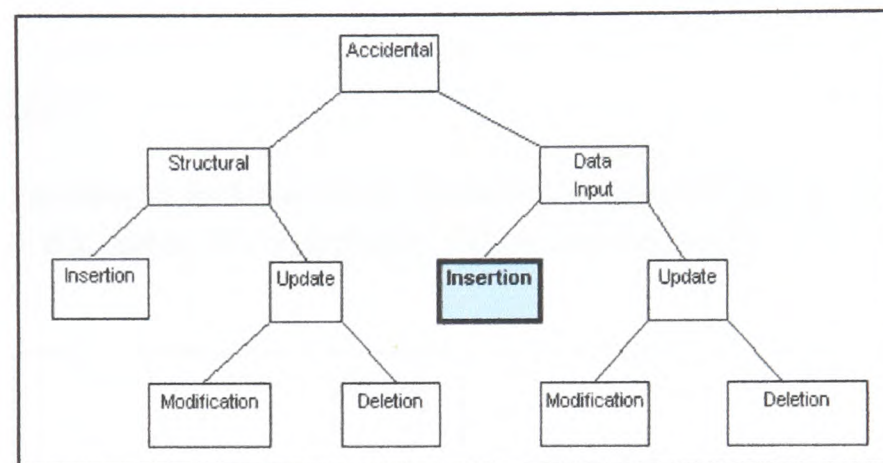


Figure 3.9: Insertion Errors (Data Input)

Example 1: Erroneous Data Input

Invalid or incorrect data is easily entered into the spreadsheet model because there are no data checks on entry. In other cases, the right data is put in the wrong cell. Wrong data can occur as the result of a data entry error or because of incorrect data from the data sources (Hayen-89). Freeman (Freeman-96) proposes the use of limit controls to deal with these errors. This would prevent the entry of figures outside tolerable ranges.

Example 2: Omissions

It is not uncommon for data entry operators to accidentally leave out certain inputs to the model. As a result, references to the corresponding input data in the workings/output section are omitted from the model.

An example of such an error was detected by KPMG (London), where there were insufficient Inputs to the model for extended periods of time. For instance, *increase in vehicle cost* is blank until 2001, even though the source of data from that date (from another worksheet) contains values for the earlier years (KPMG-98).

(ii) Update Errors

These errors are produced as a result of incorrectly updating existing data in the model. Update operations (apart from insertion) must either be modification or deletion. Therefore, update errors in this context can be divided into modification (or overwriting) errors and deletion errors.

For all *update* errors,

IF error occurs as a result of overwriting existing data

THEN **modification** error

ELSE *NOT* modification error (i.e. **deletion** error)

Modification Errors

These errors are produced as a result of changing existing data in the model. Typically these would take the form of typographical or overwriting errors committed by the data entry users.

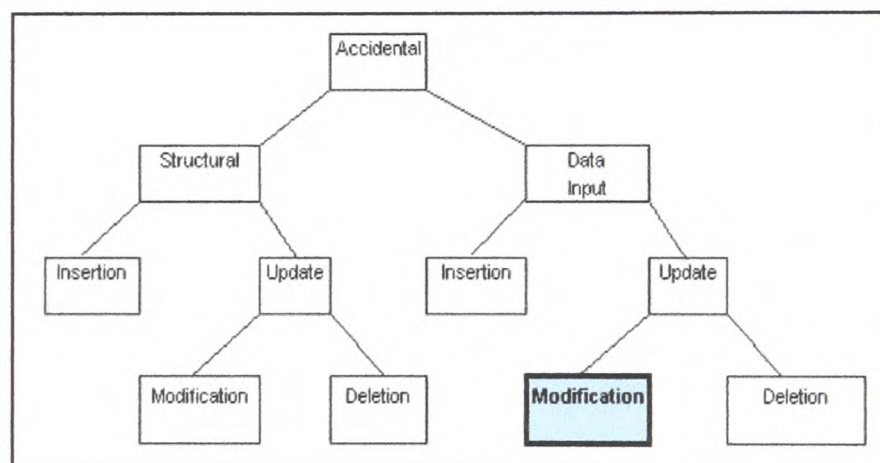


Figure 3.10: Modification Errors (Data Input)

Example 1: Overwriting of Data

A correct piece of data entered is overwritten with an incorrect input. This might be caused by an update being done in the wrong location of the spreadsheet model.

Deletion Errors

These errors, on the other hand, occur as a result of deleting or erasing previously entered data from the model. These errors are also caused by users responsible for data entry.

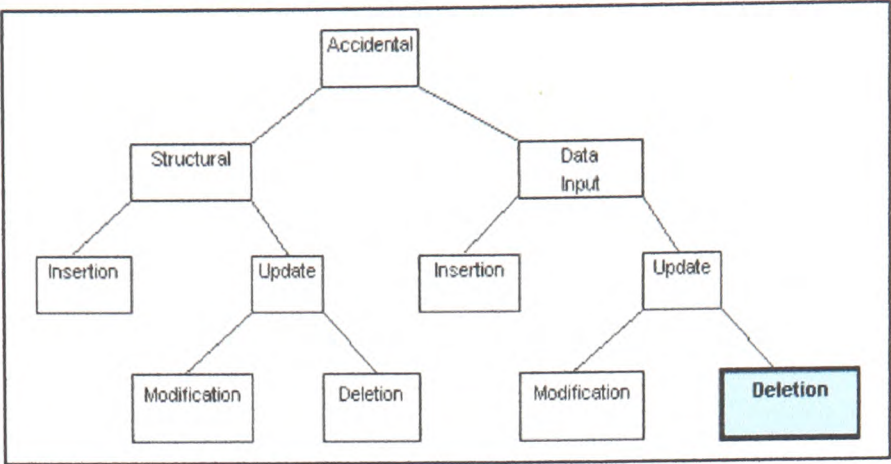


Figure 3.11: Deletion Errors (Data Input)

Example 1: Erasure of Data

A correct piece of input required by the model is simply deleted inadvertently. This is normally committed during data entry or update.

2. Reasoning Errors

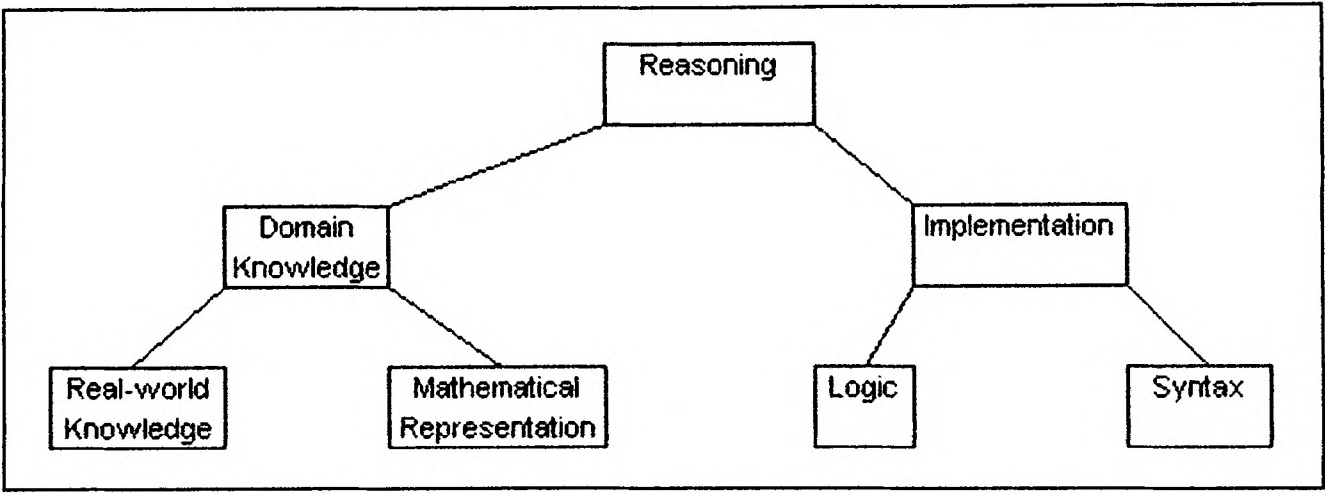


Figure 3.12: Reasoning Errors

Reasoning errors are mistakes in reasoning and therefore not accidental in nature. They are produced as a result of a lack of knowledge required to comprehend, analyse and accurately model the business function or problem in the form of an electronic spreadsheet model.

Reasoning errors can be split into two distinct categories based on an analysis of the precise cause of the errors, which in this case also involves a study of the cognisance of the model developer(s). The two classes of reasoning errors are *domain knowledge errors* and *implementation errors*. Conforming to the rules devised at the outset of the classification process, these are two mutually exclusive categories.

Any reasoning error which is not produced owing to inadequate comprehension of the underlying problem or function to be modelled on the spreadsheet, could only possibly have been caused by an incorrect implementation of the problem or function using the spreadsheet package. We shall refer to this as an *implementation error*.

Therefore, reasoning errors can be exclusively divided into two categories: *domain knowledge* and *implementation* errors. Research into the relative frequencies and real-life impact of the different types of reasoning errors have shown that implementation errors are far more common than domain knowledge errors. Domain knowledge errors are, however, generally more serious than implementation errors.

For all *reasoning* errors,

IF error occurs owing to a lack of understanding of the
underlying problem or function to be modelled

THEN ***domain knowledge*** error

ELSE *NOT domain knowledge* error (i.e. ***implementation*** error)

(a) Domain Knowledge Errors

Domain knowledge errors are specifically caused by inadequate awareness or knowledge required to identify, analyse and understand the business function or problem underlying the spreadsheet model. This knowledge is essential for modelling the problem and designing the corresponding conceptual or logical data model.

Domain knowledge errors, however, do not concern the specific features and capabilities of any particular spreadsheet package. The matrix of data and formulae that constitute the recognised spreadsheet model is an electronic representation of a business function in the real world.

This category of errors consists of two distinct classes, namely *real-world knowledge* and *mathematical representation* based errors. Any reasoning domain-knowledge error which occurs despite selection of the right algorithm must have been caused by a lack of understanding of how the algorithm is to be mathematically represented. It would therefore seem appropriate to term these sorts of errors as *mathematical representation* errors.

For all *domain-knowledge* errors,

IF error caused as a consequence of a lack of knowledge on the
underlying algorithm of a calculation or function

THEN ***real-world knowledge*** error

ELSE *NOT real-world knowledge* error
(i.e. ***mathematical representation*** error)

(i) Real-world Knowledge Errors

These errors involve creating a formula by selecting the wrong algorithm. Users may select an inappropriate template for a particular analysis or decision task. This may be due to a lack of accounting knowledge or intellectual modelling logic. For instance, a model that uses *straight-line depreciation* when an accelerated cost recovery system is desired (Hayen-89). When the underlying template or algorithm is wrong, the corresponding formula constructed would inevitably be erroneous as well. Increasing the number of developers should increase accounting knowledge and so reduce such errors (Panko-94).

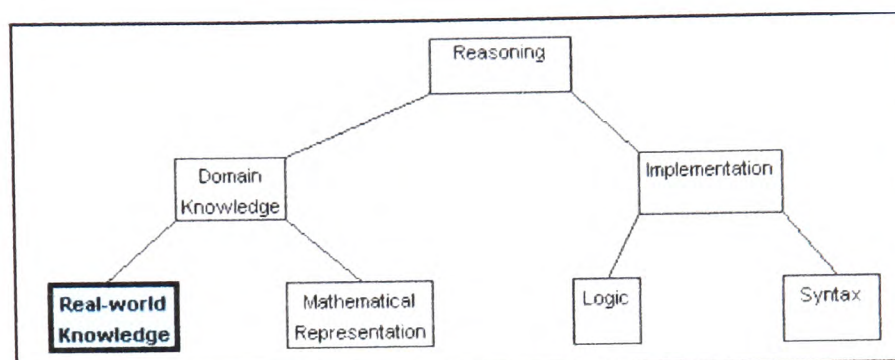


Figure 3.13: Real-world Knowledge Errors

Example 1: Exclusion of Factors from Formulae

A fairly common error in this category is the exclusion of important factors in a calculation. For instance, *bad debt provision* is excluded in an accounting calculation. Another example found was the failure to consider the impact of *losses b/fwd* (*brought forward*) in the calculation of tax charge.

Example 2: Inability to Distinguish Between Input Types

In laying out the spreadsheets, a *non-income* item is put in the column of *income* items, causing the value for *Total Income* to be erroneously large (Brown-87).

Example 3: Absence of Distinction Between Leap and Non-leap Years

This is a simple example of a real-world knowledge error whereby to calculate the daily figures for a particular leap year, the calculations divide by 365 instead of 366. KPMG, London (KPMG-98) found such errors in their clients' spreadsheets. For instance, *year 2000* is a leap year, but calculations divide by 365 not 366.

(ii) Mathematical Representation Errors

These errors involve constructing the wrong formula despite having selected the right algorithm. This is due to a lack of knowledge on how to represent a mathematically correct and accurate formula based on the correctly chosen algorithm. It is possible that a wrong or inappropriate equation is chosen to solve a problem.

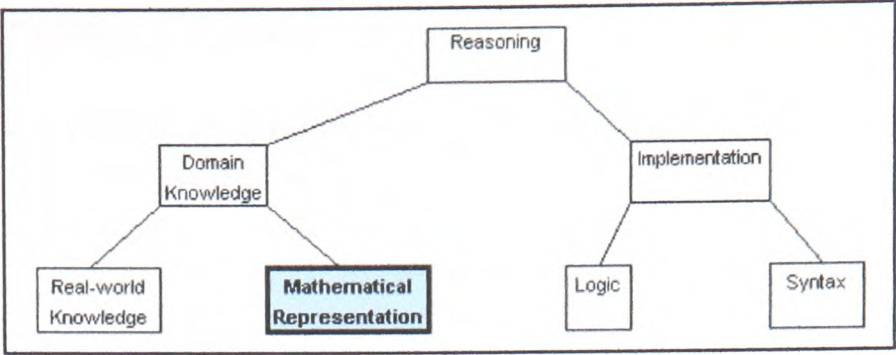


Figure 3.14: Mathematical Representation Errors

Example 1: The PERCENTAGE Error

This error occurs when the formula to calculate percentage is incorrectly written, due to a lack of knowledge of how to calculate a percentage or BODMAS (Brackets, Of, Division, Multiplication, Addition, Subtraction), by which the spreadsheet identifies precedence in calculations.

Based on Figure 3.15, an example of the error would be erroneous entries such as **B2/A2*100**, **B2*100/A2** or **B2*A2/100** in cell **C2** instead of **A2/B2*100** or **A2*100/B2**.

A		B		C	
Night Wages £		Total Wages £		Night Wages %	
1400.00		46940.00			

Figure 3.15: Percentage Error

Example 2: Incorrect Representation of an OVERALL AVERAGE Function

Based on Figure 3.16, the correct formula in **F9** is **=E9/B9** but the formula **=AVERAGE(F5:F8)** is entered instead (Chadwick-97,97a). Alternatively, the cell should be left blank if the overall average is not required. Adding the average figures together is meaningless. Although the model developer knew that an overall average was to be calculated, they incorrectly assumed that the sum of averages would give the overall average.

	A	B	C	D	E	F
1	Lazy Days Staff Budget Costs 1995-1996					
2		Staff	Basic	Overtime	Total	Average
3		Numbers	Wages £	Wages £	Wages £	Wage £
4						
5	Managers	1	17700	0		
6	Grade 1	3	45540	1400		
7	Grade 2	9	122340	2000		
8	Grade 3	12	102350	0		
9	Grand Totals	25	287930	3400		

Figure 3.16: Example to Illustrate Overall Average Error

Example 3: A Variable is Defined Twice

When developing a model, a forecast for a growth rate of **X%** is made. **X** is written into the equations that compute growth but is written in as a constant, e.g. **=[cell above] x 1.04**. Subsequently, the user might do a what-if analysis and write an equation such as **=[cell above] x [growth rate cell]**. During debugging, the two growth rates might be identical or similar. During use, they might be different (Stang-87).

(b) Implementation Errors

Implementation errors are produced due to a lack of knowledge or understanding of the full use of the functions and capabilities of the particular spreadsheet package in use, with an understanding of the spreadsheet principles, concepts, constructs, reserved words and syntax.

Implementation errors consist of *logic* and *syntax* errors. For any implementation error which does not occur as a result of a logic error, its cause has to be a syntax error. This is different from a typographical error, which is an accidental error. A syntax error, for instance, can be produced as a consequence of not knowing the spelling for reserved words in a formula.

For all *implementation* errors,

IF error is caused by a lack of comprehension of the features
 and functions of the spreadsheet package/language

THEN **logic** error

ELSE *NOT* logic error (i.e. **syntax** error)

(i) Logic Errors

Logic errors are errors caused by a lack of understanding of the functions and capabilities of a specific spreadsheet package, which enable the accurate representation of a solution or some part of it.

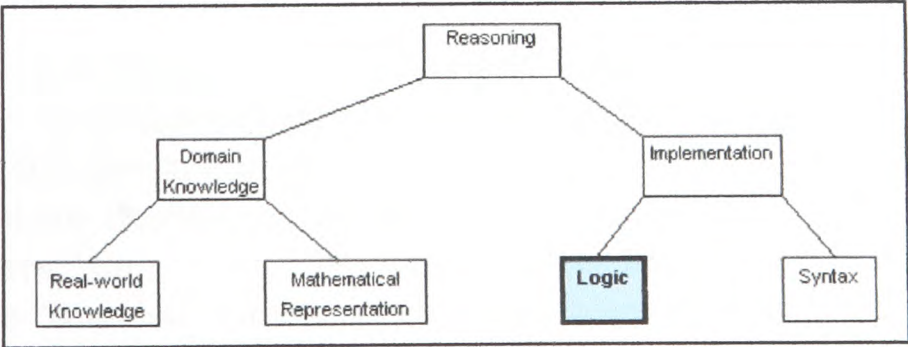


Figure 3.17: Logic Errors

Example 1: RELATIVE and ABSOLUTE Copy Problem

The relative copy causes cell references in a copied formula to alter row and column references relative to the original cell copied. People often make the false assumption that the software will automatically adapt the cell references wherever they happen to copy (Chadwick-97). On other occasions, the error is caused by the user copying a formula hidden underneath a cell value, thinking that they are copying the value from the cell (Brown-87). According to Hendry & Green (Hendry-94), novices experience difficulties in learning about relative and absolute cell references, a feature of all spreadsheets.

According to a report by KPMG Management Consulting, London (KPMG-98), in the calculation of vehicle leasing costs, the element of the formula that referred to *directors* had been lacking a \$ sign (used for *absolute copying* instead of the default *relative copying* of formulae), resulting in incorrect cell references when the formula was copied from the original cell. This resulted in an understatement of the costs (e.g. by \$432k in 2006).

Example 2: Rounding Error

When writing any spreadsheet the problem of rounding must be considered. Rounding can and should always be controlled. The best approach is to produce rounded numbers, and perform all operations on them, so that one works with numbers that are displayed, not with “hidden” values.

Based on *Figure 3.18* (Batson-91), it can be seen that the “formatted” column does not add up. The difference is small and can be attributed to rounding, but it affects the credibility of the model. It is therefore vital that a spreadsheet modeller understands what is occurring and takes measures to ensure that the rounding is controlled.

	Actual	Formatted	Rounded
A1	1.128431	1.13	1.13
A2	2.35625	2.36	2.36
A3	1.827994	1.83	1.83
=SUM(A1:A3)	5.312675	5.31	5.32

Figure 3.18: Rounding Error

In example shown in *Figure 3.18*, the “actual” column refers to how the number is stored within the spreadsheet (often up to 15 significant figures). The “formatted” column shows what appears on the screen if the column is formatted to two decimal places; the numbers themselves, however, are still held in the spreadsheet to 15 significant figures, and it is “hidden” values which are used in subsequent calculations. The “rounded” column shows what happens when each value is rounded so that the spreadsheet holds the values to two decimal places only, in which case, as shown, the column adds up correctly (Batson-91).

Stang (Stang-87) also provides an example of a rounding error. If users format to one digit to the right of the decimal, and then enter values having greater precision, the spreadsheet will round off the numbers. Thus 1.44 will round off to 1.4; the sum of 1.44 and 1.44 will round to 2.9 from 2.88. Such additions would appear to be incorrect.

Example 3: Circular Reference

Circular references in formulae often indicate that there is an error in the logic of the model and should therefore be avoided. Such references should be eliminated at the specification stage (Batson-91). This error frequently occurs in totals where the formula uses its own value in its calculation. This error will give a run-time error message and so probably occurs infrequently (Chadwick-97). A common example of a circular reference arises when calculating bank overdraft interest (Batson-86,91). This is shown in *Figure 3.19 (a)* (Batson-91).

<i>Cashflow</i>	£
Opening bank balance (overdrawn)	(x)
Add: Receipts	x
Less: Payments	(x)
Less: Overdraft interest based on closing balance	(x)
Closing bank balance	(x)

Figure 3.19 (a): Circular Reference

Each time the spreadsheet is recalculated the overdraft interest will change and update the closing bank balance ad infinitum. The error can be corrected by removing the circular reference. The correct way is shown in *Figure 3.19 (b)* (Batson-91).

<i>Cashflow</i>	£
Opening bank balance (overdrawn)	(x)
Add: Receipts	x
Less: Payments	(x)
Balance before overdraft interest	(x)
Less: Overdraft interest on balance before interest	(x)
Closing bank balance	(x)

Figure 3.19 (b): Circular Reference Resolved

According to Ditlea (Ditlea-87), a circular reference was adding the *11-month total* for a *region* to itself. As a result, the spreadsheet was mistakenly doubling a \$10 million figure every time it recalculated.

Example 4: Row is Added to the Model but not the “Bottom Line” Total

This error has been pointed out by several authors (Ayalew-00, Butler-97, Stang-87, Ditlea-87). The modeller has written an equation to find column totals, writing the equation in row seven. Data are to be entered below. The equation is written `=SUM(B8:B99)`. It works fine until a user adds data in row **100**. Because this row is beyond the range of the equation, the data is not included in the addition (Stang-87).

To quote a real-life example (Ditlea-87), the controller of James A. Cummings, Inc., a Florida construction company, was putting together a Symphony spreadsheet model to bid on a \$3 million office complex for a local utility. When he realised he hadn’t included a line for \$254,000 in overhead costs, he inserted it at the top of a column of figures, failing to notice that this entry fell outside the range of numbers to be added by the `@SUM` (`=SUM` in MS Excel) function in his formula. This undetected mistake resulted in a profit-losing “winning” bid (Ditlea-87).

(ii) Syntax Errors

Syntax errors are errors caused by a lack of precise understanding of the constructs, reserved words and syntax of a specific spreadsheet package, used to write functions and formulae.

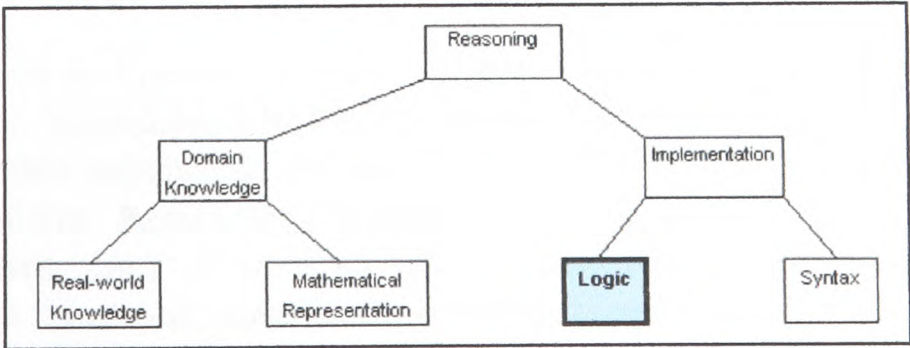


Figure 3.20: Syntax Errors

Example 1: A Keyword Within a Formula is Misspelled

A keyword within a formula is misspelled causing an error, e.g. `=AVG(...)` instead of `=AVERAGE(...)`. This error can be detected easily as the spreadsheet package would instantly respond with an indication of an error.

3.5.2 Qualitative Errors

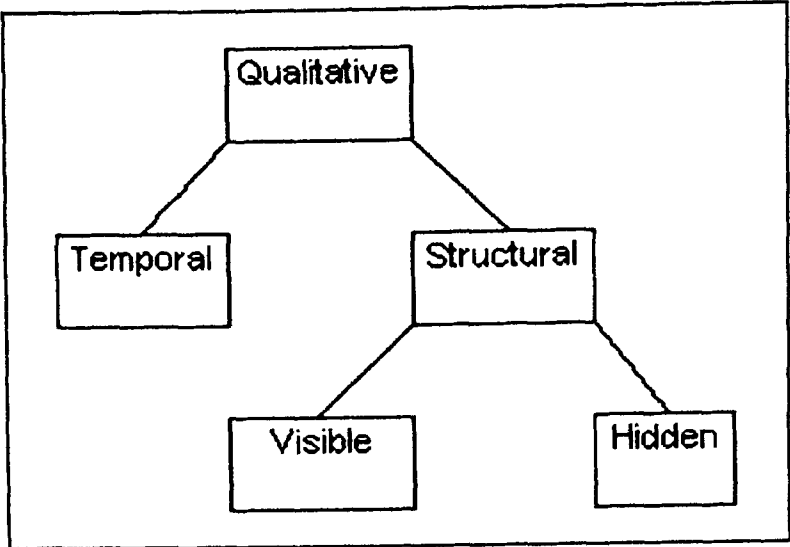


Figure 3.21: Qualitative Errors

Qualitative errors are errors that do not immediately produce incorrect numeric values but degrade the quality of the model. The model also becomes more prone to misinterpretation on the part of the user. As a result, it also becomes more difficult to update and maintain the model. A more detailed investigation into qualitative errors reveals that they can be generally divided into two different types, namely, *temporal errors* and *structural errors*.

This dichotomy is obtained mainly based on an analysis of the three views of an information system. It has been previously established that a spreadsheet system is also a type of information system.

The three views of an information system: *data*, *processing* and *behaviour*, are also applicable to spreadsheet models. Within the context of spreadsheet models, the *processing* view of a model is the network of formulae used to perform calculations on data and produce the computation results. This is also the schema of the model. The *data* view represents the various input data required for the calculations of formulae. The *processing* and *data* views are rather snapshot in nature. The *behavioural* or *temporal* view represents the effects of time and real world events on the spreadsheet model. Unlike the *data* and *processing* views, this is a dynamic view of the spreadsheet model.

A qualitative error which is not temporal in nature can be considered a structural error. This encompasses all forms of non-temporal factors or structural flaws which degrade the quality of the spreadsheet model. The structural aspect of the model in this context represents the binding of the model schema (formula network) and data.

For all *qualitative* errors,

IF error is caused by an elapse of time, which invalidates data
THEN ***temporal*** error
ELSE ***NOT temporal*** error (i.e. ***structural*** error)

1. Temporal Errors

Temporal errors are qualitative errors which invalidate data (and possibly formulae) with the passage of time. As a result, the model ceases to be reliable. Typically these errors are caused by failure or delays in updating the spreadsheet model to reflect current circumstances due to the effects of time and real world events on the model.

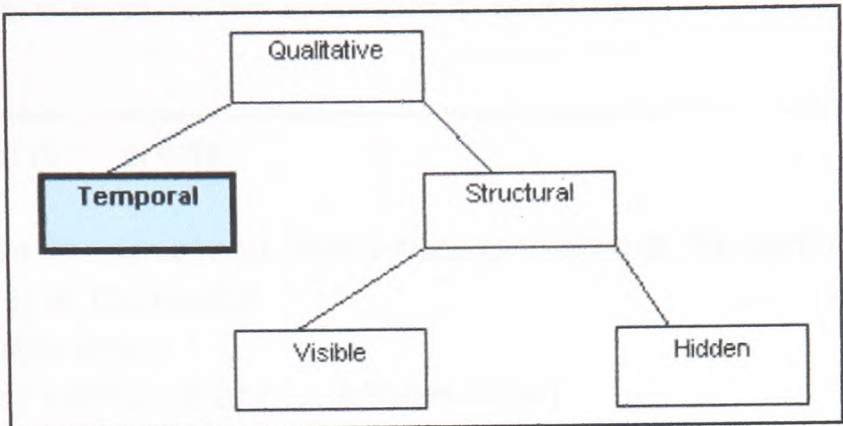


Figure 3.22: Temporal Errors

Example 1: Qualitative Error Caused by the Referencing of Non-current Data

Produced as a result of referencing a piece of data that has become invalid due to time lapse. In the example given below (Figure 3.23), this piece of data is the exchange rate from Pounds Sterling (£) to Ringgit Malaysia (RM) contained in cell F2. If the exchange rate undergoes acute fluctuations and the changes are not reflected in cell F2, the calculation in cell A8 produces a value that is invalid. This is a qualitative error and any decision made based on this value would be unreliable (Rajalingham-98).

	A	B	C	D	E	F
1		Tea(£)	Milk(£)	Coffee(£)		Exchange Rate (£ to RM)
2	1st Quarter	450	560	467		7.3
3	2nd Quarter	904	900	352		
4	3rd Quarter	872	800	233		
5	4th Quarter	123	234	901		
6						
7	Total Sale of Tea (RM)					
8	31404.6					

Figure 3.23: Example to Illustrate the Referencing of Non-current Data

Example 2: A Spreadsheet Model is Fully Debugged but an Earlier, Bug-laden Version is Used

Unless tight distribution controls are practised, it is not unusual to have 15 different versions of a budget model, with names like *budget*, *budget10*, *mybudget*, *doaksbud*, etc. Each of these variations has some chance of being revised differently by some user (Stang-87). It is possible that the most current version has been fully debugged and saved. However, a previous version with flaws is used to make important decisions.

2. **Structural Errors/Flaws**

Structural errors in this context can be defined as non-temporal qualitative errors or flaws produced as a result of poor design or layout of model structures and data. Based on the physical manifestation of these errors, they can be divided into two categories: *visible errors* and *hidden errors*. Based on an inspection of numerous qualitative structural errors, it can be concluded that any structural flaw which is not visible at the surface level of the spreadsheet model has to be hidden at the deep or formula level.

For all *structural* errors,

IF error is a structural flaw which is visible at the surface level of the model

THEN ***visible*** error

ELSE *NOT visible* error (i.e. ***hidden*** error)

(a) **Visible Errors/Flaws**

Visible qualitative structural errors are structural flaws which are visible at the surface level of the model. The detection of these errors does not require any examination of the deep or formula level of the spreadsheet model. These errors normally take the form of semantic errors which make the models more prone to misreading or misinterpretation. This could be due to the ambiguous nature of the relevant elements of the model.

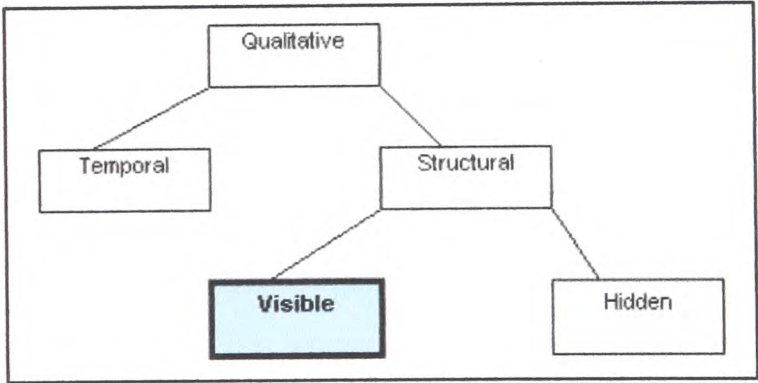


Figure 3.24: Visible Structural Errors

Example 1: Formatting Error

Formatting errors are semantic errors that occur due to lack of uniformity in the formatting of similar data. This could lead to an incorrect interpretation of their values. A common qualitative error is where the cell format is specified as *general* on the spreadsheet. Consequently, the figures have varying decimal places and make it difficult to identify a number that is incorrect, by a magnitude of 10, 100, etc. This is shown in *Figure 3.25*. The value in cell **G10** is greater than the value in cell **G9**. However, at a quick glance, it may seem as if the value in cell **G9** (**102350.25**) is greater than the value in cell **G10** (**291331.3**) due to the inconsistent use of decimal places.

	C	D	E	F	G	H	
		Number of Staff	Day Wages £	Night Wages £	Total Wages £	Average Wage £	
Grade 1		1	17700.5	0	17700.5	17700.5	5
Grade 2		3	45540	1400.55	46940.55	15646.85	6
Grade 3		9	122340	2000	124340	13815.56	7
Grade 4		12	102350.25	0	102350.25	8529.19	8
Grand Total		25	287930.75	3400.55	291331.3	11653.25	9
							10

Figure 3.25: Formatting Error

Example 2: Range Names are Misleading or Incorrectly Used

A range is named as **COSG**. This is generally assumed to refer to a cell containing *cost of goods sold* and that COSG has been calculated as they would have done. However, the designer may have been referring to *cost of operations in the general services division* (Stang-87).

(b) Hidden Errors/Flaws

Hidden qualitative structural errors, on the contrary, are structural flaws which are not visible at the surface level of the model and therefore require examination of the deep or formula level of the spreadsheet model. These errors normally take the form of complicated, confusing or inappropriate construction of formulae. Such flaws can make the model difficult to maintain and prone to inconsistencies or update anomalies. As a consequence, the integrity of the model can be steadily eroded.

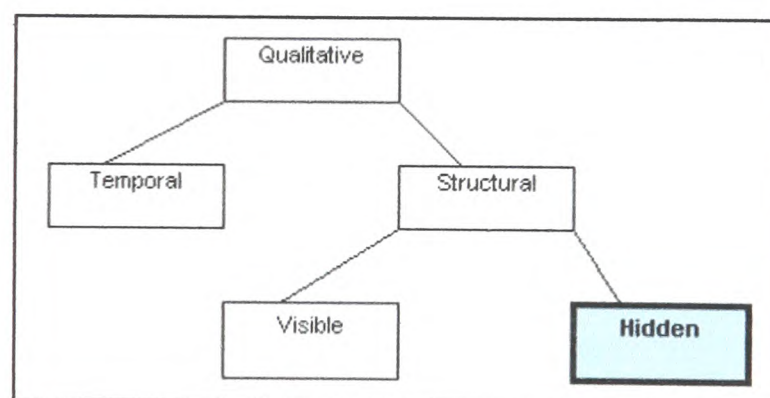


Figure 3.26: Hidden Structural Errors

Example 1: Hard-coding

A fixed value is used when a variable (cell reference) should be used instead. In other words, cells that should have calculations, contain hard-coded input numbers instead. For instance, the hard-coding for leap year adjustment (KPMG-97) or *net * 17.5%* instead of *net * a named variable or range "VAT rate"* (Butler-00a). Batson (Batson-91) also points out that some numbers, which at first sight appear to be constants, are often in fact variables, for instance, the *rate of inflation* or the *percentage value for employees' pension contributions*.

Example 2: Complexity of Formulae

The calculation of distances between transit and non-neighbouring areas is apparently based on a total area of **78,864 sqkm** divided into **163** transit or **15** non-neighbouring areas. The distance is taken as the diameter of a circle of such an area. The formula could have been written more concisely using the **PI** function (KPMG-98). Stang (Stang-87) suggests that any equation longer than 80 characters uses logic that is difficult to follow (Stang-87). Ray Butler (Butler-97) identified that addition and subtraction of numbers within single spreadsheet cells were done without thought for the audit trail or auditability.

3.6 Summary

There has been inadequate research and examination of specific errors in spreadsheet modelling. An analysis and classification of specific types of spreadsheet errors has been carried out as a precursor to the development of approaches to effectively address the problem. This chapter has presented a more comprehensive taxonomy of spreadsheet errors than ever presented or published before, based on a rational taxonomic scheme.

The main reasons for developing a classification of spreadsheet errors are as follows:

- It is a methodical approach to problem analysis.
- It has greater potential for improving comprehensive testing of a spreadsheet development methodology.
- It provides a deeper insight into the nature and characteristics of the errors.

It is evident that there are no standard methods for producing a taxonomy. The conventional generic process of classification widely used in *zoology* and *botany* has been adopted in producing the spreadsheet error taxonomy.

The *binary* structure for an error taxonomy has been found to be more beneficial compared to the previously adopted *bushy* method. The current taxonomy of spreadsheet errors is based on a binary approach that uses *dichotomies* or *IF-THEN-ELSE* rules to classify errors. Therefore, the taxonomy can be expressed in a structured form. A summary of the entire classification in this form is presented in *Figure 3.27*.

```

IF numerical error causing incorrect bottom-line value
THEN quantitative error
  IF error is caused by negligence or carelessness
  THEN accidental error
    IF error is caused by the model developer
    THEN structural error
      IF error is produced when creating the structural
        aspects of the spreadsheet model
      THEN insertion error
      ELSE NOT insertion error (ie update error)
        IF error occurs as a result of altering the structural state of the model
        THEN modification error
        ELSE NOT modification error (ie deletion error)
      ELSE NOT structural error (ie data input error)
        IF error occurs when entering new data into the spreadsheet model
        THEN insertion error
        ELSE NOT insertion error (ie update error)
          IF error occurs as a result of overwriting existing data
          THEN modification error
          ELSE NOT modification error (ie deletion error)
        ELSE NOT accidental error (ie reasoning error)
          IF error occurs owing to a lack of understanding
            of the underlying problem or function to be modelled
          THEN domain knowledge error
            IF error caused as a consequence of a lack of
              knowledge on the underlying algorithm of a calculation or function
            THEN real-world knowledge error
            ELSE NOT real-world knowledge error
              (ie mathematical representation error)
            ELSE NOT domain knowledge error (ie implementation error)
              IF error is caused by a lack of comprehension of the features
                and functions of the spreadsheet package/language
              THEN logic error
              ELSE NOT logic error (ie syntax error)
            ELSE NOT quantitative error (ie qualitative error)
              IF error is caused by an elapse of time, which invalidates data
              THEN temporal error
              ELSE NOT temporal error (ie structural error)
                IF error is a structural flaw which is visible at the surface level of the model
                THEN visible error
                ELSE NOT visible error (ie hidden error)

```

Figure 3.27: Entire Classification in *Structured Form*

CHAPTER 4

PAST WORK AND EXISTING DEVELOPMENTS

4.1 Introduction

Chapter 2 mainly presented an insight into the problem of user-generated spreadsheet errors in terms of their frequency and impact, while the previous chapter, *Chapter 3*, concentrated on the analysis and classification of specific user-generated spreadsheet errors.

This chapter presents a discussion of a spectrum of existing tools and techniques for integrity control of spreadsheet models and the different life cycles and methodologies proposed for their development. An analysis of the effectiveness and limitations of the tools and techniques is also conducted along with a critical evaluation of the various life cycles and methodologies proposed for the development of spreadsheet models.

Numerous *software tools* have been developed and marketed for the auditing and integrity control of spreadsheet models. Various *techniques* have also been proposed to enhance the quality the models. Apart from these tools and techniques, over the years, several *life cycles* and *methodologies* for the development of spreadsheet models have been proposed and presented in a host of publications.

4.2 Existing Tools and Techniques

4.2.1 Tools

Software audit tools have been around for almost as long as spreadsheets themselves (Butler-97). The following are among the most popular computer-based tools that have been developed to help combat the problem of spreadsheet errors. Not all of them are in widespread use today.

- *Spreadsheet Auditor*
- *Cambridge Spreadsheet Analyst*
- *Microsoft Excel's Built-in Auditing Functions*
- *The Excel Auditor*
- *Spreadsheet Professional Audit Tool for Microsoft Excel*
- *Spreadsheet Detective*
- *The Operis Analysis Kit (OAK)*
- *Spreadsheet Auditing for Customs and Excise (SpACE)*

The objective of this section is to only briefly introduce some of the main tools that have been available. As such a detailed description of each tool is not given.

Spreadsheet Auditor

In the mid-1980s *Spreadsheet Auditor* was marketed as an aid to auditing the then prevalent *Lotus 1-2-3* .wks and .wkl file formats. The development of the tool has ceased (Butler-97).

Cambridge Spreadsheet Analyst

Cambridge Spreadsheet Analyst was also developed as an aid to auditing *Lotus 1-2-3* .wks and .wkl file formats in the same period. The development of this tool was also terminated (Butler-97). The *Cambridge Spreadsheet Analyst* provided two new ways of viewing a spreadsheet: a citation view, which allows one to see cell relations one at a time, and a map view, which shows a condensed version of the spreadsheet revealing similar adjacent formulae (Saariluoma-91).

Microsoft Excel's Built-in Auditing Functions

These are Microsoft Excel's own built in auditing functions. These built-in functions are included in Excel as a standard (Nixon-01). The auditing tool enables the user to easily trace the **precedents** or **dependants** of any cell. The precedents of a cell are the cells **referenced** by it while the dependants of a cell are the cells that **reference** it. When tracing the precedents of a cell, an arrow points from each precedent cell to the dependant cell. This is illustrated in *Figure 4.1* (Chadwick-97). It can be seen that the precedents of cell **E13** are cells **E8** and **E9**. The audit tool also offers a facility for the user to attach a note or description to a cell.

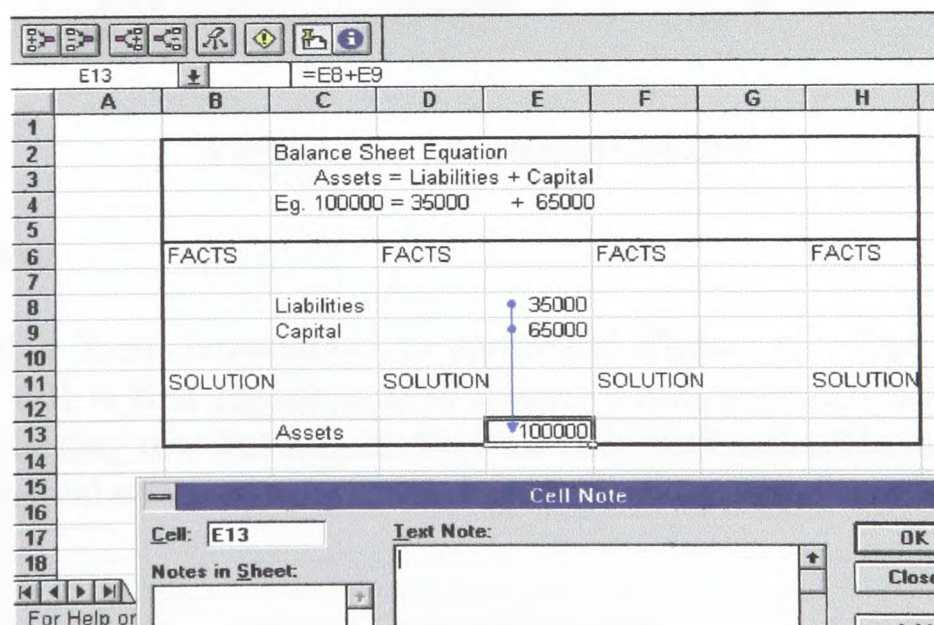


Figure 4.1: Identifying *Precedents*

The Excel Auditor

This software is produced by BYG Software and is an add-in for Excel and, despite its name, provides many functions outside the usual scope of auditing software. The Excel Auditor provides two primary and two secondary auditing tools. The primary

tools being the *Audit Map*, which provides a traditional audit map of a worksheet, and the *Worksheet Cell Analyser*, which documents the contents of a worksheet's cells. Both of these tools produce a report on a separate workbook. The secondary tools are the *Cell Roots* and *Circular Reference Analyst functions* (Nixon-01).

Spreadsheet Professional for MS Excel (by Spreadsheet Innovations)

Spreadsheet Professional is used by all of the largest UK accounting firms, and by HM Customs & Excise (Butler-97). This tool, by *Spreadsheet Innovations*, which is an add-in for *Microsoft Excel*, has served as a rather useful auditing tool. It is primarily used to test and document spreadsheet files. It has various functions to help detect errors in the spreadsheet model. Among the significant functions are the *calculation checker* and the *cell translation* facility.

Calculation Checker

This function enables the user to view the contents, potential error and precedents of a formula cell. Each precedent (cell or range) is defined in terms of its location (address), corresponding row heading and value.

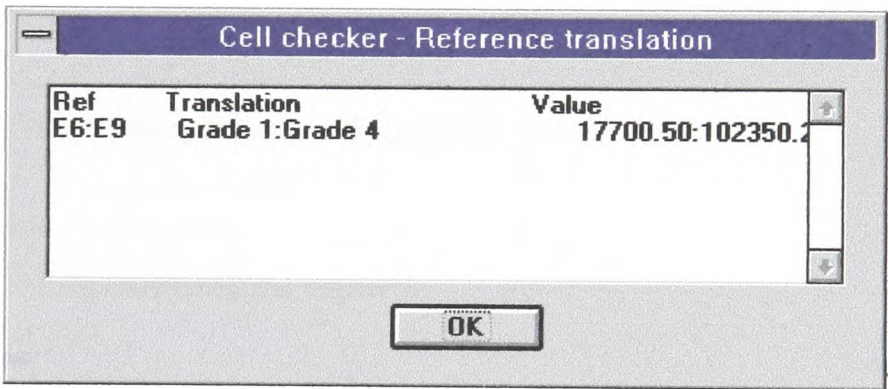


Figure 4.2: Calculation Checker

Cell Translation

An example of a '*Cell Translation*' is shown in *Figure 4.3*. The structure of the formula in the cell is first represented in a form where each cell address is replaced with its corresponding row heading. In the next row, each cell address is replaced with its corresponding value.

Sheet1!E10	= SUM (E6:E9)
Grade Total	= SUM (Grade 1:Grade 4)
287930.75	= SUM (17700.50:102350.25)	

Figure 4.3: Cell Translation

Spreadsheet Detective

The *Spreadsheet Detective* is also an Excel add-in, produced by Southern Cross Software. Nixon (Nixon-01) states that there are two fundamental ways in which the software attempts to assist users in the auditing of spreadsheets. The first is the identification of formula schema while the second is the listing of potential problems such as references to non-numeric cells or unprotected schema.

The Spreadsheet Detective's key patented features are the *shading*, which provides a proper formula map over the existing cells, and the *AutoNames*. While the Spreadsheet Detective's *AutoNames* make the use of Named Ranges largely redundant, this is a useful feature for people who do still use Named Ranges. Spreadsheet Detective can produce a report of all Named Ranges, including for which sheet they have been defined and the range to which they refer. They are integrated with the annotations or Formula report, and the report correlates them with cell labels or AutoNames.

The Detective tends to produce reports of dubious formulae rather than selecting cells, with the exception of some of its year 2000 analysis. The Spreadsheet Detective can also compare different versions of a spreadsheet, which is very important to verify that only specific changes have been made. The Detective can align both rows and columns.

The Spreadsheet Detective's advanced features are as follows:

- AutoNames
- full annotations
- useful Named range definition reports
- Year 2000 analysis
- 3D formula indication
- workbook precedent report

The Operis Analysis Kit (OAK)

OAK provides the basic map and formula report required for any spreadsheet audit. This tool has been produced by Operis Business Engineering Limited, London and takes the form of an add-in for Microsoft Excel. OAK provides the following features:

- Basic Formula Map
- Workbook Summaries, Formula Report
- Named Range analysis
- Selection of different types of cells
- Spreadsheet Comparison
- Insert/Delete Row/Column
- Development History spreadsheet

The formula map essentially copies the original spreadsheet, and then replaces all formulae in the original spreadsheet with symbols to indicate whether they are copies of other formulae above or to their left. OAK can also shade the original spreadsheet to show which cells have formulae. This is a much better option than producing a

separate map because one can see the shading and the formula results at the same time, as well as being able to manipulate the formulae. However, the OAK shading gives no indication of how formulae have been copied. It also uses cell colour rather than patterns, which can corrupt existing formatting.

The workbook summaries provide a list of all the worksheets in a book and some basic statistics about the size of each worksheet. More importantly, it provides a list of all unique formulae. OAK provides some excellent facilities to be able to rename Named Ranges and automatically update the formulae that use them. OAK can also produce a report of all Named Ranges, including for which sheet they have been defined and the range to which they refer. OAK can also automatically select cells based on different criteria. For example, it can select functions with hardwired constants or that refer to blank cells. It can also compare different versions of a spreadsheet.

Spreadsheet Auditing for Customs and Excise (SpACE)

SpACE has been developed by the HM Customs and Excise, United Kingdom. It is mainly used by VAT inspectors in auditing client spreadsheets. However, it is also available to the public. SpACE works by using a combination of search facilities, overlaid mapping options and the identification of unique formula, to highlight potential errors in a spreadsheet. It also has more in-depth auditing functions such as the ability to check lists of data for duplicates (Nixon-01).

4.2.2 Techniques

Apart from software tools developed to help control the integrity of spreadsheet models, various techniques have also been proposed. The objective of these techniques is to enhance the quality of spreadsheet models. The following list captures a selection of significant techniques described in spreadsheet literature:

- Benham's (Benham-93) Structured Techniques for Spreadsheet Development
- Kee's (Kee-88) Standard Spreadsheet Design Format
- Ronen et al's (Ronen-89) Recommended Spreadsheet Structure
- Ronen et al's (Ronen-89) *Spreadsheet Flow Diagrams (SFD)*

Benham's (Benham-93) Structured Techniques for Spreadsheet Development

Benham (Benham-93) proposes the arrangement of the spreadsheet into blocks or sections along the spreadsheet's diagonal. As a minimum, the spreadsheet should have the following sections:

- Introductory Section
- Data and Assumption Section
- Model Section (work performed by the spreadsheet)
- Analysis Section (required outcomes or results)
- Macro Section

The block diagonal structure provides a modular character to the spreadsheet. Each section is effectively isolated from row or column insertions or deletions in any other sections.

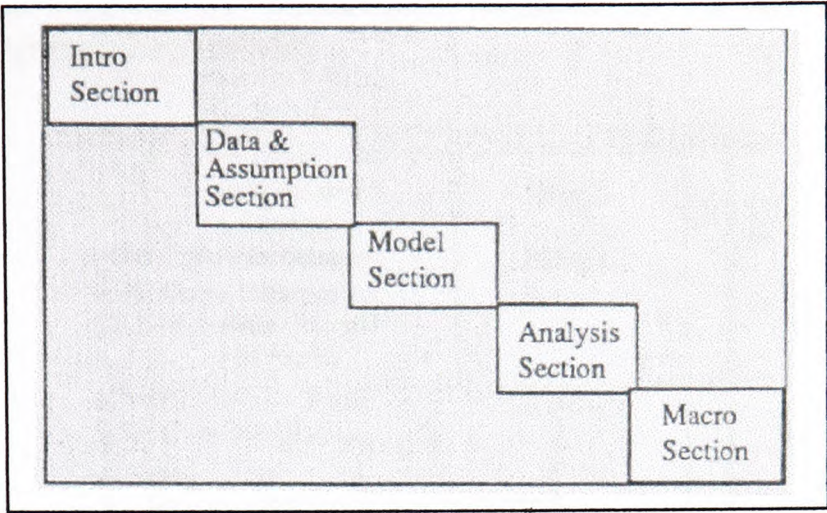


Figure 4.4: Benham’s Block Diagonal Structure

Kee’s (Kee-88) Standard Spreadsheet Design Format

Kee’s method involves the creation of a common spreadsheet format by first dividing a worksheet into rectangular blocks, where each block is used to perform one spreadsheet function. This proposed standard design format is presented in *Figure 4.5*.

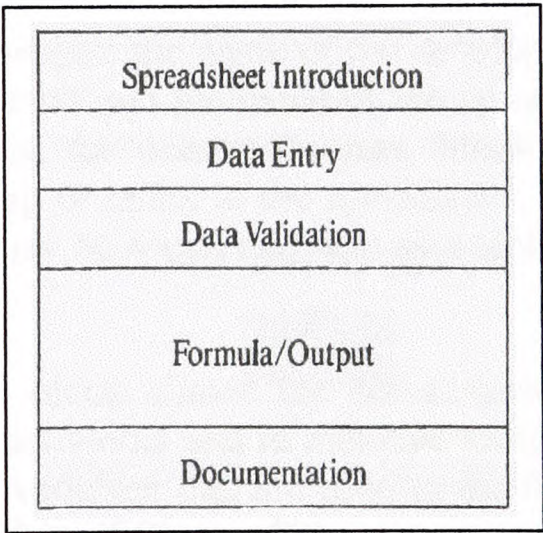


Figure 4.5: Kee’s Standard Spreadsheet Format

The **introduction** section in *Figure 4.5* is used to describe the spreadsheet. The information helps users to identify a template, the task(s) for which it was designed, and the instructions necessary for its use. In the **data entry** area, descriptive labels are placed adjacent to cells where data will be entered for subsequent processing. The design in *Figure 4.5* separates data and parameters from the formula/output section. Next to the data entry area is a **data validation** section. This is used to perform analytical tests on the input data to detect potential input errors. In the **formula/output** area, a series of spreadsheet formulae is used to process data into information. The last section of the spreadsheet is a **documentation** area. Here a detailed description of the variables, parameters, formulae and assumptions used in the spreadsheet is provided (Kee-88).

Ronen et al's (Ronen-89) Recommended Spreadsheet Structure

Figure 4.6 presents Ronen et al's recommended structure for a spreadsheet. The purpose of the structure is to separate parts of a spreadsheet into blocks to reduce the potential for errors. Figure 4.6 contains a number of blocks which, when taken together, form the spreadsheet model.

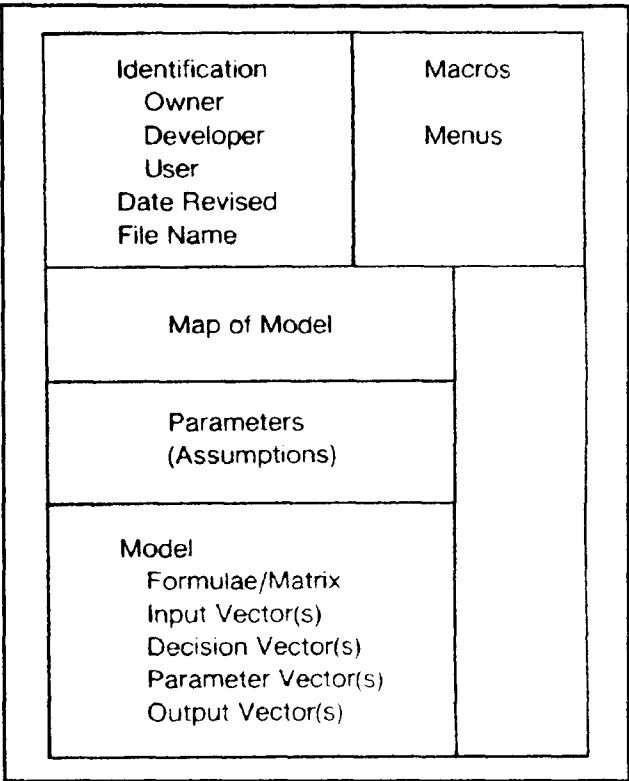


Figure 4.6: Ronen et al's Spreadsheet Structure

The **identification** block presents the name of the developer, user, and model. It also contains a list of revision dates and the name of the spreadsheet file. To the right of the identification block is the **macros/menus block**. Immediately below the identification block is a **map** or index to the spreadsheet. It contains a description of where the various blocks may be found and acts as a table of contents for the model (Ronen-89).

The large **documentation** block allows the spreadsheet developer to describe in general terms how the model works and to annotate various rows in the model. The **parameter** block contains variables that are used in the formulae. The final block in the spreadsheet is the **model** itself (Ronen-89).

Ronen et al's (Ronen-89) Spreadsheet Flow Diagrams (SFD)

Ronen et al (Ronen-89) endeavour to apply the notion of *Data Flow Modelling* in spreadsheet development. This is due to its popularity in traditional systems analysis and design as a way to promote structured, top-down design and to reduce complexity. The proposed *Spreadsheet Flow Diagrams (SFD)* are used for the same purpose.

Figure 4.7 shows the basic symbols of Ronen et al's *SFD*. A simple rectangle is used to represent input vectors, output vectors, decision vectors, and parameters. According

to Ronen et al (Ronen-89), the advantage of using a structured notation for spreadsheets is the same as the merits of such notations used in *Data Flow Diagrams*.

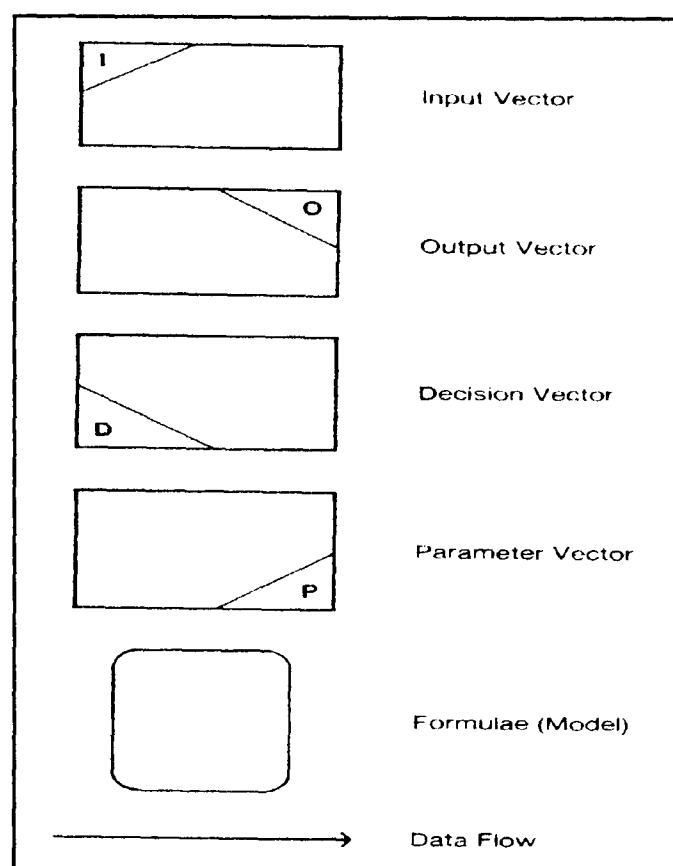


Figure 4.7: Notations of Ronen et al's *SFD*

4.2.3 Effectiveness and Limitations of the Tools and Techniques

This section presents a discussion of the effectiveness and limitations of the eight tools and four techniques described in *Sub-sections 4.2.1* and *4.2.2*.

All the software tools described in *Section 4.2.1* are primarily aimed at facilitating auditing and error detection in spreadsheet models. Though these developments have to an extent reduced errors in spreadsheets, they have not been entirely successful as the phenomenon still persists. The main reason for the lack of success of the tools is the fact that they concentrate on detecting errors rather than preventing the incidence of the errors.

There are two criteria that can be used to assess the effectiveness of a software tool:

- Its capacity to detect existing errors
- Its capacity to caution the user on potential errors, flaws and problems

The *Spreadsheet Auditor* and *Cambridge Spreadsheet Analyst* have become obsolete following the termination of their development. These tools were created to help audit *Lotus 1-2-3* file formats and would not be very useful today with the more widespread use of *MS Excel* in the Windows platform. As the tools were produced in the mid-80s, they lack the more advanced and sophisticated features of the other more recently developed tools.

The *Spreadsheet Detective* is the most effective among the eight tools assessed. It possesses an excellent capacity to detect existing errors and notify users of any

potential flaws or errors. The *Spreadsheet Detective* provides an overlay to a worksheet with different types and colours of shading along with text descriptions. This easily reveals errors such as overwritten formulae. It also produces reports of *named ranges* and *formulae*, which can be used to effectively identify dubious and potentially erroneous cells.

Among the tools evaluated (apart from *Spreadsheet Auditor* and *Cambridge Spreadsheet Analyst*), the *Excel Auditor* is the least satisfactory. Though the *Excel Auditor* offers various functions outside the usual scope of auditing software, it is neither as effective as the other tools in detecting existing errors nor identifying any potentially unsafe or problematic cells. A major limitation of the *Excel Auditor* is that it performs a laborious cell-by-cell inspection rather than using more visual techniques. However, it can be useful as a documentation tool.

Spreadsheet Auditing for Customs and Excise (SpACE) can be regarded as a very good tool for both detecting existing errors and identifying potentially problematic cells in a spreadsheet. Its effectiveness in accomplishing these is comparable to that of the *Spreadsheet Detective*. SpACE has very good auditing tools but lacks the ability to produce a formula description in natural language like the *Spreadsheet Detective*.

The *Operis Analysis Kit (OAK)* is highly effective in the detection of existing errors in a spreadsheet model. This is attributable to OAK's ability to produce a basic formula map and shading of the original spreadsheet. A disadvantage is that this can sometimes corrupt existing formatting. OAK does not fare as well as the *Spreadsheet Detective* or SpACE on the second criterion. OAK is relatively less effective in identifying potential problems, such as unprotected cells.

Spreadsheet Professional is satisfactory in both the detection of existing errors and the identification of potentially unsafe or erroneous cells. The detection of existing errors is mainly done with the help of the *Cell Translation* feature which enables quick verification of the logic of formulae. The *Spreadsheet Professional* also provides useful reports of potential problems or flaws such as unused operands of a formula, hard-coded formulae and the referencing of blank or non-numeric cells. The main limitation of this tool is that it does not offer more advanced features for error detection like the *Spreadsheet Detective*, SpACE or OAK.

Microsoft Excel's Built-in Auditing Functions do have a reasonable capacity to detect existing errors in a spreadsheet by tracing the *precedents* and *dependants* of a cell. However, the functions are not effective in identifying potential errors or potentially unsafe cells, for instance, unprotected and hard-coded formulae. Like the *Excel Auditor*, MS Excel's built-in auditing functions also have the disadvantage of concentrating on cell-by-cell inspection.

While most of the software tools focus on error detection, the techniques described in Section 4.2.2 represent efforts to reduce, if not prevent, the occurrence of errors. The advent of these techniques indicate an increased awareness of the importance of adopting more structured or systematic approaches to the development of spreadsheet models. All the techniques discussed in Section 4.2.2 have their advantages and disadvantages or limitations.

There is a significant difference between the first three techniques (*Benham's Structured Techniques*, *Kee's Standard Spreadsheet Design Format* and *Ronen et al's Recommended Spreadsheet Structure*) and the fourth technique (*Ronen et al's Spreadsheet Flow Diagrams*). The first three techniques are based on developing a standard generic structure for the entire spreadsheet model, while the fourth technique employs an established method within structured systems analysis to specifically model the workings or calculations part of the spreadsheet. The main limitation of the fourth technique (*Ronen et al's Spreadsheet Flow Diagrams*) in this respect is that it does not address the other important aspects of the spreadsheet model. Therefore, it is recommended that this technique be applied within one of the first three techniques discussed in *Section 4.2.2*.

A comparison of the first three techniques would immediately reveal a fundamental difference between the first technique (*Benham's Structured Techniques for Spreadsheet Development*) and the next two techniques (*Kee's Standard Spreadsheet Design Format* and *Ronen et al's Recommended Spreadsheet Structure*). The first technique, *Benham's method*, is based on a block diagonal structure while the other two techniques are not. The advantage of this layout is that each section or module of the spreadsheet model is not adversely affected by row or column insertions or deletions in any other parts of the model.

There are certain important similarities among the first three techniques discussed in *Section 4.2.2*. They all attempt to adopt a standard, structured and disciplined approach to spreadsheet development. Apart from that, there is also an emphasis on the division of the model into distinct modules or components. Applying a modular structure to spreadsheet models makes them appear more organised and enhances their comprehensibility. This can also reduce the potential for errors. An examination of the proposed components or modules within each of the three techniques also shows certain similarities. *Benham's method* and *Kee's spreadsheet format* explicitly separate the input, workings and output components of the spreadsheet model. *Ronen et al's spreadsheet structure* performs this segregation within the *model* component. The proposed spreadsheet layout of all three techniques contains a section that clearly describes the spreadsheet model.

Based on the assessment of the advantages and limitations of the four techniques described in *Section 4.2.2*, it is recommended that the most effective approach would involve the use of *Benham's method* combined with *Ronen et al's Spreadsheet Flow Diagrams* to model the *data*, *model (workings)* and *analysis* sections.

4.3 Spreadsheet Development Life Cycle and Methodology

4.3.1 Life Cycles and Methodologies

After a thorough review of relevant literature, the following life cycles and methodologies have been selected for further analysis and discussion.

- PricewaterhouseCoopers (PWC's) Modelling Life Cycle (Read-99)
- The KPMG Modelling Process (KPMG-98a)
- Hayen and Peters' (Hayen-89) Spreadsheet Development Life Cycle
- Panko and Halverson's (Panko-96) Spreadsheet Development Life Cycle
- DiAntonio's Method (DiAntonio-86) for Spreadsheet Development
- Ronen et al's (Ronen-89) Spreadsheet Development Life Cycle
- Chadwick et al's 5-step Methodology incorporating the 3A's Approach (Chadwick-97)

PricewaterhouseCoopers' Modelling Life Cycle (Read-99)

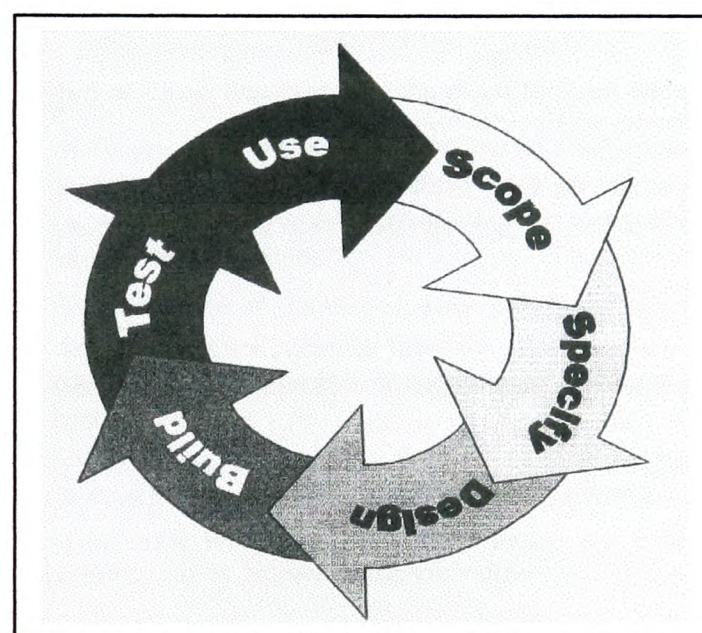


Figure 4.8: PWC's Modelling Life Cycle

Stage 1: Scope

This stage is where the nature, scale and complexity of the model are assessed. During the scope stage, decisions are made as to what needs to be included in the model and what can be omitted. The level of detail required in the input and logical assumptions is also established. Apart from that, estimates of time and resource required for the model development are also made.

Stage 2: Specify

To specify is to define the logic of the model in sufficient detail to provide an unambiguous statement of how the results will be calculated.

Stage 3: Design

The design stage involves producing the most effective structure for the model.

Stage 4: Build

The build stage is where the actual coding of the model takes place.

Stage 5: Test

To test a model is to root out errors and inconsistencies and to increase confidence in the results that the model produces.

Stage 6: Use

During this stage, there is a need to understand how to effectively present information and to manage the growth of the model.

The KPMG Modelling Process (KPMG-98a)

The modelling process applies to both, model review engagements and model development. KPMG's Modelling Process is illustrated in *Figure 4.9*.

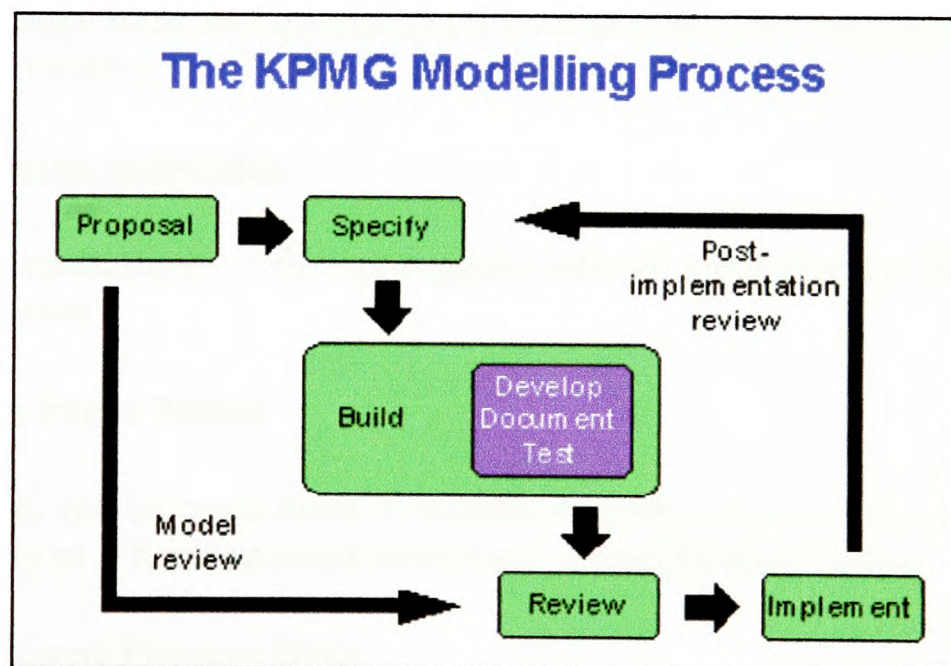


Figure 4.9: KPMG S MODELLING PROCESS

Stage 1: Proposal

This is the strategy-setting phase, and will normally involve one or more survey visits to the client by the engagement manager and/or partner. Objectives and scope of the model are agreed.

Stage 2: Specify

The key business risks which define the requirements for the model are identified. A specification document is subsequently prepared. It contains an *overview* of the model, *outputs*, *inputs* and *calculations* required.

Stage 3: Build

This stage involves three activities: *model development*, *documentation* and *testing*.

Stage 4: Review

The review may be either the core of a model review engagement, in which case the scope of the review will be stated explicitly in the proposal document, or an independent review following the *Build* stage of a model development engagement.

Stage 5: Implement

The content of the implementation stage will depend very much on whether the model is a one-off project model or an ongoing management model.

Hayen and Peters' (Hayen-89) Spreadsheet Development Life Cycle

The following steps form the *Spreadsheet Development Life Cycle* proposed by Hayen and Peters (Hayen-89):

Step 1: Determine Feasibility

Determine if a spreadsheet is the appropriate tool for analysing the business problem under consideration.

Step 2: Create Paper Model

The paper model can be considered in several different ways. It could be a workpaper created manually or a form selected from a set of standardised forms.

Step 3: Collect and Prepare Data

Collect and prepare the data required by the model.

Step 4: Enter Spreadsheet

Create the worksheets.

Step 5: Verify Spreadsheet Logic

To verify logic, spreadsheet applications should have as many built-in accounting tests as possible. The logic of the spreadsheet application can perform these tests automatically.

Step 6: Enter Application Data

If several different sets of data are to be input, the developer can establish separate areas of the spreadsheet for the input data and the equations.

Step 7: Produce Reports/Graphs

If more than one report or graph is to be produced, creating macros or command files to do the job helps ensure that the correct data are displayed.

Step 8: Review Results

The results should be reviewed for reasonableness before they get final approval.

Step 9: Prepare Documentation

In the development of a spreadsheet, the documentation should evolve. A final step should be to check its completeness.

Step 10: Sign off

After all the reviews, the final result can be signed off. Whenever a revision needs to be made, the process starts over.

Panko & Halverson's (Panko-96) Spreadsheet Development Life Cycle

Spreadsheet models, like programs, go through a series of development stages. These development stages (in order) are identified by Panko and Halverson (Panko-96) to be *requirements, design, cell entry, draft, debugging and operational use*.

Stage 1: Requirements and Design

As done in programming, it is extremely important to determine the requirements before actual construction of the spreadsheet is begun.

Stage 2: Cell Entry

This is the stage when numbers and formulae are entered in the spreadsheet cells. Many of the mistakes made at this stage are corrected immediately. However, some errors may be more frequent or more difficult to correct than others.

Stage 3: Draft

The draft spreadsheet should be tested with a variety of types of data or inspected cell-by-cell by the developer or an inspection team.

Stage 4: Debugging

In order to further reduce error rates, developers should engage in the debugging stage that involves data testing and code inspection.

Stage 5: Operational Stage

Even during the operational stage, errors are identified and corrected. However, this can be expensive and sometimes produce even more errors. As the models are in operation, extensive damage may have also been done before detection and correction of the errors.

DiAntonio's Method for Spreadsheet Development

DiAntonio (DiAntonio-86) has proposed a structured method consisting of six distinct steps for the construction of spreadsheets.

- Step 1:** The problem is understood and defined.
- Step 2:** Isolation of facts is done by splitting the spreadsheet into two parts, one for the *facts* and one for the *solution*.
- Step 3:** The solution is formatted or designed and it uses data from the *facts* part of the spreadsheet.
- Step 4:** The program is tested with sample data.
- Step 5:** The program is evaluated in terms of functionality, headings, labels and format.
- Step 6:** The program is documented either on the spreadsheet itself or in hard copy.

Ronen et al's (Ronen-89) Spreadsheet Development Life Cycle

Ronen et al's *Spreadsheet Development Life Cycle* is based on the traditional systems development life cycle. It is shown in *Figure 4.10*.

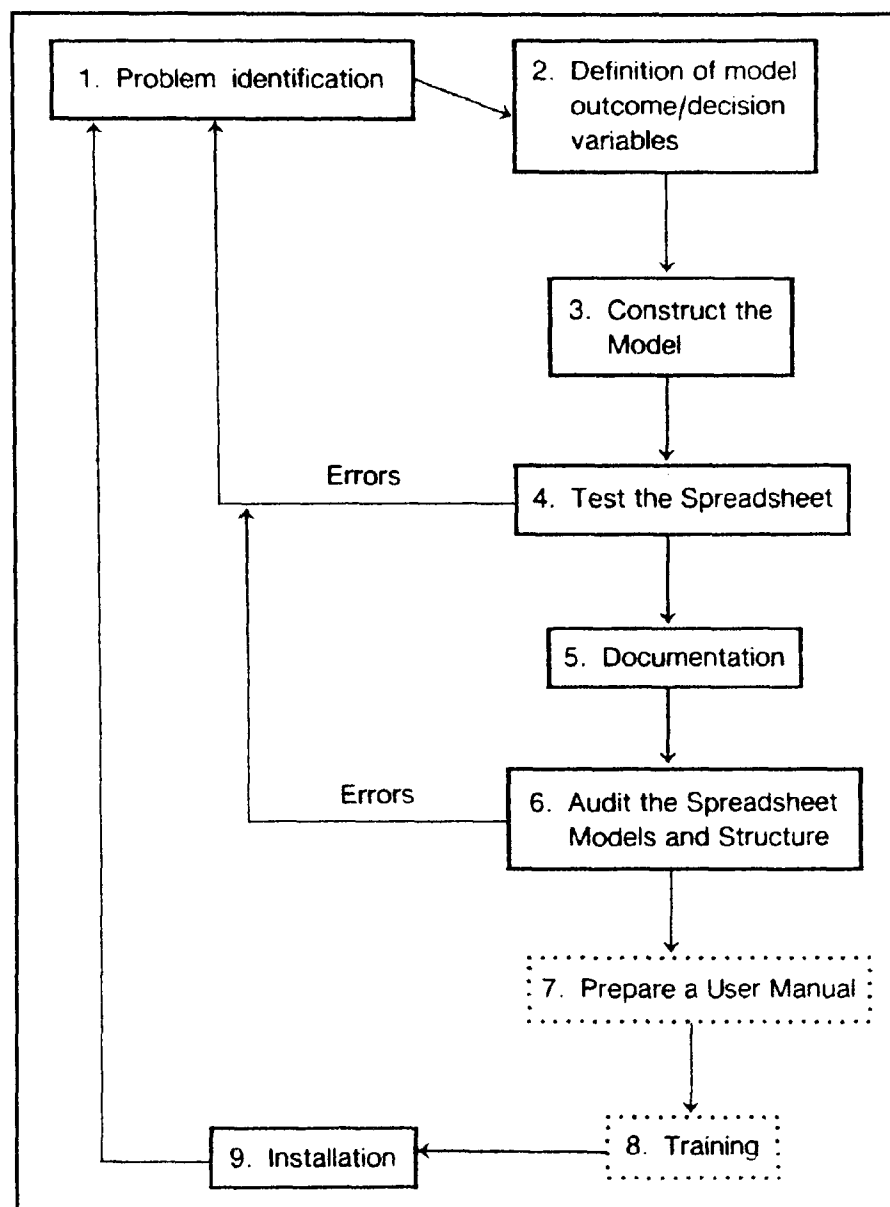


Figure 4.10: Ronen et al's Spreadsheet Development Life Cycle

Step 1: Problem Identification

The designer defines the nature of the problem to be solved.

Step 2: Definition of Model Outcome/Decision Variables

The spreadsheet is usually developed to produce results. The outcome variables need to be defined. An understanding of the outcome is generated is important. This part of the model represents the calculations which are undertaken in the model.

Step 3: Construct the Model

This stage corresponds to the traditional notion of programming. Using the various commands of the spreadsheet language, the model is built.

Step 4: Test

The results of the model are carefully tested. A hard-copy of the model and cell formulae are printed. All calculations are checked independently from the spreadsheet. The spreadsheet is also examined to see if there is an audit trail.

Step 5: Documentation

The spreadsheet model is documented on the spreadsheet itself. This involves inclusion of text on the spreadsheet that explains the model.

Step 6: Audit

The model and its structure are carefully reviewed. The use of audit packages is recommended.

Step 7: Prepare a User Manual (Optional)

For systems designed for others to use, a manual is a necessity. For applications created by the user, a manual is valuable if the application is to be used more than once.

Step 8: Training (Optional)

If the model is to be used by others, they may need to be trained prior to installation.

Step 9: Installation

The spreadsheet is prepared for use, for example, by installing it on a user's computer so that the model loads whenever the spreadsheet program is started.

Chadwick et al's (Chadwick-97) 5-Step Methodology

Chadwick et al (Chadwick-97) have proposed a five-step methodology for spreadsheet auditing, that incorporates the 3A's (appropriateness, accuracy, about-right) approach. An outline of the methodology is presented here.

Step 1: Appropriateness

Checking the appropriateness of the formula applied, from a logical point of view. Appropriateness is the correctness of the formula according to the underlying data model of the business process being modelled. The spreadsheet builder can verify appropriateness by entering the real-world description of the formula in the *cell note* for the cell. An example of this is shown in *Figure 4.11*.

	A	B	C	D	E	F	G
1	Lazy Days Staff Budget Costs 1995-1996						
2		Staff	Day Work	Night Work	Total	Average	Night
3		Numbers	Wages £	Wages £	Wages £	Wage £	Wages
4							of Tot
5	Managers	1	17700.50	0.00	17700.50	17700.50	

Cell Note

Cell: F5

Notes in Sheet:

F5: Average Wage

Text Note:

Average Wage is Total wages for the grade divided by the number of staff in that grade.

OK

Close

Add

Figure 4.11: Cell Note

Step 2: Accuracy

Checking the accuracy of the formula entered based on a correct interpretation of the data model. This can be done by checking the description given in the cell note under the appropriateness check in *Step1*.

Step 3: About Right

Checking if the resulting numeric value of the cell is about right.

Step 4: Validating a Formula Copy to a Cell/Range

Validating a formula copy to a cell or a range. This can be done by specifically showing the originating cell and the receiving range in the same colour font (Chadwick-97,00,00a). The use of this technique is illustrated in *Figure 4.12*.

	A	B	C	D	E	F
1	Lazy Days Staff Budget Costs 1995-1996					
2		Staff	Day Work	Night Work	Total	Average
3		Numbers	Wages £	Wages £	Wages £	Wage £
4						
5	Managers	1	17700.50	0.00	17700.50	17700.50
6	Grade 1	3	45540.00	1400.55	46940.55	15646.85
7	Grade 2	9	122340.00	2000.00	124340.00	13815.56
8	Grade 3	12	102350.25	0.00	102350.25	8529.19
9	Grand Totals	25	287930.75	3400.55	291331.30	55692.09

Figure 4.12: Colour-coding

Step 5: Modularisation

Modularising the spreadsheet by breaking it down into separate logical areas (modules). Each area should be prepared individually and bordered in bold lines on the screen display. Each *module* could have as many colour codings of formula copies as necessary as each will be treated as a stand-alone piece.

4.3.2 Critical Evaluation of the Life Cycles and Methodologies

The life cycles and methodologies presented in *Section 4.3.1* have various similarities and differences. They mainly vary in terms of level of detail and focus.

Apart from *Chadwick D et al's 5-step Methodology*, all the other life cycles/methodologies cover most of, if not all, the stages of spreadsheet development. *Chadwick D et al's 5-step Methodology* is therefore the least comprehensive approach. Its advantage, however, is that it places more emphasis on spreadsheet auditing compared to the other life cycles or methodologies proposed. It is recommended that this methodology be used within one of the other more comprehensive frameworks.

The most comprehensive life cycles or methodologies proposed are *PWC's Modelling Life Cycle* and the *KPMG Modelling Process*. To a large extent, both these frameworks have common steps or stages. These include *scoping, specification, design, building, testing* and *operation*. However, the precise sequence and scope of the constituent stages are different. Within each stage, the frameworks provide a detailed description or specification of the application of the relevant steps to the spreadsheet development process. The depth and comprehensiveness of both these approaches are mainly attributable to the fact that they have been developed by large multinational accounting/auditing firms. Apart from that, both frameworks are also more recently developed compared to the other life cycles/methodologies.

The other four life cycles/methodologies (*Hayen and Peters' life cycle, Panko and Halverson's life cycle, DiAntonio's method* and *Ronen et al's life cycle*) do not provide a detailed description of each stage of the life cycle. They do however address all stages of the spreadsheet development process. Apart from *Ronen et al's life cycle*, the other life cycles consist of a set of sequential stages. A study of the stages of the four life cycles shows that they are similar to the stages of the traditional systems development life cycle, especially *Ronen et al's life cycle*. The advantage of *DiAntonio's method* over the other frameworks is that it proposes the division of the spreadsheet into a *facts (data)* part and a *solutions (workings and output)* part. This produces a more organised model structure.

Among the life cycles and methodologies proposed for spreadsheet development, the most effective approach would be based on *PWC's Modelling Life Cycle* or the *KPMG modelling process*. In order to further enhance the quality of the framework, *DiAntonio's method* can be applied in the *specification/design* and *building* stages, and *Chadwick D et al's 5-step methodology* can be adopted in the *testing* or *review* stage.

4.4 Summary

This chapter has presented a discussion of a range of existing tools and techniques for improving the quality of spreadsheet models, and various life cycles and methodologies proposed for spreadsheet development. The merits and demerits of the tools and techniques, and a critical assessment of the life cycles and methodologies have also been provided.

Among the principal computer-based tools that have been developed to facilitate auditing and error detection in spreadsheet models are the *Spreadsheet Auditor*, *Cambridge Spreadsheet Analyst*, *MS Excel's Built-in Auditing Functions*, the *Excel Auditor*, *Spreadsheet Professional Audit Tool for MS Excel*, *Spreadsheet Detective*, the *Operis Analysis Kit (OAK)*, and *Spreadsheet Auditing for Customs and Excise (SpACE)*. On the other hand, some of the main *techniques* that have been proposed for the quality control of spreadsheet models include *Benham's Structured Techniques for Spreadsheet Development*, *Kee's Standard Spreadsheet Design Format*, *Ronen et al's Recommended Spreadsheet Structure* and *Ronen et al's Spreadsheet Flow Diagrams (SFD)*.

The *Spreadsheet Detective* has been found to be the most effective among the eight tools considered, having an excellent capacity to detect both existing errors and potential flaws. On the other hand, the *Excel Auditor* appeared to be the least satisfactory. Based on an analysis of the four techniques described in this chapter, the most highly recommended approach would involve the use of *Benham's method* combined with *Ronen et al's SFDs* to model the *data*, *workings* and *analysis* sections.

Various life cycles and methodologies have been proposed for spreadsheet development. They include *PricewaterhouseCoopers (PWC's) Modelling Life Cycle*, the *KPMG Modelling Process*, *Hayen and Peters' Spreadsheet Development Life Cycle*, *Panko and Halverson's Spreadsheet Development Life Cycle*, *DiAntonio's Method for Spreadsheet Development*, *Ronen et al's Spreadsheet Development Life Cycle*, and *Chadwick et al's 5-step Methodology incorporating the 3A's Approach*.

Chadwick et al's 5-step methodology is the least comprehensive approach but has the benefit of placing relatively more emphasis on spreadsheet auditing. It has been found that the most comprehensive life cycles/methodologies are *PWC's Modelling Life Cycle* and the *KPMG Modelling Process*. The most effective methodology would be a hybrid approach based on either *PWC's Modelling Life Cycle* or the *KPMG Modelling Process*, combined with *DiAntonio's method* in the *specification/design* and *building* stages, and *Chadwick et al's 5-step methodology* in the *testing or review* stage.

CHAPTER 5

PRELIMINARY INVESTIGATION AND DEVELOPMENTS

5.1 Introduction

In *Chapter 3*, a framework for classifying user-generated spreadsheet errors based on a rational taxonomic scheme was presented, while *Chapter 4* provided a discussion and evaluation of existing tools and techniques for the quality control of spreadsheet models, as well as life cycles and methodologies for spreadsheet development. Both these sets of activities were carried out in parallel.

Having gained an insight into the nature, characteristics and categories of specific types of spreadsheet errors and the existing tools, techniques, life cycles and methodologies, an investigation was carried out into various methods and approaches deemed to have the potential for enhancing the quality of spreadsheet models. This was preceded by an analysis of spreadsheet structure.

This chapter begins with a discussion of spreadsheet structure. The different aspects of spreadsheet structure considered are the components of a spreadsheet model, the structure of formulae and data dependencies. The outcome and findings of the preliminary investigation into the relevant techniques, methods and approaches, are subsequently presented. This is the main part of this chapter and represents the first step in the research programme, towards developing a comprehensive methodology for the integrity control and development of spreadsheet models.

5.2 Analysis of Spreadsheet Structure

5.2.1 Overview

A spreadsheet is a large matrix consisting of rows and columns. Rows are identified by numbers, while columns are identified by letters. The intersection of a particular row and column of a spreadsheet is an individually identifiable cell. A cell address is composed of a column label and a row label, e.g. A7 (column A, row 7). Brown (Brown-87) defines an electronic spreadsheet as a two-dimensional matrix of cells displayed on a computer screen. The contents of the cells can be *text*, *numeric constants*, or *formulae* that reference other cells. The underlying contents of a cell are not readily visible to the user; instead, what is displayed is the numeric result of the computation indicated within the cell. The formula can be viewed by moving the cursor to the cell (Brown-87).

A spreadsheet usually consists of connected components. It has a two-level structure, namely a visible (two-dimensional) surface and a hidden formula network (Saariluoma-91). Therefore, within the context of spreadsheet calculation, there are two levels: one which is visible and concrete, and the other which is more abstract and 'hidden' below the first. According to Saariluoma and Sajaniemi (Saariluoma-91), the surface level of a spreadsheet consists of a set of cells occupied by visible values. At the deep or hidden level, these cells are connected to each other and form a network

defined by a set of mathematical formulae in which variables are bound to the numerical contents of specified cells. The surface level displays data and numeric results of the formulae. Knowledge of the surface and deep levels of a spreadsheet is important when making deletions or changes to formulae. This helps in identifying the source of errors produced by the changes (Saariluoma-91).

5.2.2 Components of a Spreadsheet Model

Isakowitz et al (Isakowitz-95) propose two distinct perspectives to view spreadsheet models: *logical* and *physical*. The *logical* perspective consists of a formal and implementation-free description of the model's logic and data structures, while the *physical* level concerns storage, formatting, user interface, and other aspects that affect the model's implementation. From a physical perspective, a spreadsheet model is a collection of addressable cells, arranged in a two-dimensional grid (Isakowitz-95).

Isakowitz et al state that every spreadsheet model embeds an implicit *logical view*, which can be regarded as a set of *functional relations*. A functional relation consists of one or more attributes and of one or more tuples. However, unlike ordinary relations, functional relations have two types of attributes: *data* attributes and *functional* attributes. Data attributes define slots that store constants, whereas functional attributes are bound to functions that are calculated. Isakowitz et al (Isakowitz-95) use the term model's *schema* to refer to the set of functional relation definitions within a particular spreadsheet.

According to Isakowitz et al (Isakowitz-95), there are four principal components that characterise any spreadsheet model: *schema*, *data*, *editorial* and *binding*. The *schema* provides the spreadsheet's skeleton and stores a concise and formal definition of the spreadsheet's underlying logic. The *data* property is the structured collection of constants on which *schema* operates. The *editorial* property can be defined as what is left over in the spreadsheet model after *schema* and *data* have been carved out: titles, column and row headings, and documentation. Finally, the *binding* property is a logical-to-physical mapping that binds *schema*, *data*, and *editorial* to the spreadsheet grid, using cell addresses (Isakowitz-95).

5.2.3 Structure of a Spreadsheet Formula

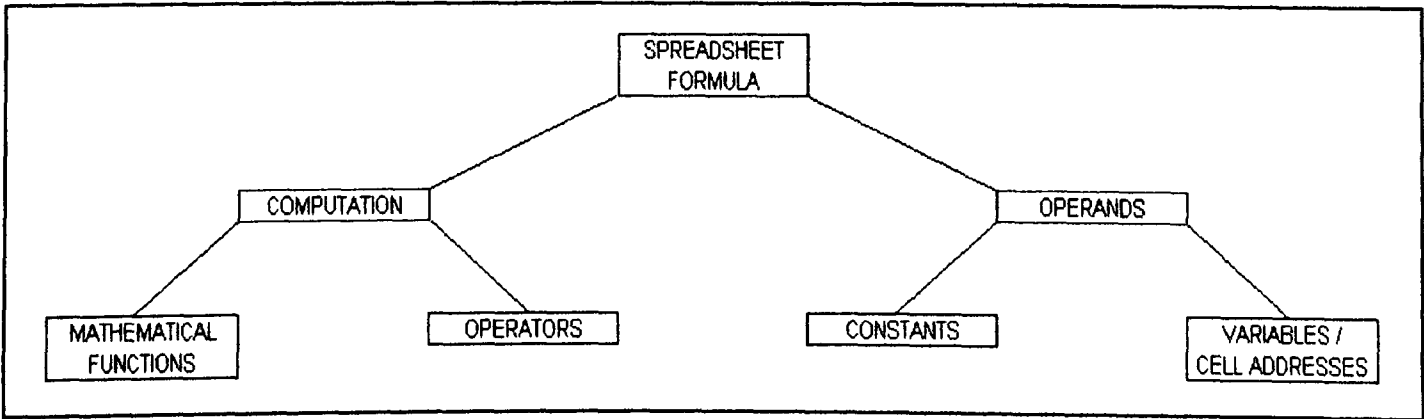


Figure 5.1: Components of a Spreadsheet Formula

Spreadsheet calculation is based on formulae. It is important to precisely define what a **formula** is and to analyse the structure of a formula. A formula is a mathematical clause. It performs calculations on constants (or absolute values) and referenced values. A formula has two principal components, a **computation** component and an **operands** component. The computation performed by a formula involves a number of **operators** and **mathematical functions** such as multiplication, division or summation of operands. Operators connect operands. Examples of operators are +, -, /, * and ^.

Operands are either **constants** or variables (references to other cells). The operands in a formula are bound to the values of the cells they refer to. From a physical perspective, **variables** take the form of referenced cells. An operator can also be applied over a block of cells or a range. A range is a rectangular block of cells.

Formulae can be divided into two broad categories based on their referencing property. They are **hard-coded formulae** and **referencing formulae**. A *hard-coded formula* always has the same numeric solution. Its numeric solution does not have the capacity to change unless the formula itself is rewritten. This is because the formula does not contain any *variables* as its operands, e.g. =45+45 or =AVERAGE(10, 15, 20, 25). A *referencing formula* does not always have the same numeric solution. Its numeric solution has the capacity to change without the formula itself having to be rewritten. This is because the formula contains at least one variable within its set of operands, e.g. =SUM(A1:A5) or =B2/12, where A1, A5 and B12 are cell addresses.

5.2.4 Data Dependencies

A spreadsheet is essentially a matrix where each entry can contain a number or a formula which references other cells. A cell **x** is said to be a *dependant* (or an *ancestor*) of another cell **y**, the *precedent* (or *descendant*), if **x** contains a formula that refers to **y** (Saariluoma-91). In this context, **y** is a *direct precedent* of **x**. If **x** refers to a cell **y**, which happens to be a dependant of another cell **z**, **z** is said to be an *indirect precedent* of **x**, but a *direct precedent* of **y**. In Figure 5.2 (Chadwick-97), Excel's auditing tool shows data dependencies. The *direct precedents* of cell **E13** are cells **E8** and **E9**. **E8** and **E9** are said to be direct dependants of **E13**.

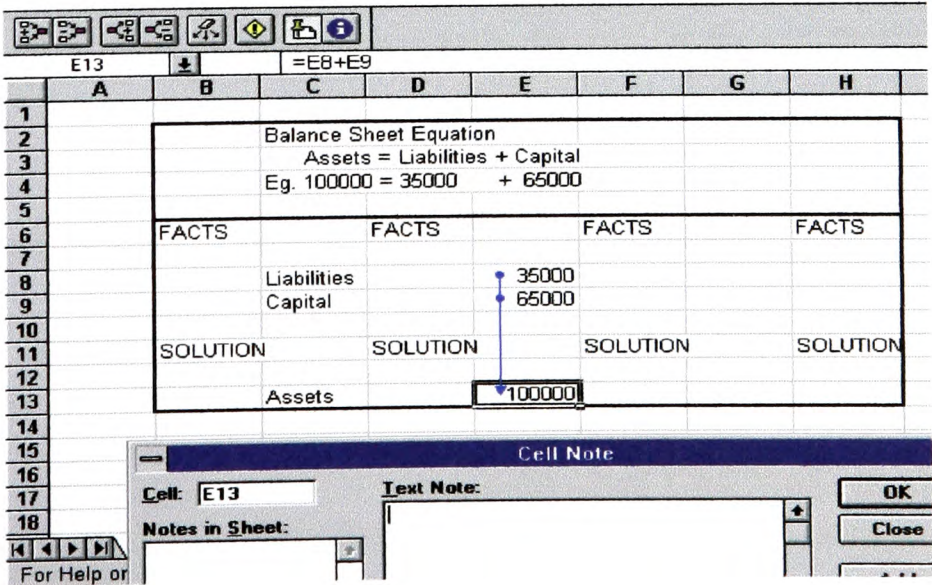


Figure 5.2: Data Dependency

Davis (Davis-96) also uses arrows to define data dependencies between cells. Referring to *Figure 5.3* (Davis-96), the formula in cell **H2** is **A4*B4**, which makes **A4** and **B4** are *direct precedents* of **H2**. **H2** is a *direct dependant* of **A4** and of **B4**. This shows that precedence and dependence are inverse relationships. As the formula in **B4** is **C6/100**, **C6** is an *indirect precedent* of cell **H2**.

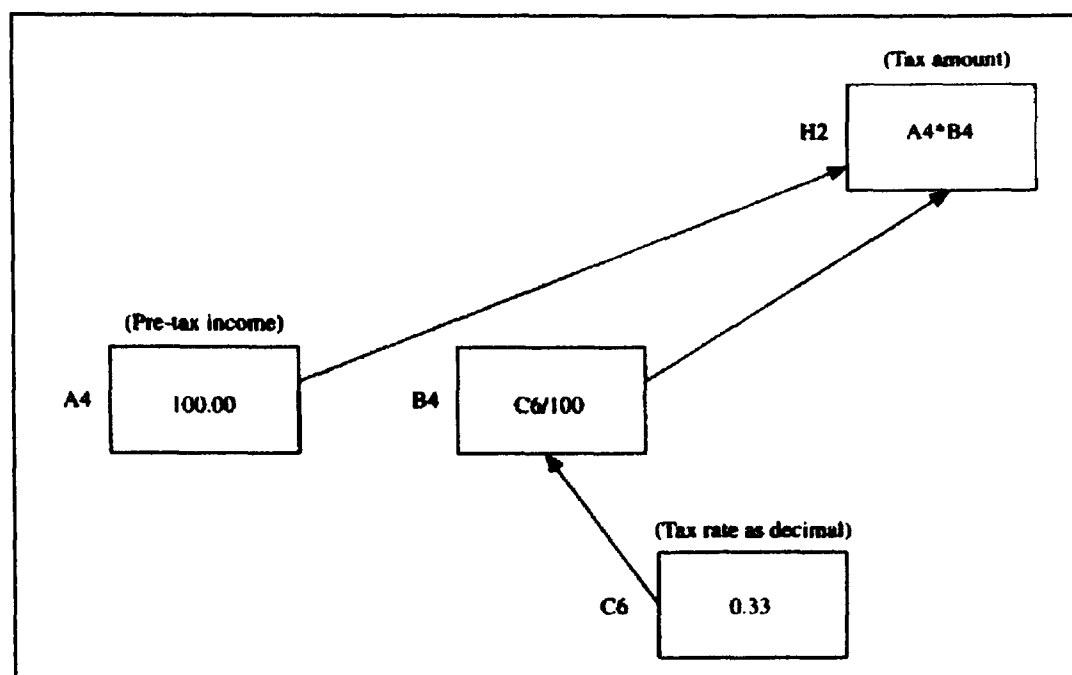


Figure 5.3: Davis' Data Dependency Diagram

5.3 Initial Approaches Explored and Developed

5.3.1 Overview

Prior to the development of the proposed structured spreadsheet modelling methodology presented in *Chapter 7*, various other approaches were explored and analysed. Some of the significant developments preceding the development of the proposed methodology are described in this section.

The approach of this research has been to examine the applicability of main-line software-engineering techniques to the needs of spreadsheet developers. These needs are partly determined by the visual nature of spreadsheets and their heavy reliance on referencing and intermediate data, and partly by the likely acceptance of techniques within the industry.

5.3.2 Modularisation Based on the Concept of an *Extent*

A technique or process of modularisation based on the concept of an *extent* was initially proposed (Rajalingham-98,99). The modular approach employs principles of software engineering such as *modularisation* and *coupling*. Support for the modular approach came from DiAntonio (DiAntonio-86) and Chadwick et al (Chadwick-97) but was weakly defined in both these sources. Within the context of spreadsheet development, *modularisation* refers to the structuring of the spreadsheet model into distinct blocks or modules with data being passed between them. An important justification for this approach is that the human mind finds it difficult to interpret and

process large chunks of data. When data is logically and systematically split into smaller parts, it simplifies analysis.

The modular approach dictates the division of the physical model (spreadsheet data) into distinct modules. The fact that the spreadsheet is separated into separate blocks or modules suggests that a modular approach is being taken, based on an analysis of spreadsheet structure. The term given to a distinct module of the spreadsheet is an **extent**. An *extent* can be defined as a matrix representing a logical area or module of the spreadsheet. An extent is a range with special properties. It has various special characteristics. A spreadsheet **model** is defined as a collection of inter-related extents.

The minimum size of an extent is a 2 by 2 range (4 cells). The first column of an extent contains the row headings while the first row of an extent bears the column headings. Every cell within a particular column (except the first column) is associated with the same column heading, which occupies the top cell of that column. Similarly, Every cell within a particular row (except the first row) is associated with the same row heading, which occupies the left-most cell of that row.

Column headings and row headings of an extent must be defined by the user. No two cells can have exactly the same combination of column heading and row heading as there cannot be two or more column headings or row headings with the same name, although a column heading can share the same name with a row heading.

The following steps are taken in defining an *extent* (Rajalingham-98,99):

Step 1

Every value must be placed at the intersection of a particular labelled column and a labelled row, and must be semantically consistent with the meaning the pair has in real life.

Step 2

Every new entry or value for which there already exists both a corresponding column label and a corresponding row label, must be entered in the cell at the intersection of the particular column and row.

Step 3

If a new entry only has either a corresponding column label or a corresponding row label present within the existing structure, then the missing column/row label is added to the extent. If the new entry has column and row labels that do not semantically match any of the existing column and row labels, it must be placed in a different extent. The resulting generic structure of an extent is shown in *Figure 5.4* (Rajalingham-98). The spreadsheet model shown in *Figure 5.5* is an example of an *extent*. The process of modularisation, based on a similar approach, was subsequently presented more elaborately with examples by Chadwick et al (Chadwick-99a).

	q	q+1	q+2	l	
q,r							r
							r+1
							r+2
							.
							.
							.
						l,m	m

Figure 5.4: Extent

- The **boundary** of the extent is defined in terms of its top-left cell (column **q**, row **r**) and bottom-right cell (column **l**, row **m**).
- **Column headings** are contained in cells in the first row (row **r**) of columns **q+1** to column **l**.
- **Row headings** are contained in cells in the first column (column **q**) of rows **r+1** to row **m**.
- **Data values** are contained in all the other cells except the top-left cell (column **q**, row **r**)

C	D	E	F	G	H	
	Number of Staff	Day Wages £	Night Wages £	Total Wages £	Average Wage £	5
Grade 1	1	17700.50	0.00	17700.50	17700.50	6
Grade 2	3	45540.00	1400.55	46940.55	15646.85	7
Grade 3	9	122340.00	2000.00	124340.00	13815.56	8
Grade 4	12	102350.25	0.00	102350.25	8529.19	9
Grand Total	25	287930.75	3400.55	291331.30	11653.25	10

Figure 5.5: An Example of an Extent

5.3.3 ‘Diagonalisation’ of a Spreadsheet Model

This approach involves placing *extents* diagonally on the spreadsheet (Rajalingham-99). This prevents any column or row of the spreadsheet from containing more than one label. It also isolates every cell entry from row or column insertions or deletions in any other parts of the model. The idea underpinning this technique was inspired by the block diagonal structure proposed by Benham et al (Benham-93), described briefly in *Section 4.3.2 in Chapter 4*.

An example of a diagonalised model is displayed in *Figure 5.6* (Rajalingham-99). The structure shows that there are cells within a particular extent that are (permanently) unused. These should be marked or labelled appropriately (e.g. N/A for not applicable) and protected. The model in *Figure 5.6* consists of two inter-related extents. The diagonalisation of these extents leaves behind two unused regions for every two extents. These regions can be used to describe the relationship (if any) between the two extents.

A	B	C	D	E	F	G	
	Tea(£)	Milk(£)	Coffee(£)	Ringgit Malaysia			1
1 st Quarter	450	560	467	N/A			2
2 nd Quarter	904	900	352	N/A			3
3 rd Quarter	872	800	233	N/A			4
4 th Quarter	123	234	901	N/A			5
Total Sale	=SUM(B2:B5)	=SUM(C2:C5)	=SUM(D2:D5)	=SUM(E2:E5)*G10			6
						Exchange Rate (£ to RM)	7
					2 Months Ago	6.9	8
					Last Month	8.1	9
					Current	7.3	10
					Average	=AVERAGE(G8:G10)	11

Figure 5.6: Diagonalised Model

This approach has numerous benefits. It enables the various extents of a particular spreadsheet model to be laid out in an organised fashion. This may decrease the probability of the occurrence of errors. Any ambiguity concerning the meaning of data is also removed as each *cell entry* is associated with a meaningful pair of column and row labels. Apart from that, this benefit is also derived from the fact that a particular row or column in the spreadsheet is only associated with one meaningful label (Rajalingham-99).

5.3.4 Visual Representation of a Formula in *Natural Language*

As a by-product of the modular approach, it is possible to visually represent the elements of a spreadsheet formula (Rajalingham-98,99). Most of the errors that occur in spreadsheets concern formulae. When such errors are committed, it is often difficult to detect and correct them based on the original structure of the formulae that appears in the formula bar of the spreadsheet screen. This is primarily due to the use of cell addresses in the formulae to refer to data.

This problem can therefore be overcome if formulae were represented in a more visual, English-like and comprehensible form. This will certainly facilitate the validation and audit of spreadsheet formulae. The proposed technique for visually representing spreadsheet formulae will be able to present formulae in such a form. Any software tool used to implement this technique will be able to convert a formula written by a user in conventional form, expressed in terms of cell addresses, into a form that is more readable and visual. This is done mainly by displaying the corresponding column and row labels of each cell referenced by a formula. This makes every spreadsheet cell value meaningful and also ensures that the user understands this meaning when creating and using the spreadsheet.

Several methods of presenting such formulae have been developed in this form (Rajalingham-98). These different methods are presented as *methods 1, 2 and 3*, based

on the spreadsheet model in *Figure 5.5*. Referring to *Figure 5.5*, formulae are present in the following cells: **G6 to G10**, **H6 to H10**, and **D10 to F10**.

The formulae selected to illustrate the various methods are:

CELL	FORMULA
F10	=SUM(F6:F9)
H10	=G10/D10

Method 1: Algebraic English

Cell Representation of the Formula

F10	=SUM(Night Wages £_Grade1:Night Wages £_Grade4)
H10	=Total Wages £_Grand Total/No of Staff_Grand Total

This method simply converts each cell address to its corresponding column and row headings but retains the binary operators.

Method 2: Fully English

Cell Representation of the Formula

F10	<i>Night Wages £_Grand Total</i> = SUM (Night Wages £_Grade1 (to) Night Wages £_Grade4)
H10	<i>Average Wage £_Grand Total</i> = Total Wages £_Grand Total (divided by) No of Staff_Grand Total

This method converts each cell address to its corresponding column and row headings as well as each binary operator from symbol to natural language.

Method 3: Graphic Display

Cell: F10
Representation of the Formula:

F10	Night Wages £	=SUM	F6	Night Wages £	TO	F9	Night Wages £
Grand Total	3400.55		Grade 1	0.00		Grade 4	0.00

Cell: H10
 Representation of the Formula:

H10	Average Wage £	=	G10	Total Wages £	DIVIDED BY	D10	Number of Staff
Grand Total	11653.25		Grand Total	291331.30		Grand Total	25

This is the most visual of the three methods. Apart from associating each cell address with its column and row headings, this method also displays the value contained in the particular cell. In each display, a different colour is used for each different column heading and row heading. For instance, in the first example, all three cells shown have column heading (*Night Wages £*) in the same colour (*blue*) text to indicate they are all in the same column. However, the row headings (*Grand Total*, *Grade 1* and *Grade 4*) are in different colours (*red*, *green* and *purple*) indicating that they are in different rows.

A survey (Rajalingham-98) was carried out to determine the preference of students to these visual methods. The students were presented with four choices: the normal *MS Excel* formula style and the three mentioned above. They were asked to rank them in order of clarity and ease of understanding. There were 63 respondents to the questionnaire. 46 respondents (73%) indicated preference for the visual methods. 21 of them (34%) chose *method 3* (the graphic display) as the most clear and easiest to understand. It was the most appealing of the four choices, with the normal formula style (26%), the algebraic English (18%) and the totally English (22%).

5.3.5 The Use of Tree Structures to Express Data Dependencies

Hierarchical decomposition is a technique commonly used in software engineering to gradually break down the complexity of programs. A similar approach can be adopted in spreadsheet design and development. The objective of this technique is to represent the elements of a spreadsheet formula in the form of a tree (Rajalingham-99,00). This can be used to confirm our understanding of the formula structure and may be used as a means of documenting the design of the spreadsheet model. All types of formulae can be represented in the form of a tree, including the spreadsheet (e.g. *MS Excel*) built-in functions.

The general format of a function is as follows (Kantaris-94):
 = name (argument1, argument2 ...)

'name' is the function name, and 'argument1', 'argument2', etc., are the arguments required for the evaluation of the function. Arguments must appear in a parenthesised list as shown above and their exact number depends on the function being used. However, some functions do not require arguments and are used without parentheses.

The examples displayed in *Figure 5.7* (Rajalingham-99,00) clearly show how this task is performed. The tree represents all the elements of a particular formula (hard-coded constants, cells referenced, operators and mathematical functions).

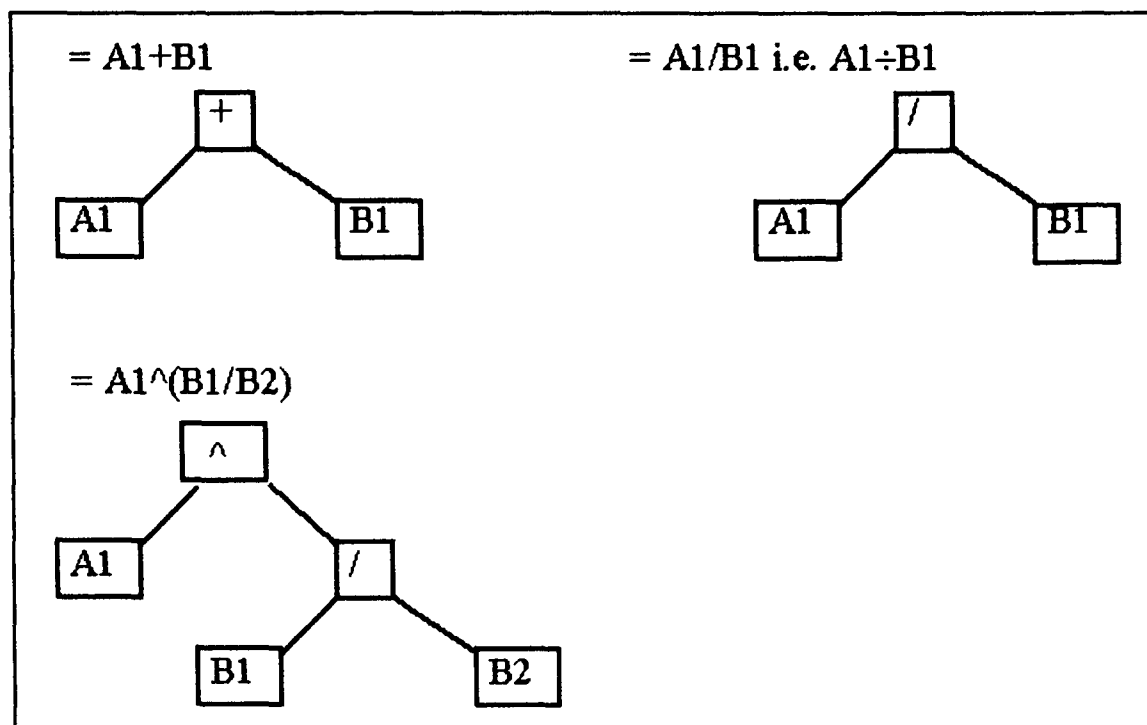


Figure 5.7: Formulae Represented Using Tree Structures

As all functions are of the same form, = **name (argument1, argument2 ...)**, we can represent each in the form of a tree (not necessarily a binary tree). The root would now contain the function name while each argument would form a node. An example is given in *Figure 5.8* (Rajalingham-99,00).

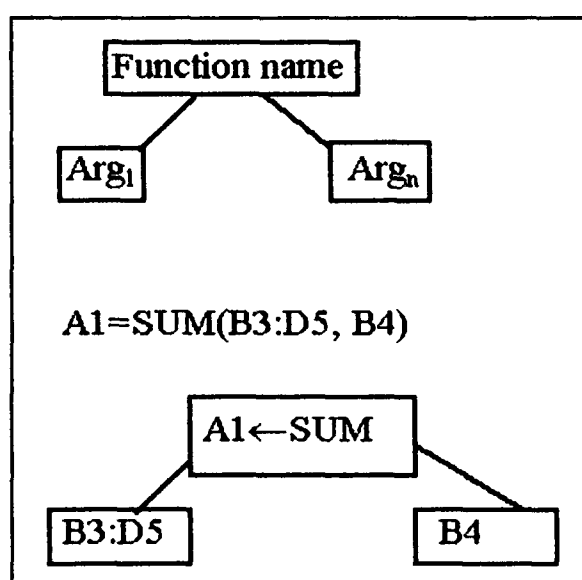


Figure 5.8: Formulae Represented Using Tree Structures

Tree structures can also be used to represent the logical aspect of the formula, independent of physical location. Examples of this are given in *Figure 5.9* (Rajalingham-00). In *Figure 5.9*, *cost of goods sold* is the sum of *opening stock*, *purchases* and *carriage inwards*. Based on *Figures 5.7, 5.8 and 5.9*, it can be seen how these tree structures can be used to facilitate comprehension, analysis and documentation of formulae. The use of tree structures have been proposed in the analysis and design stages of the *Integrated Spreadsheet Engineering Methodology* presented by Rajalingham et al (Rajalingham-99a). This methodology is mainly based on the classical systems development life cycle by Aktas (Aktas-87).

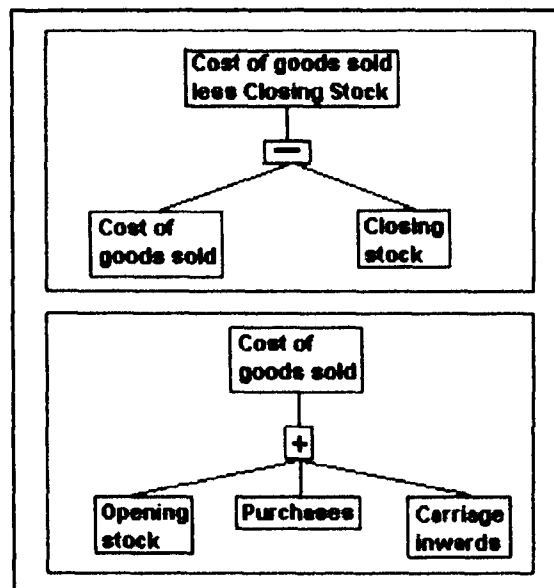


Figure 5.9: Logical Aspect of Formulae

5.4 Summary

A spreadsheet is a two-dimensional matrix of cells that has a two-level structure consisting of a visible surface and a hidden formula network. A spreadsheet model can be viewed from both a logical and physical perspective. It is made up of four main components: *schema*, *data*, *editorial* and *binding*. A spreadsheet performs calculations through formulae. A spreadsheet formula consists of a *computation* component and an *operands* component. If a cell x contains a formula that refers to cell y , x is the *dependant* while y is the *precedent*.

An initial software engineering based method developed was a technique of modularisation based on the concept of an *extent*. Using this technique, the physical spreadsheet model is split into distinct but logically related modules (or matrices) with special characteristics, called *extents*. This technique was subsequently enhanced through the *diagonalisation* of the spreadsheet model. This involves placing *extents* diagonally on the spreadsheet to isolate cell entries from row or column insertions or deletions in other parts of the model.

A by-product of the modular approach is a technique for visually representing elements of a spreadsheet formula in a more comprehensible form. This facilitates more effective validation and audit of spreadsheet formulae. Alternative methods of presenting formulae in such a form include *algebraic English*, *fully English* and *Graphic Display*. By combining the techniques of visual modelling and hierarchical decomposition, *tree structures* can be used to model data dependencies during spreadsheet analysis and design. These tree structures can represent both the logical and physical views of a formula. This enables better comprehension, analysis and documentation of spreadsheet formulae.

CHAPTER 6

SOFTWARE ENGINEERING PRINCIPLES AND JACKSON STRUCTURES

6.1 Introduction

The flexibility and freedom offered by a spreadsheet has set it apart from conventional applications and programming languages. However, as discussed in *Chapters 2 and 3*, spreadsheets are more prone to errors compared to conventional programs and applications. Even these conventional programs and applications had numerous errors and flaws that were successfully reduced with the application of structured methods. A natural approach to enhancing the quality of spreadsheets should therefore involve the application of structured methods and software engineering principles.

Chapter 5 presented the outcome and findings of a preliminary investigation into various methods and approaches deemed to have some potential in improving the quality of spreadsheet models. This was followed by a more thorough examination of relevant software engineering principles and structured techniques, and their potential application to the design and development of spreadsheet models.

This chapter discusses related software engineering principles and methods, as well as their application to spreadsheet development. The main techniques and principles underpinning the proposed structured methodology are derived from these methods and techniques. Extensive emphasis is placed on Jackson structural forms as the application of these structures is an essential part of the proposed methodology. A general discussion of software engineering principles and their application to spreadsheet development is first presented. The rationale for the selection of Jackson structural forms is then explained along with the concepts, notations and rules of Jackson structures. This is followed by a discussion of other relevant software engineering principles. In the next chapter, *Chapter 7*, the proposed structured methodology for the development and integrity control of spreadsheet models is elaborately described and presented with illustrative examples.

6.2 Software Engineering Principles

This section presents a general discussion of software engineering principles and their application to spreadsheet development. There is no universally accepted definition of *software engineering* (Jones-90). It has numerous definitions.

Sommerville (Sommerville-01) defines software engineering as an engineering discipline which is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. The *IEEE Standard Glossary of Software Engineering* defines software engineering as the systematic approach to the development, operation, maintenance, and retirement of software (IEEE-83). Steward (Steward-87) states that the field of software engineering is concerned with all of the activities involved in the solution of problems through the development of computer systems.

These and most of the other definitions offered, clearly establish the scope of and general approach to software engineering. It is a systematic approach that encompasses all aspects, stages and activities involved in the development of software systems. This can be applied to spreadsheet development. Spreadsheet development should adopt a systematic and organised approach that covers all stages and activities of the spreadsheet building process.

There are various principles of software engineering that are applicable to spreadsheet development. Many publications (Bell-00, Sommerville-01, Jones-90) state that software engineering is concerned with the selection and development of the most appropriate methods, tools and techniques used for producing software. According to Sommerville (Sommerville-01), software engineering methods are structured approaches to software development which include system models, notations, rules, design advice and process guidance. Most of the methods and techniques are based on a graphical representation of system models as the basis for system specification or design. This principle can be employed in spreadsheet development. In order to adopt a structured approach, appropriate methods, tools and techniques can be borrowed or developed, and used within the spreadsheet building process.

The investigation of the field of software engineering has revealed that other important principles are also applicable to spreadsheet development. They are as follows:

- an emphasis on finding out and defining the exact requirements of users (Bell-00, Steward-87)
- formal specification of the requirements of a system (Bell-00)
- greater emphasis on quality control and eliminating errors (Bell-00, Jones-90)
- look at the broad picture first, ignoring details, then look at successive smaller parts in greater detail (Steward-87).

The normal stages of the software life cycle (van Vliet-96, Jones-90) are:

- Specification
- (Requirements) Analysis
- Design
- Implementation
- Testing
- Operation and Maintenance

Each software system passes through these stages. A software development process model describes how, and in what order, these stages are organised and carried out. The following are the main software development process models that have been proposed or developed (Jones-90, van Vliet-96, Bell-00):

- (traditional) waterfall
- prototyping
- formal methods
- spiral

6.3 Rationale for Selection of Jackson Structures

This section contains a discussion of why Jackson structural forms are considered to be the most appropriate for the proposed methodology described in the next chapter, *Chapter 7*. A justification of why other approaches have been dismissed, is also provided.

The problem of errors in spreadsheet development can, in many ways, be compared to the days of main-line software development before the advances due to structured programming, analysis and design. Numerous publications (Ronen-89, Benham-93, Isakowitz-95, Panko-96, Kavanagh-97) have proposed the adoption of these techniques to spreadsheet development, in order to overcome the problem.

Several programming and design methodologies originated during the 1960s and 1970s, with goals to systematise the process of software analysis and design, in order to reduce errors and improve quality in the development process. Among the important *data-oriented* methods proposed were M.A. Jackson's *Jackson Structured Programming* (Jackson-75, Cameron-83, Ingevaldsson-86), *Chen's Entity Relationship (E-R) Data Modelling* (Chen-76) and the *Warnier-Orr* methodology (Warnier-81, Orr-81). In the 1980s and 1990s, these methods were supplemented by *object-oriented* methods (Rumbaugh-91, Booch-94). As these methods concentrate primarily on the logical structure of data, it was believed that they could be potentially applied effectively in spreadsheet development.

There are several important reasons for selecting *Jackson Structured Programming (JSP)*. The principal purpose was for practical reasons. The spreadsheet user community or market is varied and unsophisticated. This rules out more complex methods such as *Chen's E-R Data Modelling* and *Object-oriented Modelling*. A diagrammatic tool is essential for logical modelling. This is a basic software engineering principle.

Among the various methods considered, it was found that the simplest tool in concept is *Jackson Structures* based on *JSP*. This is primarily because it relies only on data dependencies. Data dependencies are very well understood in the spreadsheet community as a result of their familiarity with cell references and the use of auditing tools. Therefore, as a first step, it was quite clear that the use of Jackson structural forms seemed to be the most favoured candidate. This is primarily due to the current state of spreadsheet users' computing knowledge and experience. According to Ingevaldsson L (Ingevaldsson-86), *JSP* notation can be easily taught to end users. The other methods such as *E-R modelling*, the *Warnier-Orr methodology* and *Object-oriented methods* require relatively high spreadsheet user skills.

Jackson Structured Programming (JSP) has been fairly widely promulgated, particularly in Europe, where it has been successful as a standard and in the development of software systems (Cameron-83). Programmers using *JSP* have found that it results in few, if any, logical errors (Ingevaldsson-86). He also states that the clearly defined step-by-step approach adopted enables different programmers applying *JSP* to present similar solutions to the same problem.

It appears that there are several possible advantages to the adoption of a structured approach based on Jackson structures. These advantages may be summarised as follows:

- a structured diagrammatic representation of the *logical design* of the spreadsheet model's schema
- a well-defined approach to *modularisation*
- a *top-level overview* of model and module structures
- a structured *indented* format to the layout of the model as a whole and its modules
- the possibility of *automatic structuring* of new spreadsheet models and automatically re-structuring existing spreadsheet models

6.4 Concepts and Notations of Jackson Structures

Jackson structures (Jackson-75) are named after their originator Michael Jackson. The essence of *Jackson Structured Programming (JSP)* is the structure diagram and its relationship to block structure, with its three key constructs of *sequence*, *selection* and *iteration*. Jackson structures offer an elegant diagrammatic way of showing sequence, selection and iteration in program or data structures (Weaver-02).

Figure 6.1 shows a structure diagram, representing a typical block structured module. The repeated parts of the structure are denoted by an asterisk (*) in the top right-hand corner. The structure parts which are selections and therefore mutually exclusive, are denoted by a small circle in the top right-hand corner of the box. The diagram shows that **A** consists of a repeated block **B**, and each **B** is made up of either **C** or **D**. **C** is a sequence of blocks **E** and **F**.

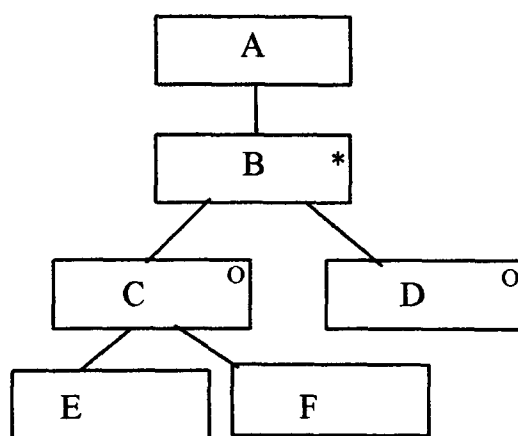


Figure 6.1: An Example Jackson Structure Diagram

The **top box**, **A**, contains the name of the structure. This name describes the contents of the structure. The bottom boxes or '*end leaves*' (i.e. those that have no other boxes below them) are known as **structure elements** (Weaver-98) or **leaves** (Ingevaldsson-86). In *Figure 6.1*, the leaves are **D**, **E** and **F**. **Structure boxes** (**B** and **C**) are all of the intermediate boxes, between the top box and the end-leaves (Weaver-98).

6.4.1 Sequence

A sequence has two or more parts, occurring once each, in order (Jackson-75). The sequence of the blocks or boxes of a Structure Diagram is read from left to right. Based on Figure 6.2 (Ingevaldsson-86), A is a sequence of B, C and D. D in turn, is a sequence of E and F. We refer to the bottom blocks/boxes, B, C, E and F as *leaves* (Ingevaldsson-86) or *structure elements* (Weaver-98).

In Figure 6.3 (Weaver-02), X is a sequence of A, B, C and D. The diagram is read from lefty to right. Therefore, A is followed by B, B is followed by C and so on so forth (Weaver-02). X is the *root node* or *top box*. It can also be regarded as the *parent* of A, B, C and D. On the other hand, A, B, C and D are considered *children* of X. There is effectively a *one-to-many* relationship between a *parent* and a *child* with *parent* at the *one-end* of the relationship. Though a *parent* can have one or more *children*, each *child box* must belong to one and only one *parent*. A Jackson structure always has one *root node* (Weaver-02) or *top box*. It appears at the top of the diagram.

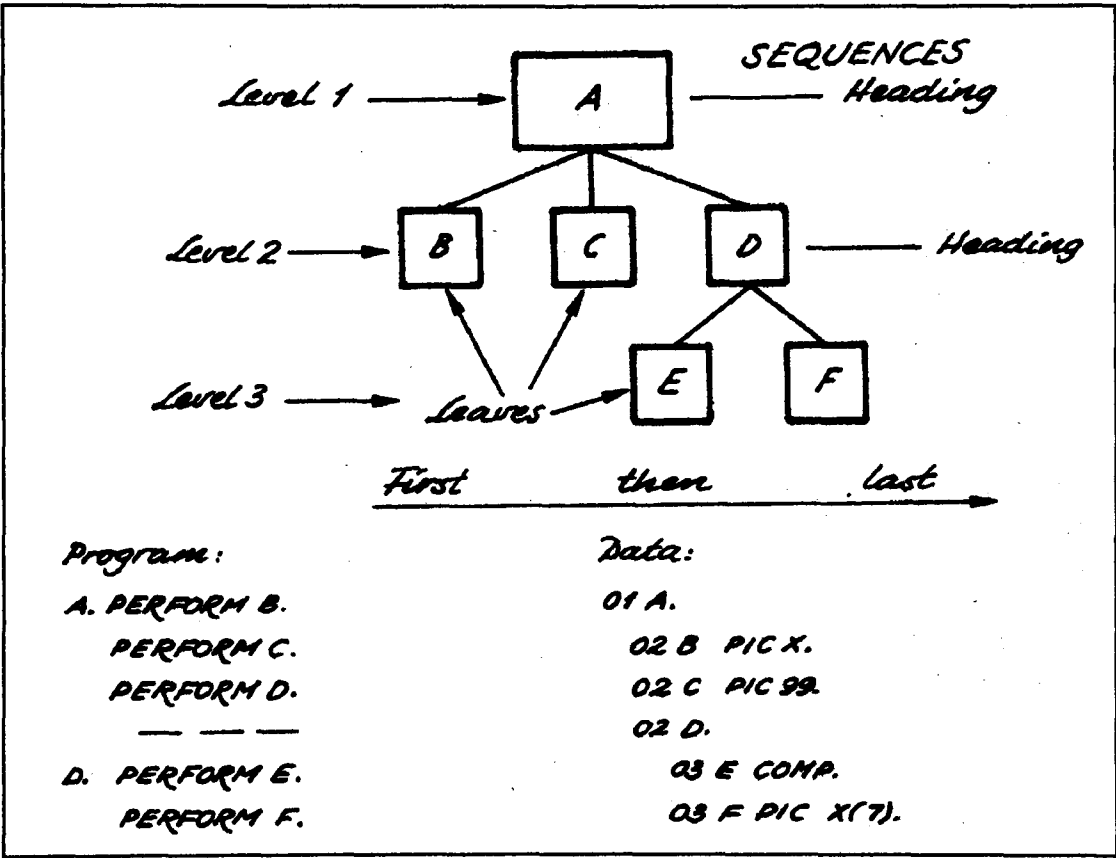


Figure 6.2: Sequences

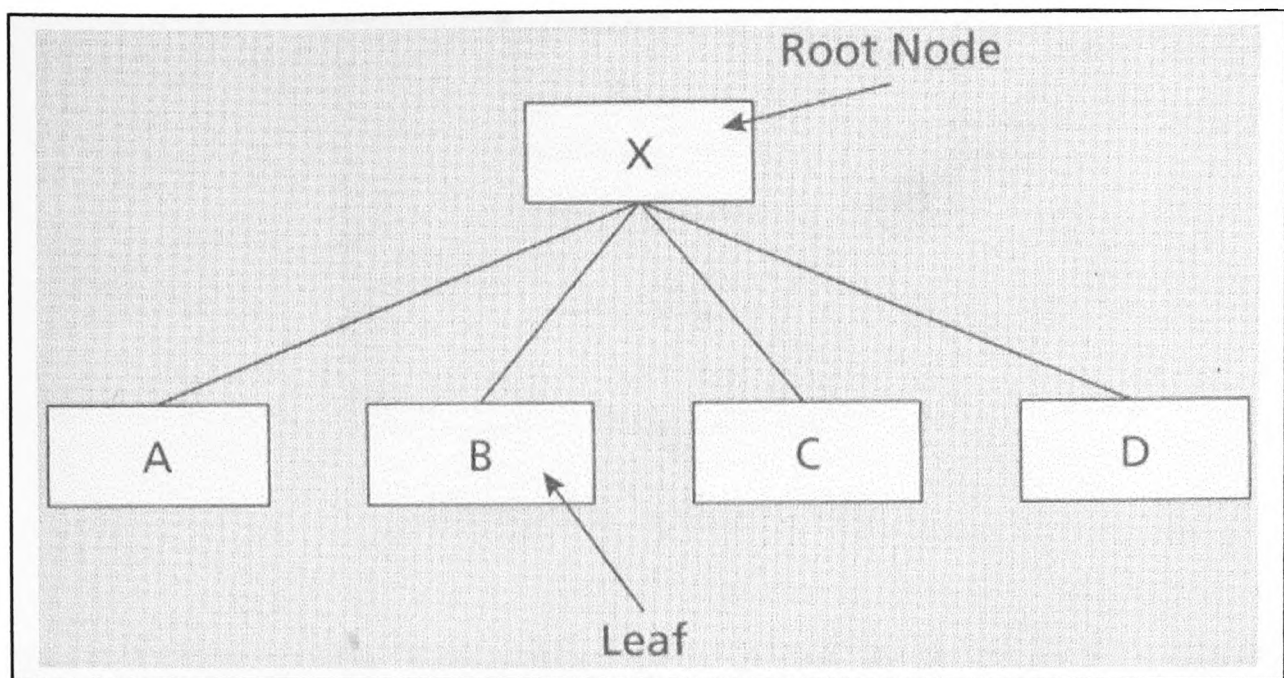


Figure 6.3: Sequence

6.4.2 Selection

Some parts of the structure diagram are mutually exclusive, i.e. there has to be a selection of one element or another, but not both. A selection has two or more parts, of which one, and only one, occurs once for each occurrence of the selection component (Jackson-75). Selections are denoted by a small circle in the top right-hand corner of the box. Referring to *Figure 6.4*, **A** is a selection of either **B** or **C**, but not both. Some elements of a structure diagram may be entirely optional, i.e. a null selection is possible. Based on *Figure 6.5*, **A** is a selection of **B** or nothing else.

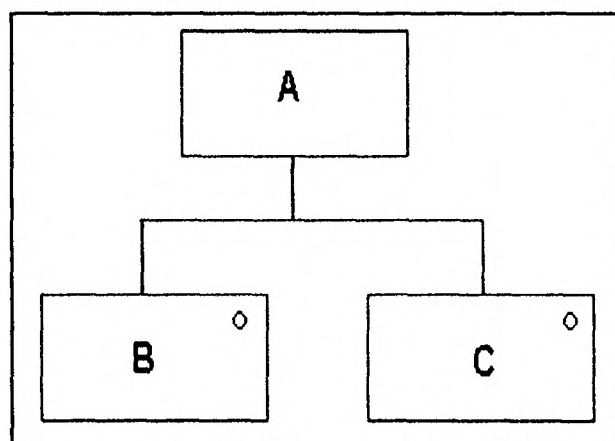


Figure 6.4: Selection

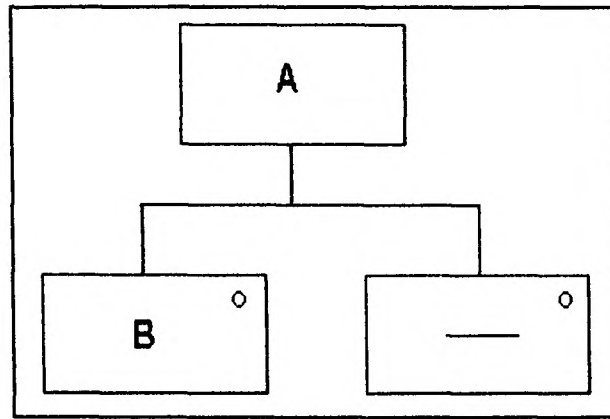


Figure 6.5: Null Selection

In *Figure 6.6* (Weaver-02), **Y** is a selection of **B** or **C** or **D**. When **Y** is reached, one and only one of **B**, **C** and **D** must be selected.

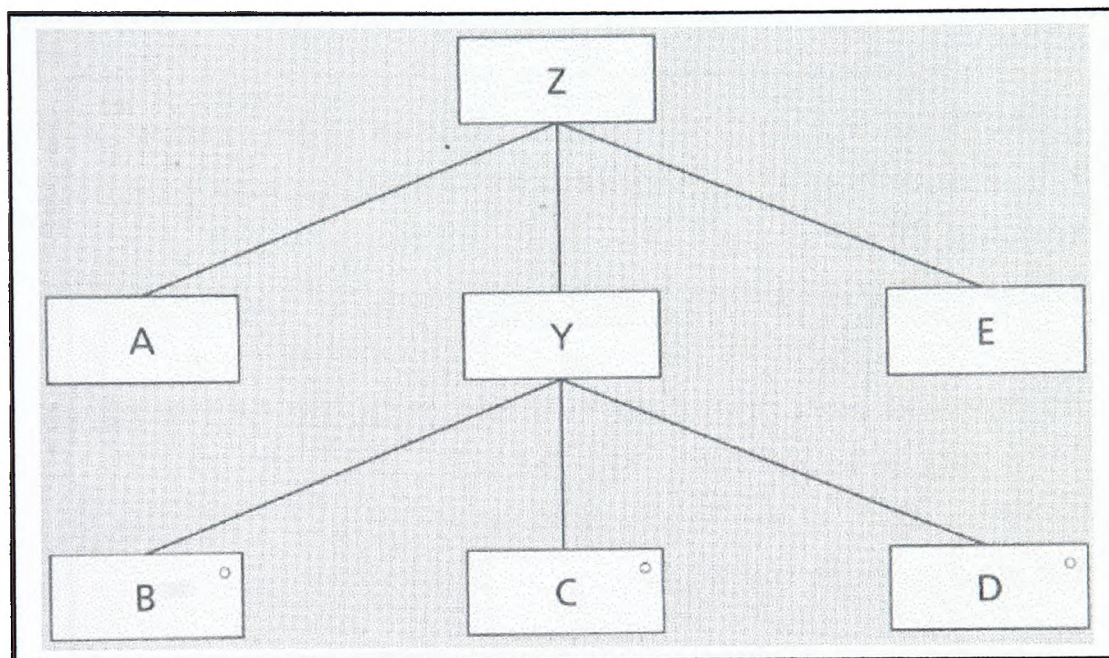


Figure 6.6: Selection

6.4.3 Iteration

Parts of the structure diagram may have to repeat several times. Iteration or repetition is indicated by an asterisk (*) in the top right-hand corner of the box. An iteration has one part, which occurs zero or more times for each occurrence of the iteration component itself (Jackson-75). All parts of a structure diagram below an iteration box or component would be subject to the iteration (Weaver-98).

Based on *Figure 6.7*, the iteration box **B** means that the sequence of **C** and **D** is also iterated. In other words, the sequence can be repeated. Although an iteration can have only one child (a box marked with an *), that child can be the parent of an entire sub-structure (Weaver-98).

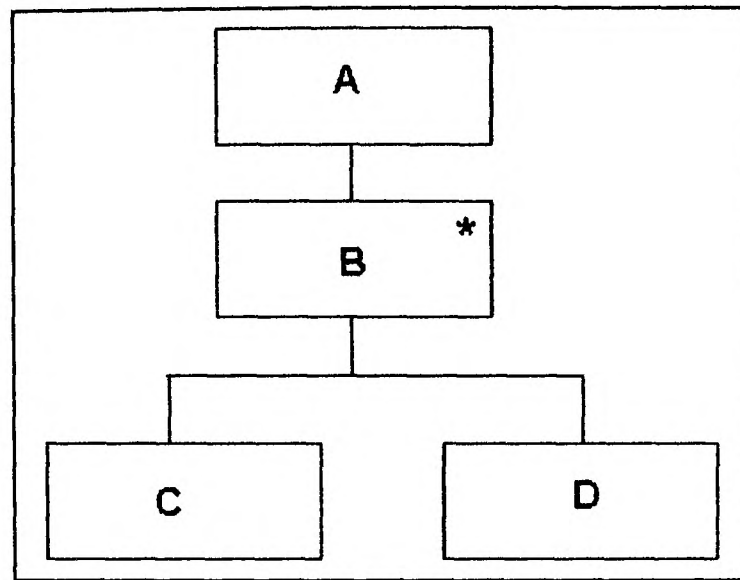


Figure 6.7: Iteration

In *Figure 6.8* (Weaver-02), **W** is an iteration of **B**'s.

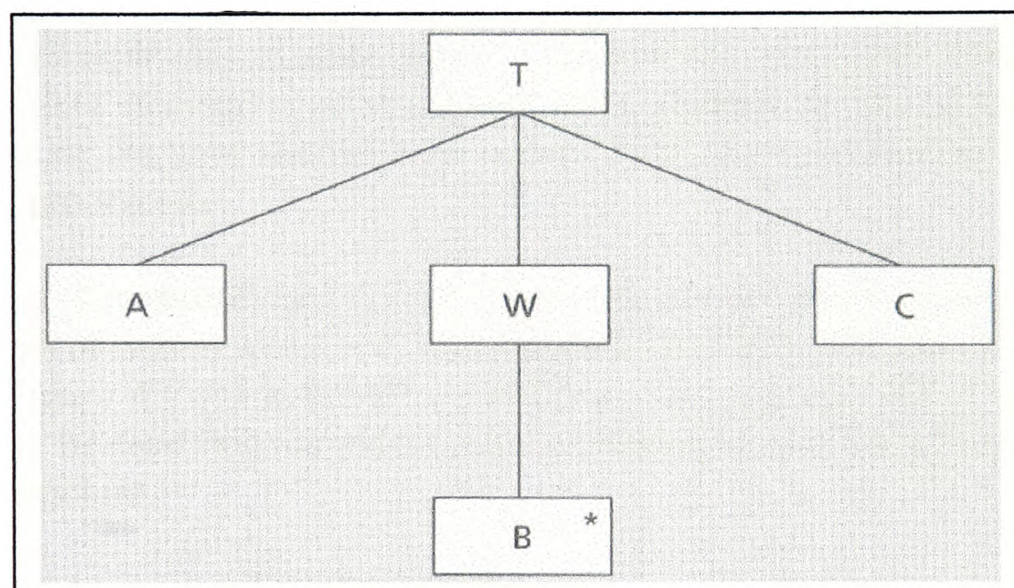


Figure 6.8: Iteration

6.5 Jackson Structure Rules

There are a number of basic rules that must be observed in the creation of Jackson-like structures:

- All boxes hanging from a single 'parent' box must be of the same type, i.e. all selections, or all sequence boxes (Weaver-98).
- A sequence must have only components without symbols in the next lower level (Ingevaldsson-90).
- A selection must have selection parts only in the next level below (Ingevaldsson-90). All children (two or more of them) of a selection must be boxes with an 'o' in the top right-hand corner (Weaver-02).
- All selection boxes must have a structure box above them (Weaver-98).

- An iteration must have only a single iterated component in the next lower level (Ingevaldsson-90).

6.6 Other Principles and Techniques

A meticulous study of various other principles and techniques from the fields of software engineering, programming and information systems has revealed that some of these techniques can be employed in the analysis, design and construction of spreadsheet models.

Indentation and Translation of Data Structure into Structured Form

Indentation is an important technique used in structured programming. The philosophy of structured programming, as outlined in (Dahl-72) promotes the indented form for code. This form has led to huge improvements in the comprehension of code, leading to improvements in productivity, auditing and maintenance (Knight-00). Later work (Jackson-75) proposed methods for the translation of data structure into structured form. Jackson proposed that the form of the data structure diagram should be extracted from the natural structure existing in the data to be processed.

Figures 6.9 (a) (Knight-00) and 6.9 (b) (Ingevaldsson-86) show examples of how the structured form of data is extracted from the data structure. The indented structure on the right of Figure 6.9 (a) is the structured programming equivalent of the structure diagram. It can be seen that the indentation is consistent with the levels of data within the Jackson structure.

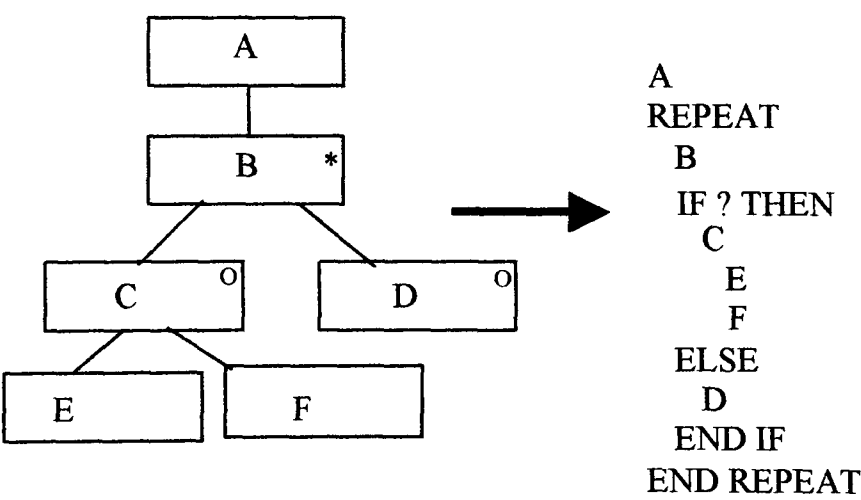


Figure 6.9 (a): Extraction of Indented Structured Form

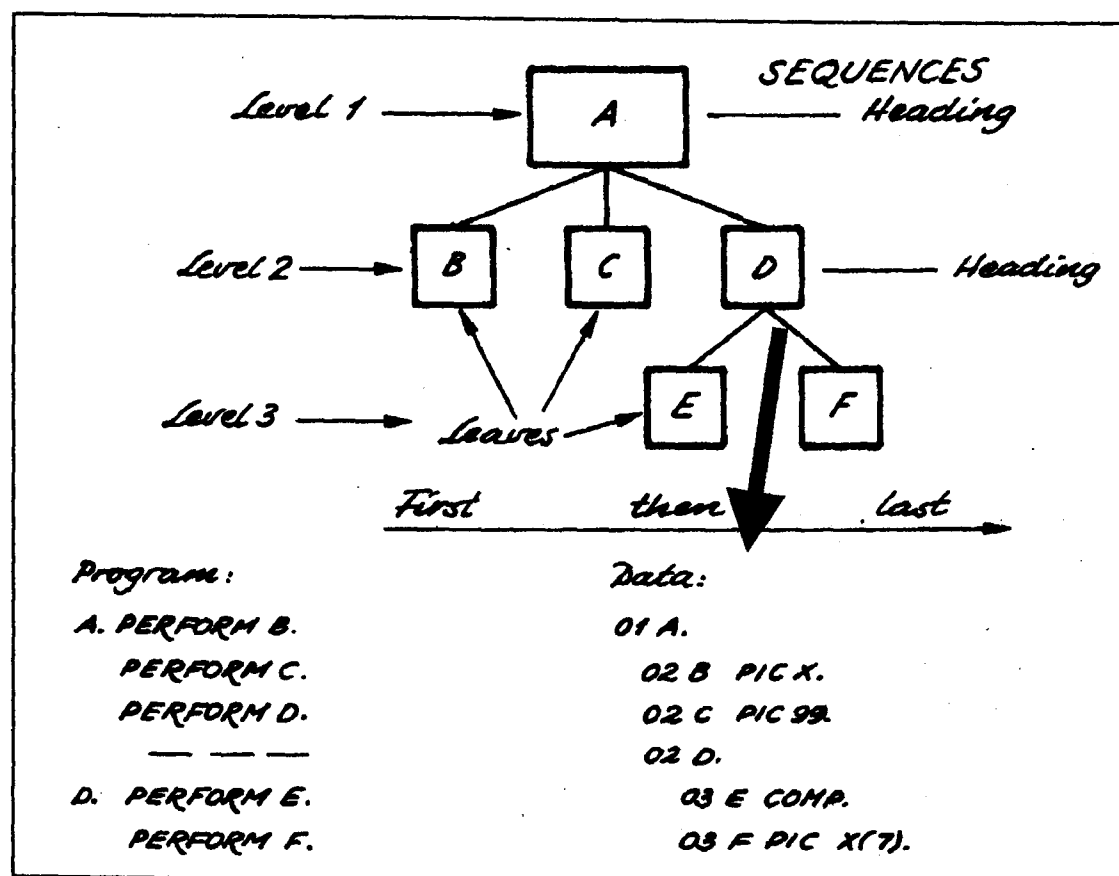


Figure 6.9 (b): Extraction of Indented Structured Form

Virtual Columns

The term 'virtual columns' is used as the multiple physical spreadsheet columns are viewed as a single *logical* column. As such, each row can only contain exactly one function or calculation. The formulae and inputs corresponding to their labels are entered in a set of (virtual) columns consistent with the indentation of the labels. They are located in different *virtual columns*, according to their position in the data structure. When these formulae and inputs appear in different 'virtual columns', the comprehensibility of the model is improved significantly. The precedents of each calculation can be easily identified.

Separation of Inputs, Model Schema (Workings/Calculations) and Outputs

According to Benham (Benham-93), the foundation for this separation is consistent with Sprague and Carlson's (Sprague-82) characterisation of decision support systems as having a data component, model component and user-interface/presentation component. Kee (Kee-88) proposes the use of a central data entry area to make data entry easier and to prevent input errors.

Modularisation

The concept of modularising software lies at the heart of software engineering methodologies. The idea of breaking down a complex piece of software into smaller, relatively isolated sub-components is an appealing one from many points of view. Maintenance, testing and de-bugging, re-use and estimation are all facilitated by modularisation.

Modularisation can be used as a mechanism for segmenting or decomposing a spreadsheet model into smaller parts. Each part is known as a module. Modularisation is the key to successful software engineering, allowing complex systems to be broken down into manageable sub-systems, for ease of comprehension and maintenance. Indeed, the basic principle guiding modularisation can be said to characterise different software engineering methodologies.

Object-oriented software engineering is characterised by Parnas's information hiding principle (Parnas-72), and Stevens, Constantine and Myers' structured approach (Stevens-74) is characterised by the concept of code cohesion. In the proposed structured spreadsheet development methodology, modules are defined by graphical properties of data structure diagrams.

6.7 Summary

Software engineering principles and methods, as well as their application to spreadsheet development, have been discussed in this chapter. This includes a detailed description of Jackson structural forms as the application of these structures is an essential part of the proposed methodology. Although there is no standard definition for *software engineering*, it is widely accepted that software engineering is a systematic approach that encompasses all aspects, stages and activities involved in the development of software systems. Spreadsheet development can also adopt a systematic approach that covers all stages of the spreadsheet building process.

Among the main software engineering principles that can be applied to spreadsheet development include the development of appropriate methods, tools and techniques, precise requirements definition, formal specification of requirements, greater focus on quality control, and adopting a top-down approach. The normal stages of the software life cycle are *specification, analysis, design, implementation, testing and operation and maintenance*. The main software development process models include the *waterfall model, prototyping, formal methods* and the *spiral model*.

The adoption of structured systems development techniques has been widely proposed to effectively deal with the problem of spreadsheet errors. In systems development, among the main methods developed include *Jackson Structured Programming, Entity Relationship Modelling, the Warnier-Orr method* and *Object-oriented methods*. From an investigation of the suitability of these methods to spreadsheet development, *Jackson structural forms* based on *Jackson Structured Programming* has emerged as the most desirable method. This is mainly due to its maturity, simplicity, relevance and practicality. The Jackson method is far more likely to be accepted compared to the other methods, which require relatively high spreadsheet user skills.

The three key constructs of *sequence, selection and iteration* form the basis of Jackson structural forms based on *Jackson Structured Programming*. There are also certain basic rules that must be followed when developing Jackson structures. Other important principles and techniques that can also be employed in the development of spreadsheet models include *indentation and translation of data structure into structured form, virtual columns, separation of inputs, workings and outputs, and modularisation*.

CHAPTER 7

THE PROPOSED STRUCTURED METHODOLOGY

7.1 Introduction

Chapter 6 provided an understanding of related software engineering concepts and principles, and their potential application to the design and development of spreadsheet models. The principal method focused upon was the use of *Jackson structures*.

Based on the software engineering principles and structured techniques investigated, a comprehensive structured methodology for the construction and integrity control of spreadsheet models has been developed. This chapter presents the proposed methodology in detail. It begins by discussing the development and synthesis of the methodology from the material considered in *Chapter 6*. The various stages of the methodology are described in detail with suitable examples. The methodology's potential for quality improvement is also discussed.

The proposed structured methodology represents a significant development or advance in the research into the development and integrity control of spreadsheet models. Preliminary versions of the methodology are presented by Rajalingham, Chadwick, Knight and Edwards (Rajalingham-01,02; Knight-00; Chadwick-99).

The proposed methodology imposes a strict discipline in the process of spreadsheet development using software engineering principles. This reduces the occurrence of errors as spreadsheet models are designed and constructed in a structured and organised manner. The methodology distinctly describes a technique for modelling the spreadsheet problem and subsequently mapping the design onto the physical spreadsheet according to prescribed rules and a structured algorithm.

7.2 Development and Synthesis of the Proposed Methodology

This section provides an account of the development of the proposed methodology and its synthesis from the material considered in the previous chapter, *Chapter 6*.

7.2.1 General Software Engineering Principles

Based on the discussion of software engineering principles in *Chapter 6*, it has been found that many of these principles are applicable to the development of spreadsheet models. Therefore, these principles have been incorporated into the proposed methodology. It has been established that software engineering is a systematic approach that encompasses all aspects, stages and activities involved in the development of software systems. The proposed methodology adopts a systematic and organised approach that covers all stages and activities of the spreadsheet building process. *Figure 7.1* shows the relationship between the proposed structured methodology and the normal stages of the software development life cycle given in

Chapter 6. The different stages of the proposed methodology are described in detail in the next section, Section 7.3.

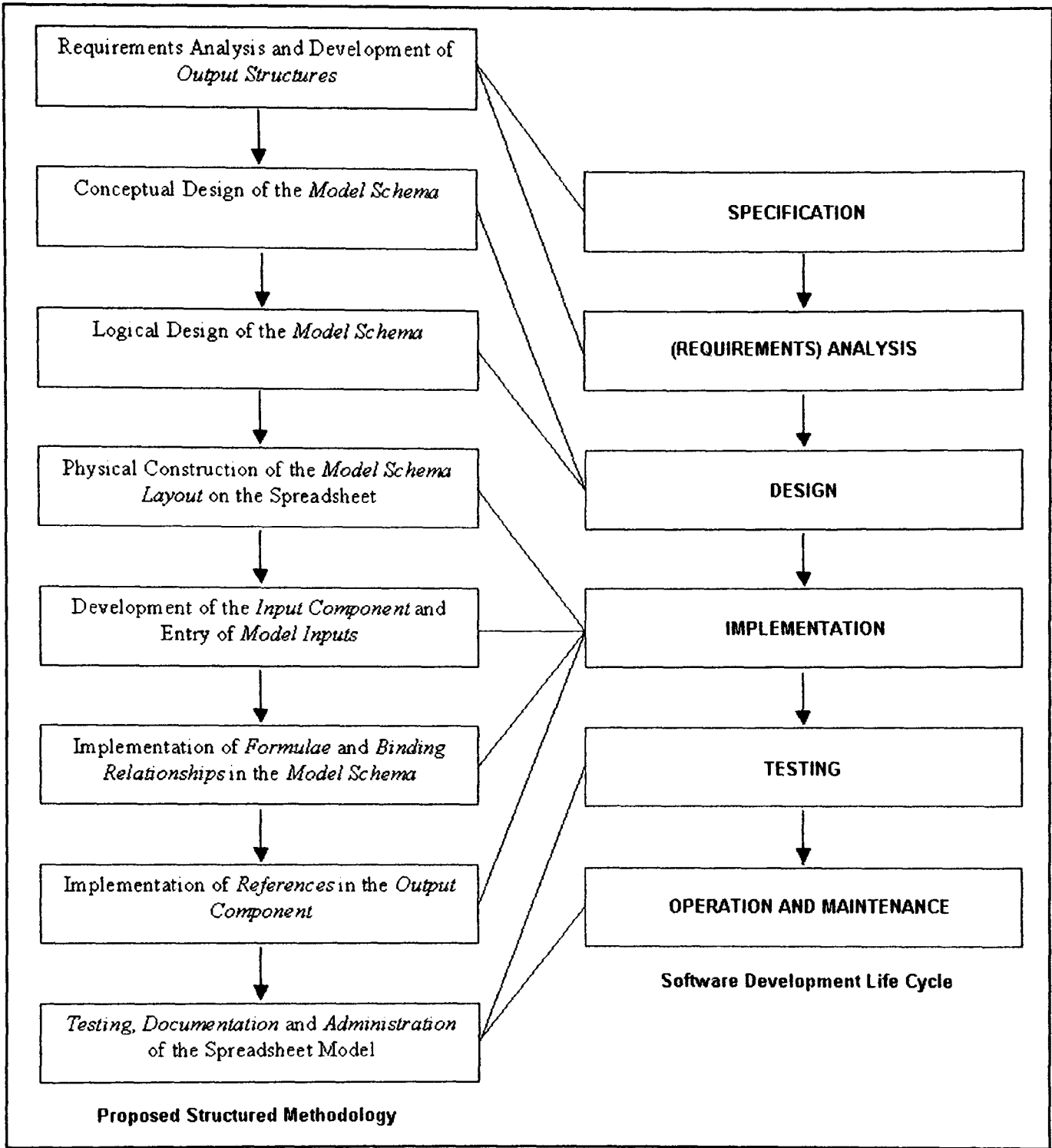


Figure 7.1: The Proposed Methodology and the Software Development Life Cycle

Various other principles of software engineering discussed in the previous chapter have also been used to develop the proposed methodology. Appropriate tools and techniques are used within the methodology as software engineering is concerned with the selection and development of the most appropriate methods, tools and techniques used for producing software. These include *Jackson structural forms, indentation and translation of data structure into structured form, virtual columns, separation of inputs, workings and outputs, and modularisation*. The proposed methodology also includes models, notations, rules and design advice. Techniques such as Jackson structures are used to produce a graphical representation of the spreadsheet model as a basis for specification or design.

Among the other important software engineering principles used to develop the proposed methodology are as follows:

- An emphasis on eliciting and defining the exact requirements of users in *Stage 1*.
- Formal specification of the requirements of the spreadsheet system in *Stage 2* and *Stage 3*.
- Focus on a high-level view or broad picture first, followed by a look at successive smaller parts in greater detail in *Stage 1*, *Stage 2* and *Stage 3*.
- Greater emphasis on quality control and eliminating errors in all stages of the methodology.

7.2.2 Application of Jackson Structures

The suitability of a front-end of the proposed methodology for spreadsheet development, based on the Jackson structural forms (described in *Chapter 6*) has been investigated. It has been found that the conceptual or logical design of spreadsheet models can be represented in a form identical to a *Jackson structure*. This technique is used in *Stages 2* and *3* of the proposed methodology.

When Jackson structures are used to represent the logical design of a spreadsheet model, they can distinctly show all the relationships within the model's schema. As described in *Chapter 6*, Jackson tree structures are based on three key constructs: *sequence*, *selection* and *iteration*. These constructs can show the sequence, optionality and iteration of data items. The three constructs of Jackson structures are also applicable to the design of a spreadsheet model.

Sequence

Referring to *Figure 7.2 (Staff Budget)* based on Chadwick et al (Chadwick-97), there is a need to calculate the *average staff wages*.

	Number of Staff	Day Wages £	Night Wages £
Grade 1	1	17700.50	0.00
Grade 2	3	45540.00	1400.55
Grade 3	9	122340.00	2000.00
Grade 4	12	102350.25	0.00

Figure 7.2: Inputs for the Staff Budget Model

The formula is:

$$\text{average staff wages} = \text{total wages} / \text{total number of staff}$$

Total wages and *total number of staff* are therefore direct precedents of *average staff wages*. *Total wages* being one of the operands of the formula is made up of *total day wages* and *total night wages*. The formula is:

$$\text{total wages} = \text{total day wages} + \text{total night wages}$$

Based on this analysis, a partial Jackson structure can be constructed, comprising a set of hierarchical sequences. This is presented in *Figure 7.3*.

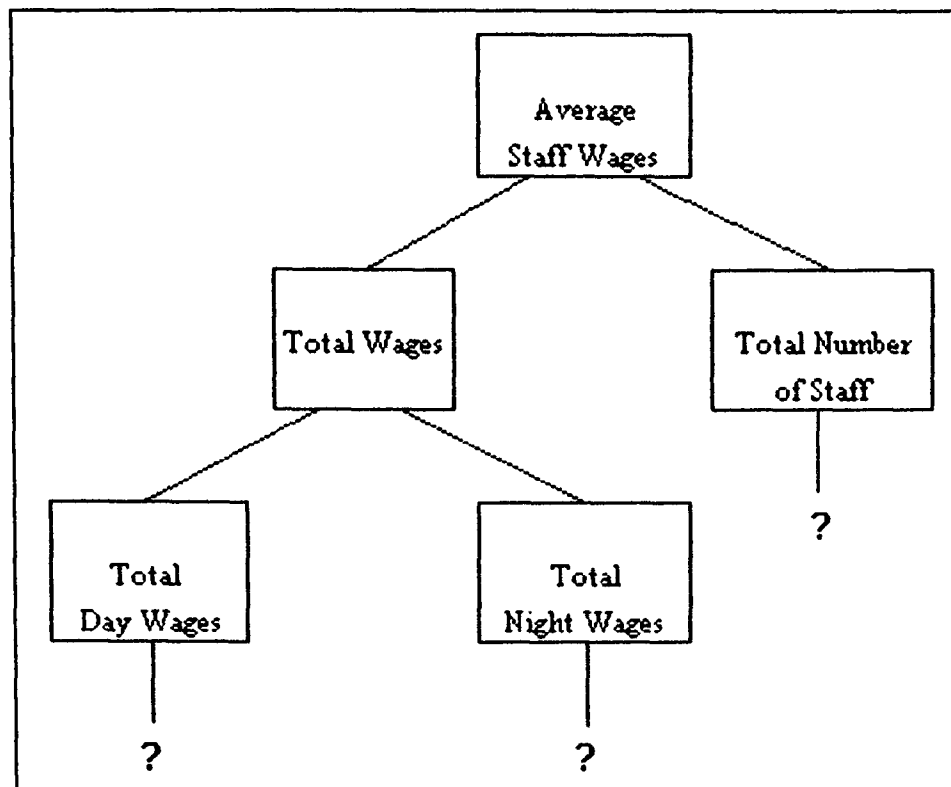


Figure 7.3: Sequences

Selection

This feature of Jackson structures can be used in spreadsheet models to represent mutually exclusive sets of direct precedents for a particular formula. For the purpose of clarity, appropriate conditions can be attached to selection structures.

In the calculation of *tax*, the formula is:

IF *taxable profit* > 0
 THEN *tax* = *taxable profit* * *tax rate*
 ELSE *tax* = 0

In a spreadsheet cell, the corresponding formula for tax would be written in the form:
 = IF (taxable profit > 0, taxable profit * tax rate, 0)

In either case, *taxable profit* and the constant *zero* would be direct precedents of *tax*. The operands forming the condition within the formula are mandatory precedents. As such they are represented using sequence boxes.

Additionally, depending upon the value of *taxable profit*, *tax* would have either *taxable profit* and *tax rate*, or *zero*, as its direct precedent(s). This part of the formula can be shown using a selection structure. This would be represented in the form of a Jackson structure as displayed in *Figure 7.4*.

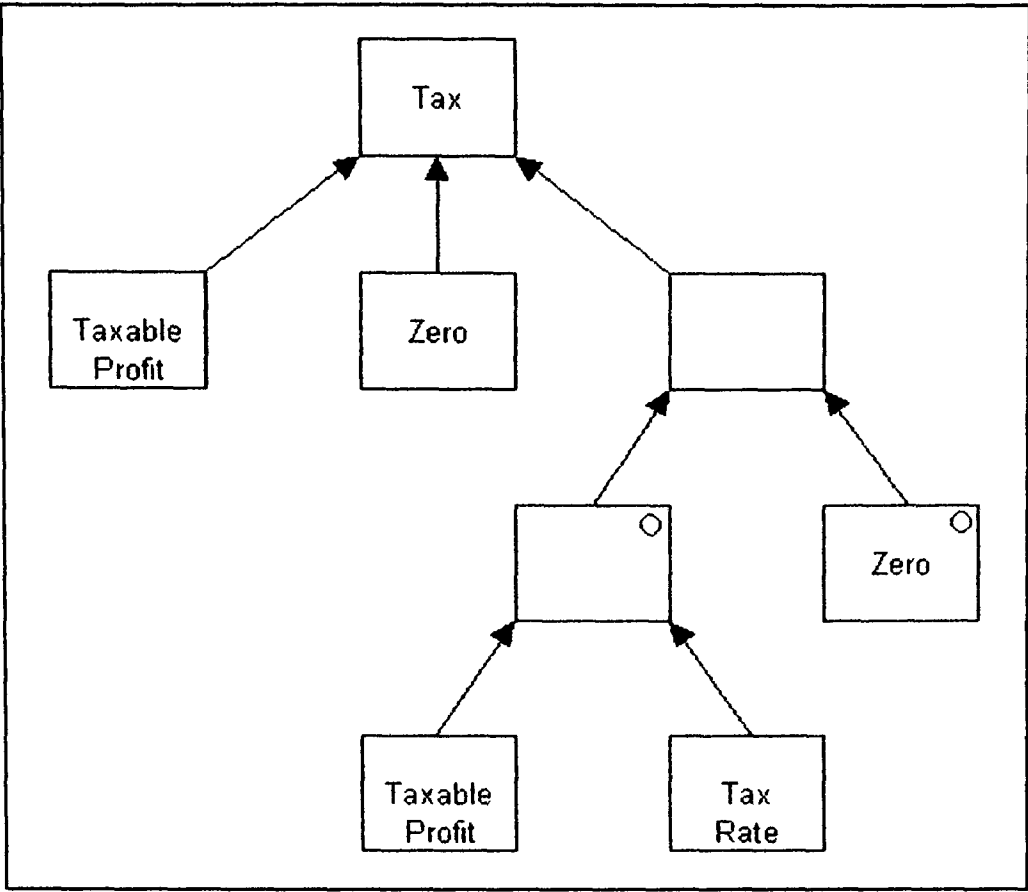


Figure 7.4: Selection

Iteration

Within the context of spreadsheet models, iterations in Jackson structures can be used to show parts of a model that may repeat several times. An iterated component represents multiple instances, where each instance corresponds to a different time period, group, category, etc.

Based on the Staff Budget model example, the average wage for each grade is also required. The formula to calculate this for each grade is exactly the same. This part of the logical design is shown in *Figure 7.5*.

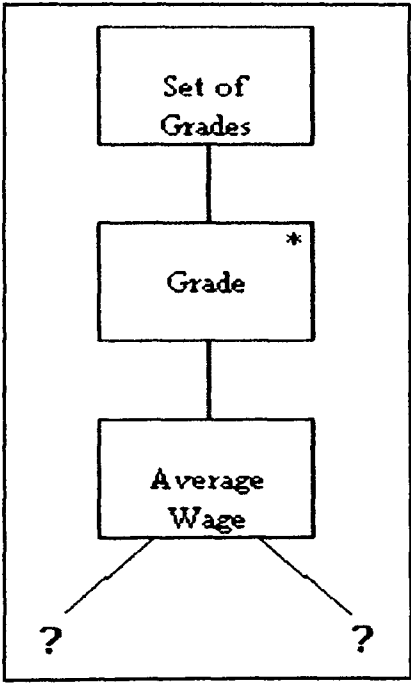


Figure 7.5: Iteration

It has also been identified that *total day wages* is defined as the sum of *day wages* for each grade while *total night wages* is the sum of *day wages* for each grade.

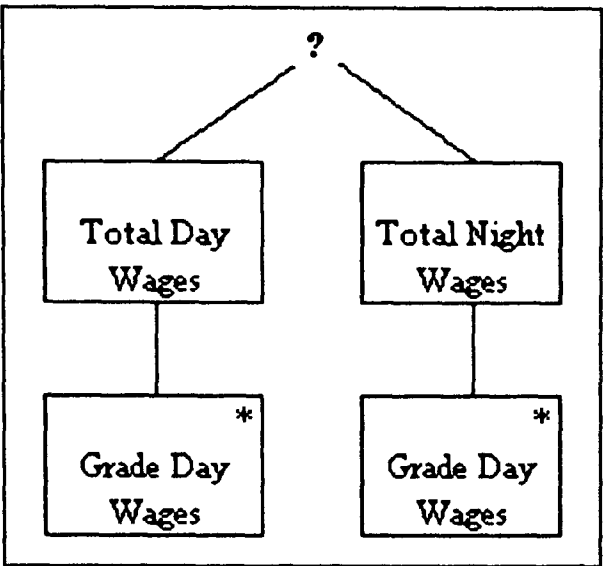


Figure 7.6: Iteration

This part of the model can now be incorporated into the sequence structure shown in Figure 7.3 and the iteration structure in Figure 7.5. The resulting Jackson structure is displayed in Figure 7.7. This structure represents the logical design of the entire model.

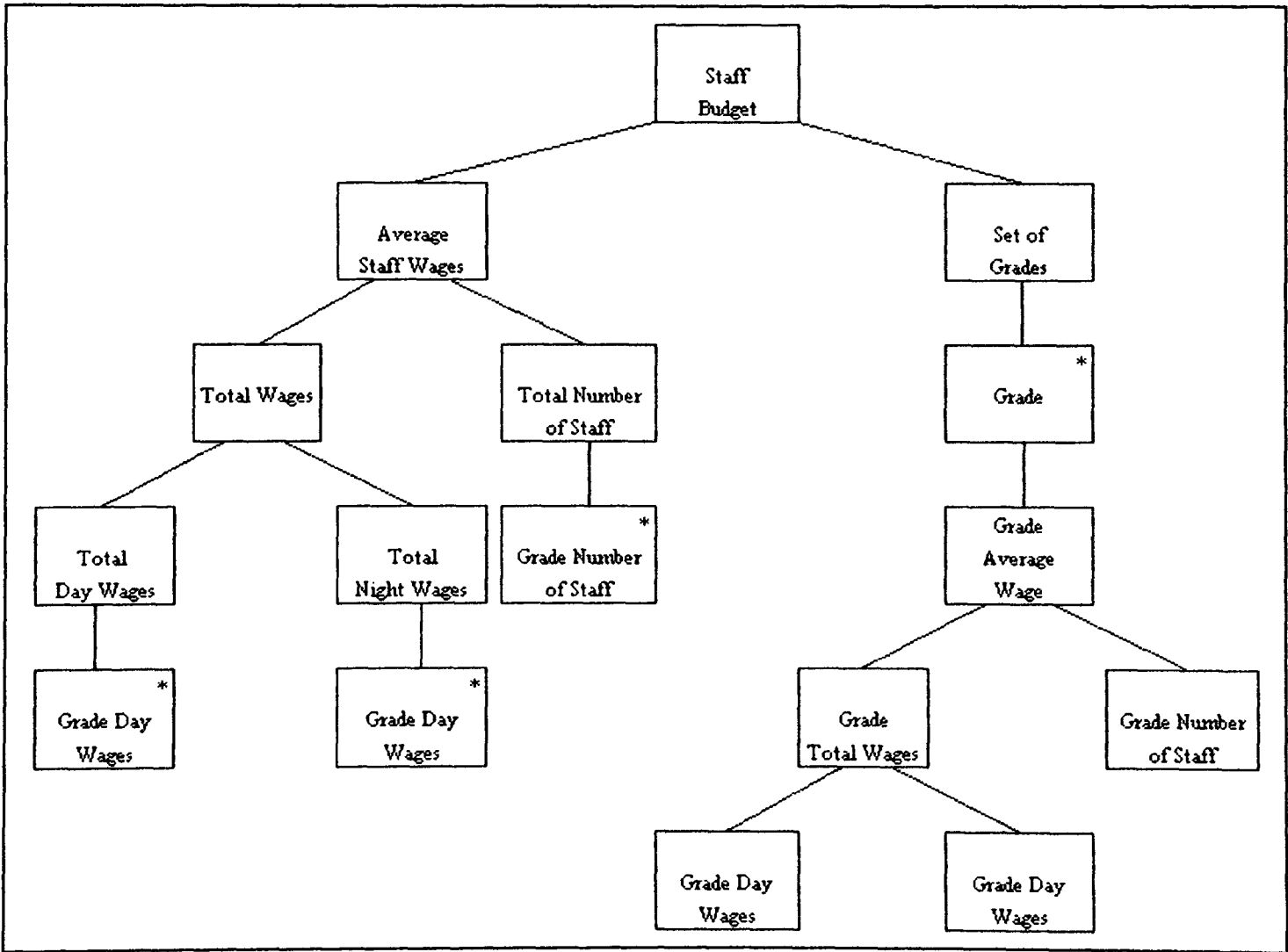


Figure 7.7: Logical Design

7.2.3 Other Principles and Techniques

There are other software engineering and programming principles and techniques described in *Chapter 6*, that have also been used to further develop and enhance the proposed methodology.

Indentation and Translation of Data Structure into Structured Form

Jackson (Jackson-75) has shown that there is direct correspondence between data and program structures, and that *structure diagrams* can be directly mapped onto the corresponding program code. This technique is used in *Stage 4* of the proposed methodology, to translate the logical design represented in the form of a Jackson structure into a structured spreadsheet. This is illustrated using the example of the *Staff Budget* model. *Figure 7.8* shows the extraction of the structured spreadsheet from part of the logical design produced earlier (displayed in *Figure 7.7*). The indented structure on the right of *Figure 7.8* is the structured programming equivalent of the structure diagram.

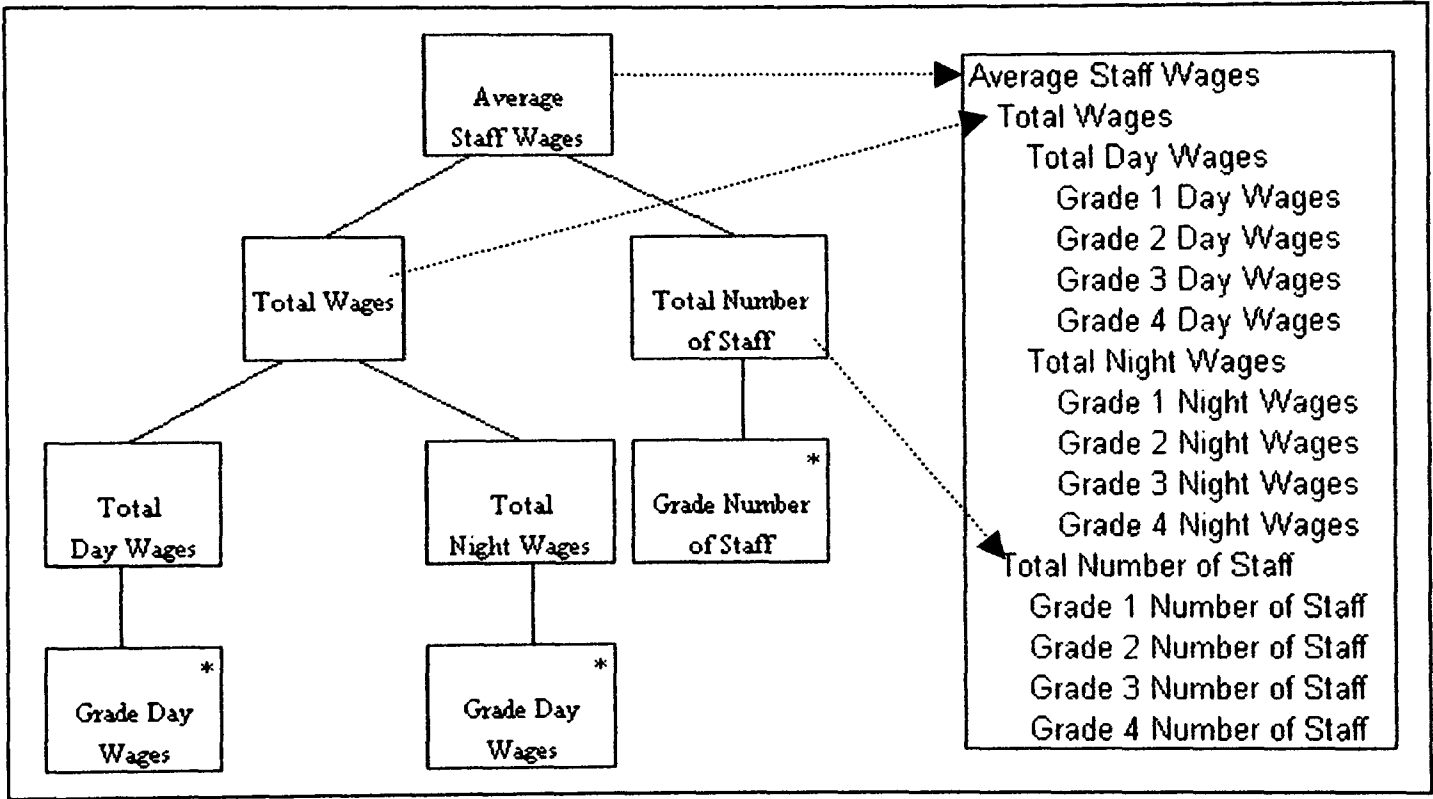


Figure 7.8: Translation of Jackson Structure to Structured Form

Virtual Columns

This technique is applied in *Stage 4* and *Stage 6* of the proposed methodology. The positioning of formulae and inputs in virtual columns is demonstrated in *Figure 7.9*, based on the *Staff Budget* model.

Average Staff Wages	11,653.25		
Total Wages		291,331.30	
Total Day Wages			287,930.75
Grade 1 Day Wages			17,700.50
Grade 2 Day Wages			45,540.00
Grade 3 Day Wages			122,340.00
Grade 4 Day Wages			102,350.25
Total Night Wages		3,400.55	
Grade 1 Night Wages			0.00
Grade 2 Night Wages			1,400.55
Grade 3 Night Wages			2,000.00
Grade 4 Night Wages			0.00
Total Number of Staff	25		
Grade 1 Number of Staff		1	
Grade 2 Number of Staff		3	
Grade 3 Number of Staff		9	
Grade 4 Number of Staff		12	

Figure 7.9: Virtual Columns

Separation of Inputs, Model Schema (Workings/Calculations) and Outputs

This principle or technique is adopted as an essential part of the proposed structured methodology. DiAntonio's method (DiAntonio-86) also advocates the segregation of facts by dividing the spreadsheet into two parts, *facts* (*input*) and *solution* (*schema/calculations* and *output*). The *Staff Budget* model example is used to show the three divisions of the spreadsheet model: *input*, *schema/calculations* and *output*. This is illustrated in *Figure 7.10*.

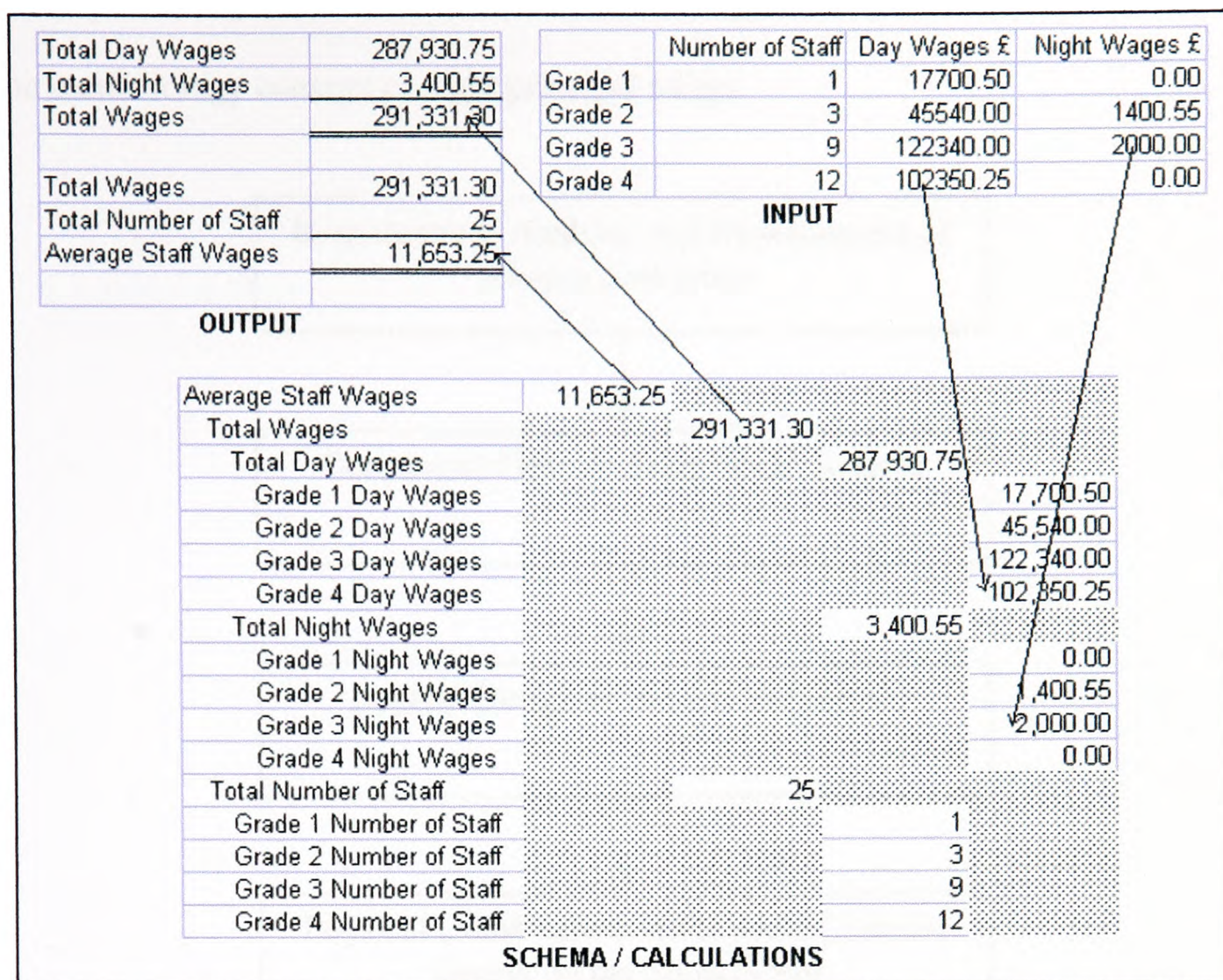


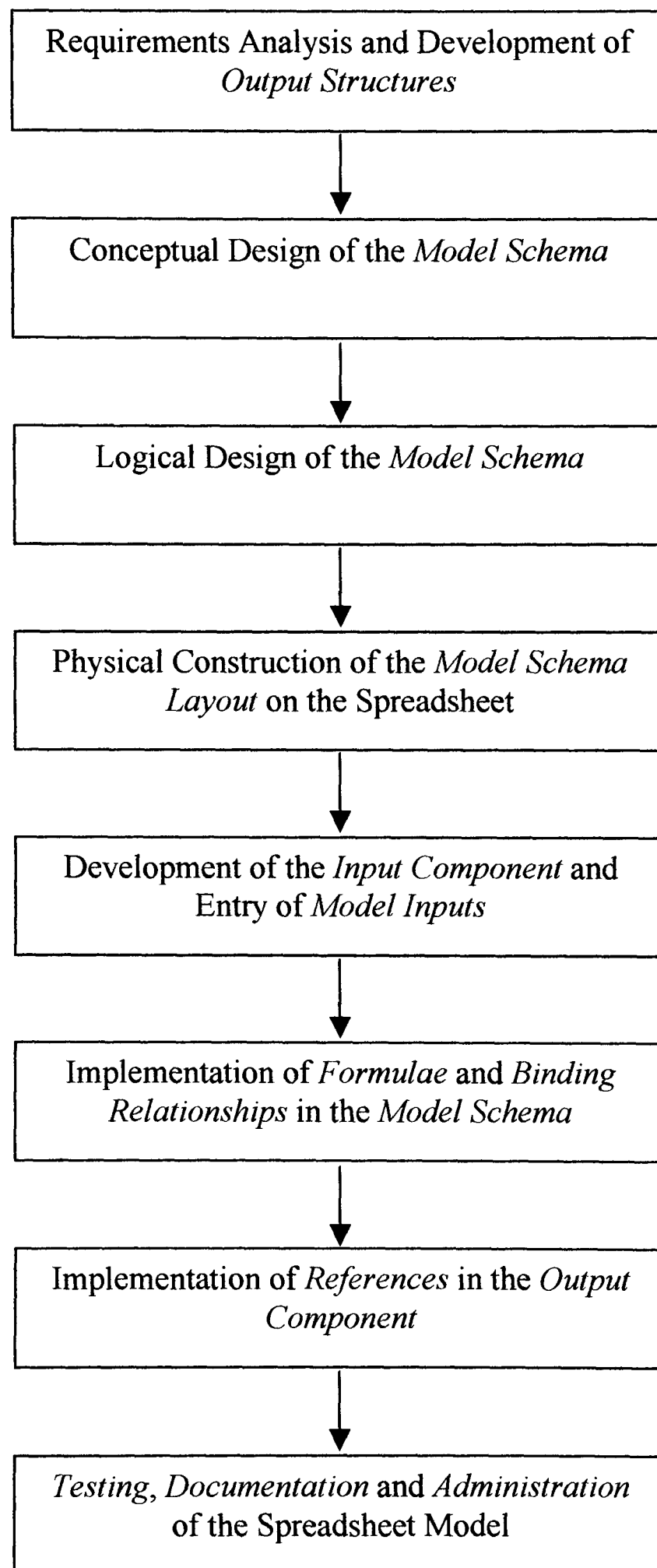
Figure 7.10: Separation of Input, Workings and Output

Modularisation

The principle of *modularisation* is used within the proposed methodology. In *Stage 3* and *Stage 4*, rules are formulated to systematically segment or decomposing the spreadsheet model into smaller parts or modules. This is more elaborately explained in the next section, *Section 7.3*.

7.3 The Proposed Structured Methodology

The methodology consists of eight principal stages:



STAGE 1:

Requirements Analysis and Development of Output Structures

This stage is carried out from the perspective of the *model sponsor(s)* or *interpreter(s)*. The model sponsor is the person who requests that the model be built and ensures the required resources are available. Agreement of the objectives of the model is the responsibility of the model sponsor (Read-99). The model interpreters are the end-users who interpret or use the output of the spreadsheet model for a particular purpose or to make business decisions.

The first stage comprises two steps:

- **Step 1: Requirements analysis and specification**
- **Step 2: Design of outputs and development of output structures**

In *Step 1*, the requirements of the model sponsors or interpreters are elicited and analysed. The overall objective or purpose of the spreadsheet model is also established. Based on the information gathered, an assessment of the nature, scale and complexity of the model is carried out. Read and Batson (Read-99) have defined a set of tasks under the *scope* stage of their *Spreadsheet Best Practice Methodology*, some of which are appropriate for application in this step. The model developer(s) have to

- decide what needs to be included in the model and what can be omitted;
- understand in outline how the model will work;
- estimate the time and resource required for the model development; and
- agree the above with the key stakeholders.

Step 2 of this stage involves translating the requirements of the model sponsors/interpreters into a set of spreadsheet model outputs. Each spreadsheet model would normally have one or more associated outputs. The methodology insists on the presentation of outputs on one or more separate worksheets. They should neither appear in the worksheet containing the spreadsheet model schema, nor the worksheet containing the model inputs.

The structure of each output is designed and implemented on the physical spreadsheet. Only the editorial aspects of each desired output are implemented at this stage. These include titles, headings and descriptive labels for formula and data. Each desired output of the spreadsheet model is designed from the perspective of the model sponsors/interpreters.

There is a need to distinguish between inputs (or numeric constants) and formulae in the model's output(s). Having determined the various formulae required, the underlying logic of each formula calculation and its domain are defined. This is independent of any particular implementation platform. At the end of the methodology, references to the *model schema* and *model inputs* are added to the output component of the spreadsheet model. No calculations would be present in the outputs. However, there could be multiple outputs presenting the same information at different levels of detail or even in different layouts, to suit a variety of purposes.

The model sponsor/interpreters do not normally make changes to the outputs when the spreadsheet model is in operation. However, they may alter the structure or format of

the outputs if deemed necessary. This will not affect the integrity of the underlying spreadsheet model, which is embedded in the *model schema*.

STAGE 2:

Conceptual Design of the Model Schema

The *model schema* represents the workings or calculations component of the spreadsheet model. The purpose of constructing the *model schema* is to systematically and methodically perform the interim and final calculations based on the required or desired model output(s). An essential characteristic of the proposed structured methodology is the separation of inputs, calculations and outputs. The model schema, representing the spreadsheet model's underlying logic, is therefore separated from the inputs and outputs. From a physical perspective, the model schema is created on a separate worksheet.

In developing the conceptual model, the first step is to distinguish between inputs and formulae contained within the model output(s). An analysis of all formulae is carried out in order to construct the conceptual model.

The main steps involved in this stage are as follows:

- **Step 1: Determine the operands of each output formula**
- **Step 2: Establish relationships between formulae at a logical level**
- **Step 3: Identify root formulae or formulae with no dependants**
- **Step 4: Use a Jackson structure to represent the direct and indirect precedents of each root formula**
- **Step 5: Merge the structures of all the root formulae**

The first step involves determining the operands of each output formula. This step is carried out as a means of determining all root formulae appearing in the outputs. A root formula is defined as a formula that has neither direct nor indirect dependants. They are therefore not referenced by any other formula within the spreadsheet model.

The conceptual design of each root formula is represented in the form of a *Jackson* structure (Jackson-75). In a large number of spreadsheet models, it is highly possible that there is just one root formula.

Each node of a sequence or selection (depending on its position in the Jackson structure) represents either a formula or a piece of data. If the node is a *structure element* or *leaf* (or *end-leaf*), it represents data (numeric constant) or input. An iterated component that is not a leaf represents a structure (or sub-structure) that can occur zero or more times. Such iterated components have special properties, which will be discussed later. If an iterated component is shown as an end-leaf in the Jackson structure, it represents a set or range of inputs that is always manipulated or operated as a group rather than individually.

The root formulae are placed at the top of the Jackson structure, hanging from a box containing the title of the spreadsheet model. The direct precedents of each root formula are then positioned immediately below it, adjacent to each other. Each node is decomposed step by step, until every end-leaf or bottom node has been identified and

represented. The conceptual design of the entire model schema is the combination of the structures of all root formulae into a single Jackson structure with its root node containing the title of the spreadsheet model.

When a top-down approach is adopted without showing duplication of nodes, the structure of the model schema could take the form of a graph instead of the desired tree structure. The purpose of this is to distinctly show instances of multiple dependants of a particular formula of the model schema. This potentially results in a structure as shown in *Figure 7.11*.

From the structure in *Figure 7.11*, we can observe the following points:

- **A** is a *root formula*. It therefore represents a formula with no dependants.
- **D** and **E** are mutually exclusive (due to the *selection* constraint) precedents of **A**.
- The direct precedents of **D** are a *sequence* of **F** and **G**.
- The direct precedents of **K** are a *sequence* of **L** and **M**.
- **M** is a function of zero or more *iterations* of **N**.
- **N** is a range of zero or more related inputs (constants).
- **B**, **C**, **F**, **H**, **J** and **L** are *leaves* as they do not have any precedents. This shows that they are inputs (or constants). **B**, **C**, **F**, **H**, **J** and **L** are therefore read or referenced from the *input component*, which will be constructed later.
- **G** has two dependants, **D** and **E**, and therefore forms a graph.
- **K** also has multiple dependants, **E** and **I**; another graph is formed.

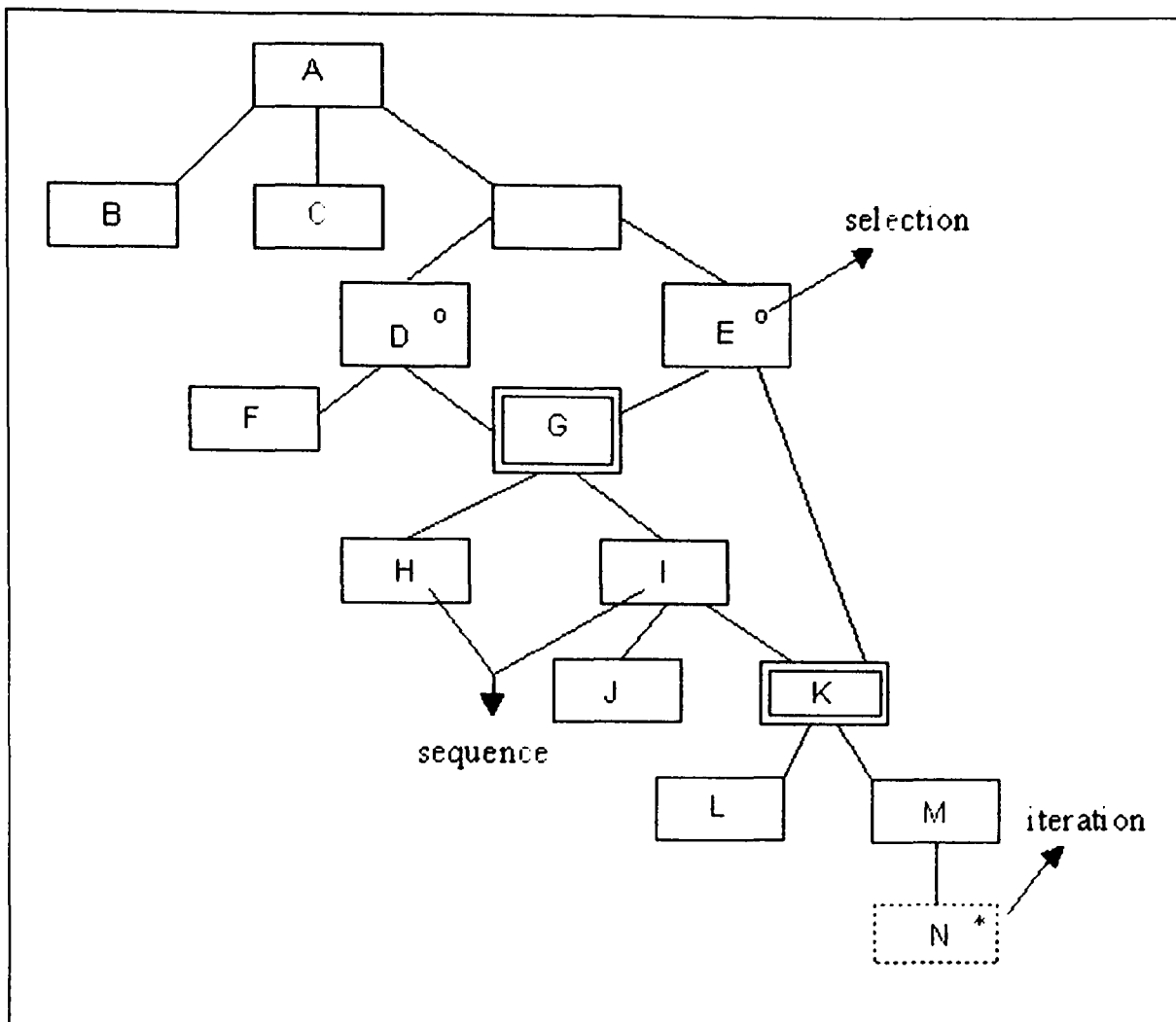




Figure 7.11: The Conceptual Design in Graph-form

A double-line box  (as opposed to the single-line box) represents a formula or data (numeric constant) with multiple dependants. A dashed box  represents a range of related inputs (constants) treated and manipulated as a set. In order to effectively model the conceptual and logical designs of a spreadsheet model, these notations have been added to the conventional Jackson notations (Jackson-75) borrowed from software engineering.

STAGE 3:

Logical Design of the Model Schema

The logical perspective consists of a formal and implementation-free description of the model's logic and data structures (Isakowitz-95). The purpose of *Stage 3* is to resolve sub-structures with formulae or data with multiple dependants. A formula or data with multiple dependants normally form a graph. Structurally, the aim at this stage is to transform all *graph sub-structures* in the conceptual model to *trees* so that the entire model is in the form of a Jackson-like tree structure. From a more logical perspective, the objective of performing this task is to enable the direct mapping of the Jackson structure to the spreadsheet based on Jackson's method of mapping the data structure diagram to a computer program.

Figure 7.11 shows an example of a generic conceptual design containing graph sub-structures. For instance, there is a loop in the relationships connecting E, G, I and K, so that we no longer have a tree form. In this chart, K is a precedent of both E and I. We can turn the graph into a tree-structure. In order to accomplish this, two important steps prescribing the rules have to be observed:

Step 1

Each node or sub-structure with multiple dependants is duplicated and each copy is assigned as a direct precedent of every dependant of that node. Nodes with multiple dependants can be easily identified from the conceptual design as they are represented by double-line boxes. This is illustrated in Figure 7.12.

By performing this task, the graph structure is resolved into a tree-structure. However, in order to prevent multiple occurrence of the entire sub-structure, only the root node of each duplicated sub-structure appears in the logical design of the model at this point. Their precedents are therefore not included in the model.

Based on Figure 7.12, G and K are duplicated in order to resolve the graph structure, into a tree structure. The precedents of G and K are not included in the model. K is not even shown as a precedent of G in order to comply with the rule that precedents of duplicated nodes are not included in the main structure of the logical design.

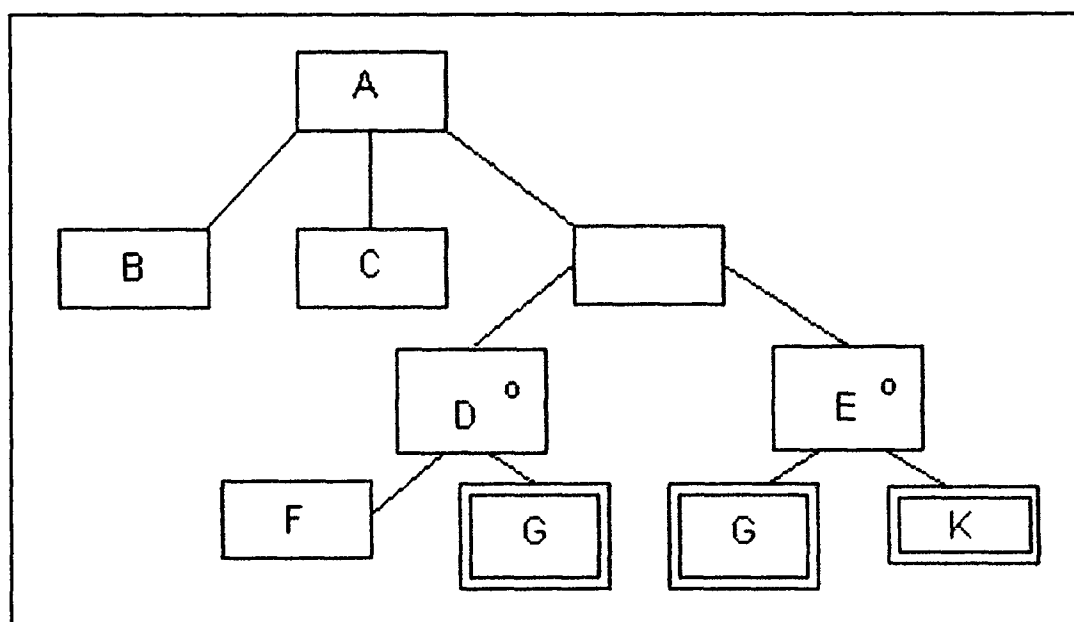


Figure 7.12: The Logical Design of the Main Structure Based on Step 1

Step 2

If a duplicated node has precedents (and therefore forming a sub-structure or branch), a distinct structured module is created, the logical design of which is represented by a separate Jackson tree structure. The structured module consists of the duplicated node as its root node/formula and the precedents of the particular formula. The structures resulting from the application of this step/rule are illustrated in Figures 7.13 (a) and 7.13 (b).

If the duplicated node is a *leaf* and therefore has no precedents, there is no need to define it as a separate module. As a rule, only a node or formula with precedents can be defined as a common module.

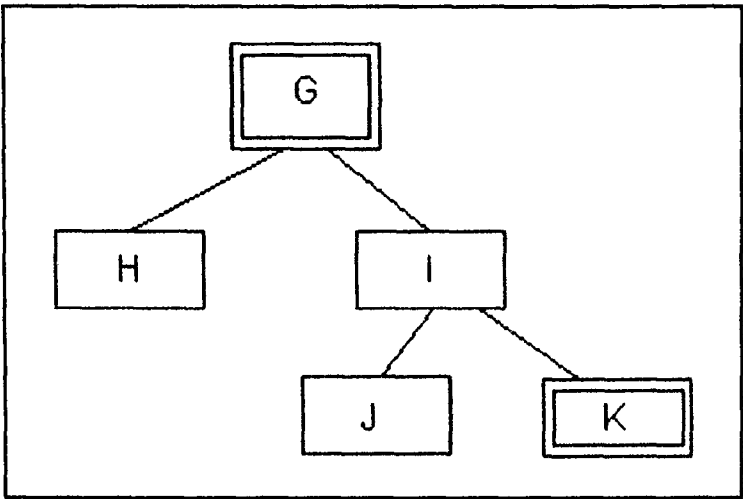


Figure 7.13 (a): The Logical Design of Module **G** (Based on *Step 2*)

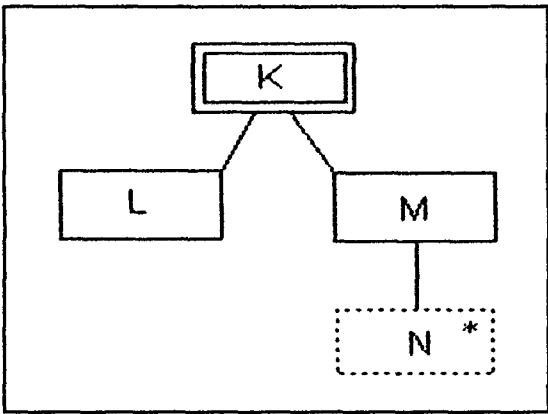


Figure 7.13 (b): The Logical Design of Module **K** (Based on *Step 2*)

Based on *Figures 7.13 (a) and 7.13 (b)*, the sub-structures **G** and **K** are defined as separate modules, each of which will occur once in the implemented spreadsheet model. This is discussed more elaborately in *Stage 4*.

The conceptual design shown in *Figure 7.11* has now been transformed into a logical design consisting of three modules, represented by three separate Jackson structures. The modules consist of a main or primary module and two secondary modules. *Figure 7.14* shows the relationship between the modules.

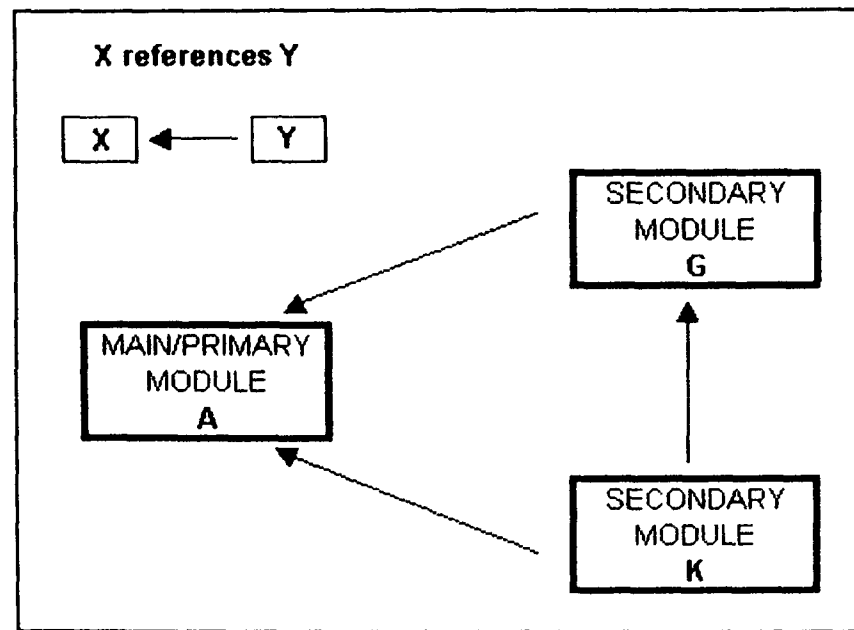


Figure 7.14: Relationship Between Modules

In general, we can always reduce a graph structure to a tree by this method, which conveniently produces a unique modularisation of the spreadsheet model.

STAGE 4:

Physical Construction of the Model Schema Layout on the Spreadsheet

The logical design of the model (represented as Jackson tree-like structures) is systematically mapped onto the physical spreadsheet based on rigorous rules prescribed by the methodology.

To maintain the structure modelled in the logical design in the spreadsheet view, the indentation principle is used, both on the row labels and on the corresponding values themselves. The values are indented by assigning a spreadsheet column to each level of indentation. These columns can be referred to as *virtual columns*. Based on the generic logical design shown in *Figures 7.12, 7.13 (a) and 7.13 (b)*, the corresponding structure of the spreadsheet view at this stage is shown in *Figure 7.15*.

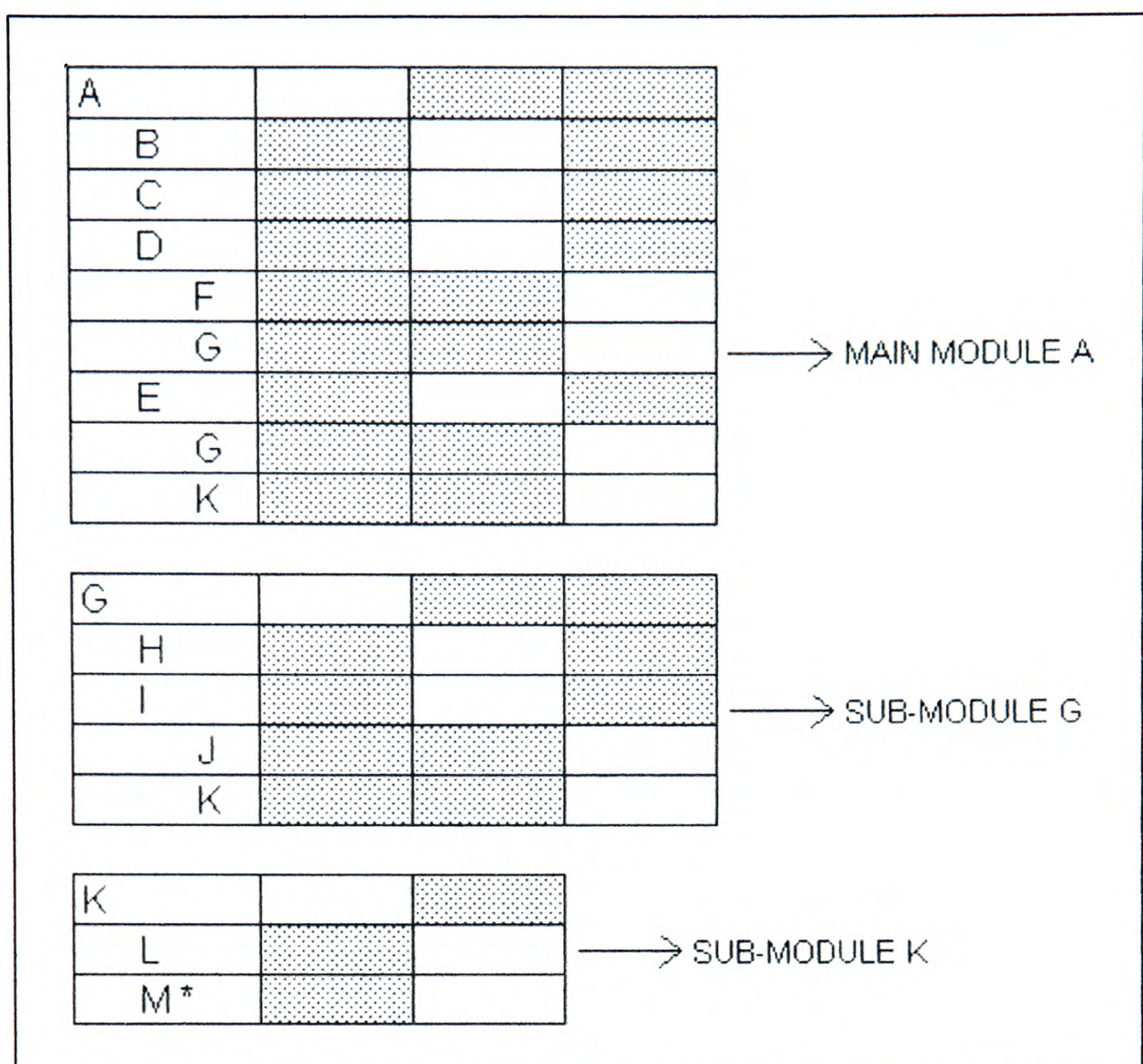


Figure 7.15: Model Schema Layout

Based on *Figure 7.15*, **D** is a function of **F** and **G**, while **G** is a function of **H** and **I**. Therefore, the precedents of **Formula_D** are **F** and **G**, whereas the precedents of **Formula_G** are **H** and **I**.

STAGE 5:

Development of the Input Component and Entry of Model Inputs

It is important to control data integrity and maintain the consistency of data in information systems. A frequent error committed by users is the accidental overwriting of formulae. This is usually due to the fact that data values (or inputs) and the formulae that reference them are placed in close proximity to each other and users are sometimes unable to distinguish between the two.

Data input represents a special problem in spreadsheet design, with its own special requirements. The technique proposed to overcome this problem is to put all inputs in a separate worksheet. This is called the *input section*. This strategy is similar to the method introduced by DiAntonio (DiAntonio-86). DiAntonio's method advocates the isolation of *facts* by splitting the spreadsheet into two parts, one for the *facts* and one for the *solution*. DiAntonio's *facts* part corresponds to our *input* component.

The end-users responsible for data entry enter data in the input section only. The elements of the input section are based on the *leaves* of the Jackson structures. The

input component contains all *data* and *assumptions* used in the spreadsheet model. It is not always necessary to explicitly separate the two. Benham (Benham-93) recommends that this section be partitioned into *decision variables*, *environmental variables*, and *parameters*.

The design of this part of the user interface should be as free from constraints as possible; so as not to hinder the main objective: ease of use and absence of data errors. We are therefore, quite at liberty to put all data input cells into unstructured modules, since there are never any dependencies between them. Any dependency relationship in spreadsheet involves a calculated cell, and either other calculated cells or data input cells. However, they do not exist between data input cells and other data input cells.

The *model schema* only holds absolute copies of the corresponding data in the input section. It is also protected as a precaution against any overwriting of data, and can only be manipulated by the programmer or model developer.

Based on the *leaves* identified in the Jackson structures, the *input* section can be created. The input section is constructed on a separate worksheet and should be labelled as such. The data input end-users must only be allowed to manipulate the input section for the entry and update of data. They are responsible for entering all the inputs to the spreadsheet model in this section. Based on *Figures 7.12, 7.13 (a), 7.13 (b)* and *7.14*, the inputs to the model are B, C, F, H, J and L.

A problem that can be anticipated at this stage is the difficulty in adding or deleting data from the *input* section while having the changes reflected in the *model schema*. In view of this problem, the methodology requires that a group of related inputs be defined as a range and only the range is referred to in the model schema. A reference to a group of related inputs or an input set (range) is shown in the Jackson structure by a *leaf* shown as a dashed-line box and represented as an iterated component.

Based on *Figure 7.13 (b)*, N represents a group of related inputs. Therefore, the elements of N are defined as a range in the input component. It can also be observed in *Figure 7.13 (b)* that M is a function of N. In the model schema, M references the range N. The elements of N are not physically present in the model schema. This way, any changes that take place within N will not affect the integrity of the formulae or calculations in the model schema. *Figure 7.16* shows the input component derived from the logical design of the spreadsheet model.

	B	Input _B
	C	Input _C
	F	Input _F
	H	Input _H
	J	Input _J
	L	Input _L
N		
	N ₁	Input _{N1}
	N ₂	Input _{N2}
	N _n	Input _{Nn}

Figure 7.16: Input Component

All constants (like *tax*, *interest* or *depreciation rates*) should be held in the input section and referenced or looked up by formulae in the structure and workings section of the model. True constants that will never change (e.g. *number of days of the week*) should be protected as a precaution against accidental overwriting. All constants can be placed in a separate sub-section within the input component. As an additional form of quality assurance, inputs should be subjected to programmed validation checks to reject any entry that falls outside a permissible range of acceptable values. The input component or data entry modules are accessible to the builder, maintainer, auditor with test authorisation, and user with data entry authorisation.

STAGE 6:
Implementation of Formulae and Binding Relationships in the Model Schema

The structured spreadsheet modules represent the model structure or calculation/workings section. The structured spreadsheet modules also facilitate auditing and comprehension of the composition/meaning of calculations (expressed as formulae).

The various formulae can now be physically constructed and all relationships implemented. This stage involves constructing the various formulae required in the model structure. The outline or layout of the model structure has already been produced and will be used as a basis for the creation of the appropriate formulae.

Each calculation would correspond to a unique label. This calculation must take the form of a formula which references other formulae and inputs (from the input component). This task can and must only be carried out by the model developer. Each and every formula is subsequently *protected* as a precaution against any accidental overwriting of formulae, especially by the data entry end-users.

The *model schema* (Figure 7.17) should only contain simple formulae – if necessary, complex calculations should be broken down into simple stages over a series of cells. This way, even complex calculations will be easy to understand – especially if annotated (Buler-97). Relatively complex formulae can be annotated, either with the spreadsheet program’s notes feature, to explain how key calculations work (Buler-97).

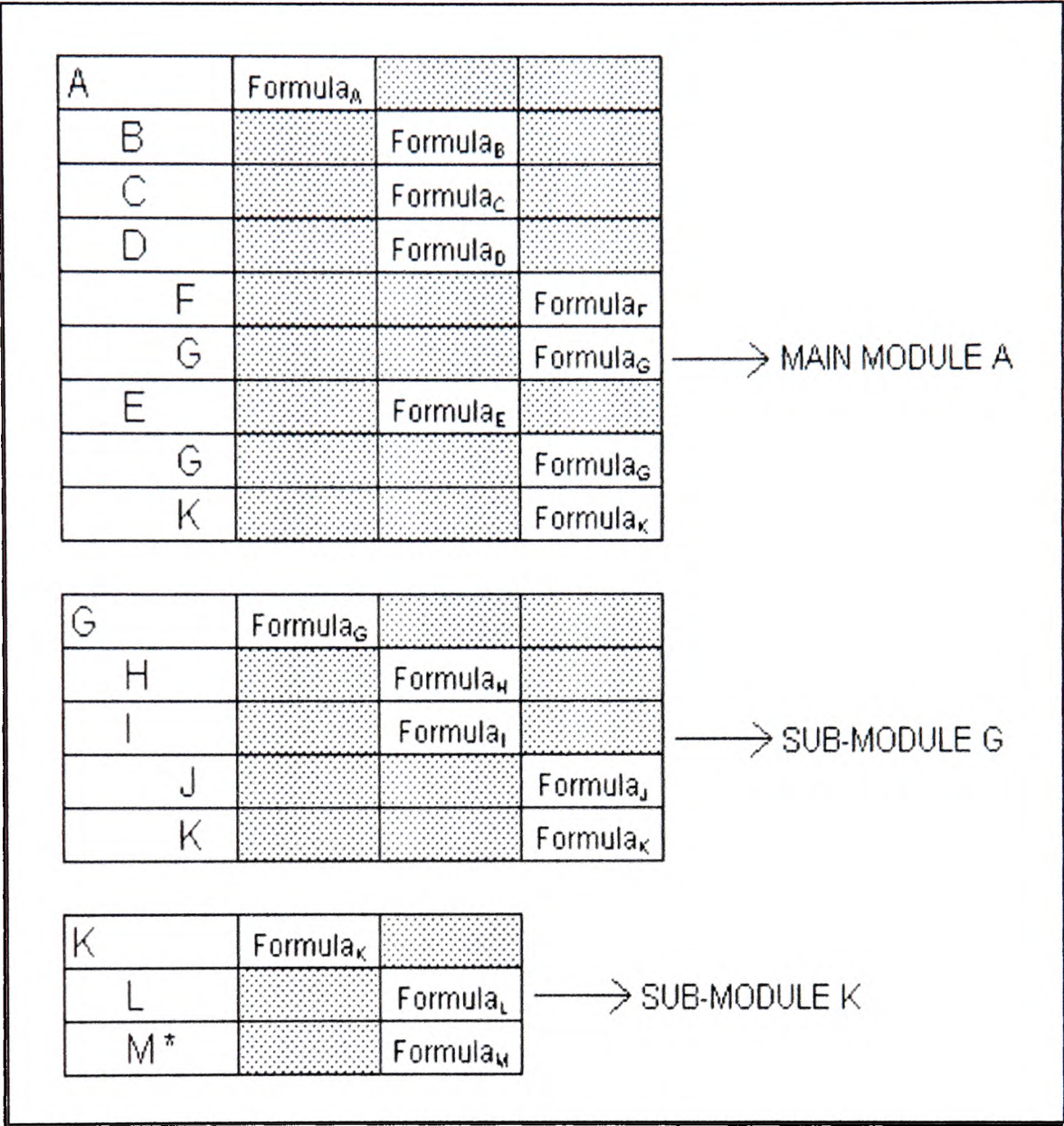


Figure 7.17: Model Schema

There are two distinct types of structures within the *model schema*. There are fixed and volatile parts of the model schema. The formulae and relationships between formulae are all fixed and can only be changed by the model developer. On the contrary, there can be iterations of structures or parts of the model which can be added or removed. In order to facilitate this, there has to be a mechanism to add or delete iterated structures or sub-structures. These can be identified from the logical design of the spreadsheet model schema. The addition/deletion of such iterated structures must be consistent with the addition/deletion of the corresponding inputs.

The *model schema* comprising structured spreadsheet modules facilitate auditing and comprehension of the composition/meaning of calculations (expressed as formulae). They are the interface accessible with read/write access to the model developer (builder and maintainer), and with read access for the user and auditor.

STAGE 7:

Implementation of References in the Output Component

References to corresponding *formulae* in the *model schema* and *data* in the *input component*, can at this stage be entered into the relevant cells of the *output component*.

STAGE 8:

Testing, Documentation and Administration of the Spreadsheet Model

There should be organisational standards in place for the testing, documentation, and maintenance or administration of spreadsheet models (McMickle-89, Simkin-87: cited in Isakowitz-95). This stage brings the spreadsheet model development process to a conclusion. It consists of three principal steps:

- **Step 1: Testing**
- **Step 2: Documentation**
- **Step 3: Administration**

As this stage is not considered a core aspect of the methodology, each of its constituent steps will be addressed only briefly and in passing. It is recommended that conventional software engineering approaches and principles be used for the testing, documentation and administration of the spreadsheet model.

Step 1 of this final stage requires that the entire spreadsheet model be rigorously tested before it goes into operation. The spreadsheet model is tested with a comprehensive set of test data. Ray Butler (Butler-97) proposes that the spreadsheet model should also be reviewed by someone other than the developer for errors before being brought into use.

In *Step 2*, documentation of the spreadsheet model is incorporated into the model itself, typically on a separate worksheet. Kee (Kee-88) states that documentation materials provide the instructions needed to apply a template properly, as well as the technical details needed to understand its underlying structure. Without adequate documentation, it is often easier to develop a new template than to review somebody else's program (Kee-88).

Step 3 addresses the administration of the spreadsheet model. After the spreadsheet model goes into operational use, proper administration of the spreadsheet model is essential. Mason and Keane (Mason-89) have proposed that a *model administrator* regulates and monitors spreadsheet modelling activities across the organisation.

7.4 Application of the Proposed Methodology

In order to illustrate the application of the methodology in practice, three different spreadsheet models are used as examples.

Example 1: Trading and Profit and Loss Account for a Particular Year

In this example, the methodology is applied in the construction of a spreadsheet model comprising a single module (as defined by the methodology). It is based on a *Trading and Profit and Loss Account for a particular year* (Ward-96). The original model is shown in *Figure 7.18*.

This is a simple model which does not require resolution of graph structures, which potentially result in the creation of separate modules, and recursive relationships. Most of the essential concepts and principles of the methodology are demonstrated, except the technique of *modularisation*. Module formation is shown in the second example, based on a *Post-tax Income Distribution Model*.

T Howe Ltd			
<i>Trading and Profit and Loss Account for the year ended 31 December 19X4</i>			
Sales			135,486
Less Cost of goods sold			
Opening stock	40,360		
Add Purchases	72,360		
Add Carriage inwards	1,570		
	<u>114,290</u>		
Less Closing stock	52,360	61,930	
Gross profit		<u>73,556</u>	
Less Expenses			
Salaries	18,310		
Rates and occupancy	4,515		
Carriage outwards	1,390		
Office expenses	3,212		
Sundry expenses	1,896		
Depreciation: Buildings	5,000		
Equipment	9,000		
Directors' remuneration	<u>9,500</u>	52,823	
Net profit		<u>20,733</u>	
Add Unappropriated profits from last year		15,286	
		<u>36,019</u>	
Less Appropriations			
Proposed dividend	10,000		
General reserve	1,000		
Foreign exchange	<u>800</u>	11,800	
Unappropriated profits carried to next year		<u>24,219</u>	

Figure 7.18: The Conventional Layout

The application of the proposed methodology in the analysis, design and implementation of this model is presented in detail in *Appendix C: Example 1*.

Example 2: *Post-tax Income Distribution Model*

In this example, the methodology is applied in the construction of a spreadsheet model composed of multiple modules. In this respect, it is deemed to be a more complicated model than the spreadsheet model used in the first example. It is based on a *Post-tax Income Distribution Model* (Slater-90). The original model is shown in *Figure 7.19 (a)* and an abridged version of the same model in *Figure 7.19 (b)*.

The technique of modularisation, a critical and integral part of the proposed methodology, is demonstrated through this example, in addition to the other features and characteristics of the methodology.

Table 3.5 Post-tax income distribution for 1975/6 and 1985/6							
Income after tax 1975/6		Number (thousands)	Total income	Number		Income	
				%	Cumulative %	%	Cumulative %
675 but under 750	750	357	255	1.63	1.63	0.45	0.45
750 but under 1000	1000	1350	1190	6.15	7.78	2.11	2.57
1000 but under 1250	1250	1780	2000	8.11	15.88	3.55	6.12
1250 but under 1500	1500	1840	2530	8.38	24.27	4.50	10.62
1500 but under 1750	1750	1850	3000	8.43	32.69	5.33	15.95
1750 but under 2000	2000	1750	3280	7.97	40.66	5.83	21.78
2000 but under 2500	2500	3270	7350	14.90	55.56	13.06	34.84
2500 but under 3000	3000	2830	7760	12.89	68.45	13.79	48.63
3000 but under 4000	4000	4150	14 300	18.90	87.35	25.41	74.04
4000 but under 5000	5000	1670	7360	7.61	94.96	13.08	87.12
5000 but under 6000	6000	575	3120	2.62	97.58	5.54	92.67
6000 but under 8000	8000	377	2550	1.72	99.30	4.53	97.20
8000 but under 10 000	10 000	97	852	0.44	99.74	1.51	98.71
10 000 and more		57	725	0.26	100.00	1.29	100.00
Income after tax 1985/6							
1750 but under 2000	2000	635	1190	2.89	2.89	0.83	0.83
2000 but under 2500	2500	1470	3290	6.68	9.57	2.30	3.14
2500 but under 3000	3000	1410	3850	6.41	15.97	2.70	5.83
3000 but under 3500	3500	1670	5420	7.59	23.56	3.79	9.63
3500 but under 4000	4000	1670	6250	7.59	31.15	4.38	14.00
4000 but under 4500	4500	1530	6510	6.95	38.10	4.56	18.56
4500 but under 5000	5000	1490	7070	6.77	44.87	4.95	23.51
5000 but under 5500	5500	1280	6700	5.82	50.69	4.69	28.20
5500 but under 6000	6000	1170	6710	5.32	56.00	4.70	32.90
6000 but under 7000	7000	2110	13 700	9.59	65.59	9.59	42.49
7000 but under 8000	8000	1760	13 100	8.00	73.59	9.17	51.67
8000 but under 10 000	10 000	2560	22 900	11.63	85.22	16.03	67.70
10 000 but under 12 000	12 000	1400	15 300	6.36	91.58	10.71	78.41
12 000 but under 15 000	15 000	956	12 700	4.34	95.93	8.89	87.31
15 000 but under 20 000	20 000	616	10 500	2.80	98.73	7.35	94.66
20 000 and more		280	7630	1.27	100.00	5.34	100.00

Source: *Annual Abstract of Statistics*. Reproduced with the permission of the Controller of Her Majesty's Stationery Office.

Figure 7.19 (a): The Original Model

As the aim here is to illustrate how the proposed methodology would be applied in the construction of the above model, its data content is reduced for simplicity. We are more concerned about the structure of the model rather than its data. The abridged version of the model is shown in *Figure 7.19 (b)*.

Post-tax income distribution for 1975 and 1985						
<i>Income after tax 1975</i>	<i>Number (thousands)</i>	<i>Total income</i>	<i>Number</i>		<i>Income</i>	
			<i>%</i>	<i>Cumulative %</i>	<i>%</i>	<i>Cumulative %</i>
675 but under	750	357	255			
750 but under	1000	1350	1190			
1000 but under	1250	1780	2000			
1250 but under	1500	1840	2530			
1500 but under	1750	1850	3000			
<i>Income after tax 1985</i>						
1750 but under	2000	635	1190			
2000 but under	2500	1470	3290			
2500 but under	3000	1410	3850			
3000 but under	3500	1670	5420			
3500 but under	4000	1670	6250			

Figure 7.19 (b): Abridged Version of the Original Model

In *Appendix C: Example 2*, the application of the proposed methodology in the analysis, design and implementation of this model is clearly demonstrated.

Example 3: Trading and Profit and Loss Account for Several Years

In this example, the methodology is used to build a *Trading and Profit and Loss Account for several years* instead. Unlike the previous example, this model contains recursive relationships. The model is based on the conventional layout shown in *Figure 7.20*.

	1996	1997	1998	1999
Sales		135,486	212,670	180,900
Cost of goods sold				
Opening stock		40,360	22,766	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
		112,720	104,290	248,854
Closing stock	40,360	22,766	44,700	25,430
Gross Profit		45,532	108,380	404,324
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
		52,823	28,600	31,140
Net profit		98,355	124,480	149,760
Unappropriated profits from last year		15,286	140,821	113,070
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240
		11,800	11,410	9,990
Unappropriated profits carried to next year	15,286	101,841	113,070	507,404

Figure 7.20: Trading and Profit and Loss Account for Several Years
(Conventional Layout)

The application of the proposed structured methodology in the analysis, design and construction of this model is distinctly demonstrated in *Appendix C: Example 3*.

7.5 Potential for Quality Improvement

This section discusses the proposed methodology's potential for enhancing the quality of spreadsheet models. There are various features and characteristics within the methodology that contribute to the quality improvement of the models.

The proposed methodology specifies a systematic and disciplined method for analysing, designing and building spreadsheet models, and a standard structure for the models. According to Kee (Kee-88), such an approach forces developers to build their applications within a logical framework. This simplifies spreadsheet construction and enhances reliability.

Without standards and a structured methodology in place, model developers would develop spreadsheets in a wide variety of styles and layouts. Depending on the nature of the models and the competence of the model developer, the models would vary in terms of their comprehensibility, reliability and maintainability. By strictly conforming to the proposed structured methodology, a group of model developers asked to independently construct a spreadsheet model, should, generate models with virtually identical structures. These models would also possess the various desirable attributes of spreadsheet models. This gives scope for peer review at the logical design stage. The fact that there is a standard for logical design (using *Jackson* structures) means that design errors can be spotted much earlier in the process. This is the essence of quality software production. Moreover, the structure diagrams (logical model) also provide certain achievable sub-goals for the development. This also facilitates peer group walkthroughs and review at an early stage in the design, and has a benefit for quality control of the spreadsheet models.

The methodology essentially involves structured analysis of data, based on *Jackson structures*. It is shown that this analysis allows a straightforward modularisation, and that individual modules may be represented with indentation in the *block-structured* form of structured programs. The benefits of this structured format are increased comprehensibility, ease of maintenance, and reduction in errors. The model can be interpreted in an unambiguous way. The methodology also has the capacity to provide a global sense of the structure of a spreadsheet model using *Jackson structures*.

According to Brown and Gould (Brown-87), formulae are represented in a location that is physically separate from the spreadsheet itself and that the user typically has a "window" onto only one formula at a time. They have stated that an improved interface might make formulae more visible and salient in the interface, and might represent formulae integrated with, rather than separate from, the spreadsheet itself. The proposed methodology caters for these requirements by organising the formula and its operands in a structured manner, and in close proximity. This makes the formulae highly visible in the interface. The inter-relationships between the various formulae could also be easily inferred.

Figure 7.21 (a) shows the spreadsheet model resulting from the application of the proposed methodology based on *Appendix C: Example 1*.

	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	24,219				
07	Net profit		20,733			
08	Gross Profit			73,556		
09	Sales				135,486	
10	Cost of goods sold				114,290	
11	Opening stock					40,360
12	Purchases					72,360
13	Carriage inwards					1,570
14	Closing stock				52,360	
15	Total expenses *			52,823		
16	Unappropriated profits from last year		15,286			
17	Total appropriations *		11,800			

Figure 7.27 (a): Model Schema

Figure 7.21 (b) presents a graphic representation of the dependencies between elements of the model schema. The logic of the model is easily comprehensible as it can be easily seen that each formula is a function of elements in the next virtual column.

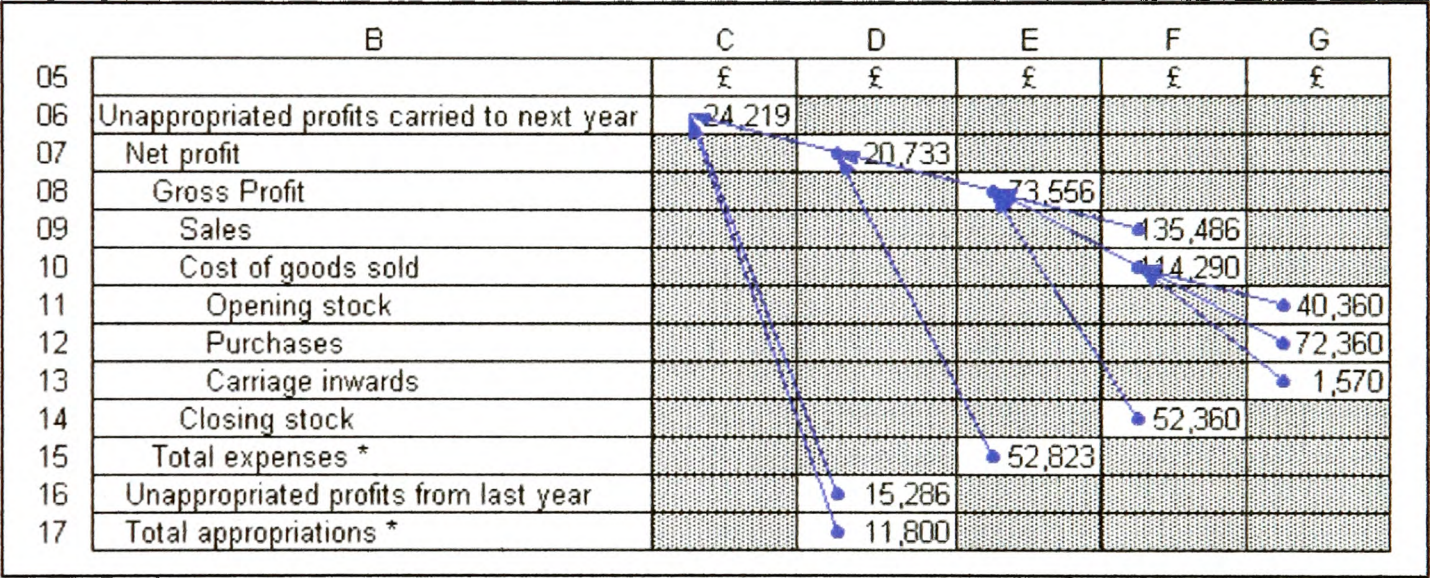


Figure 7.21 (b): Data Dependencies

Based on Figure 7.21 (b) it can be noticed that both the semantics and the data are clarified in this layout. For example, we can see straight away on the semantic level that *Unappropriated profits carried to next year* is derived from three figures: *Net Profit*, *Unappropriated profits from last year* and *Total appropriations*.

On the data level we see that 24,219 is made up from 20,733, 15,286 and 11,800. Likewise, we see immediately (from the asterisk *) that *Total expenses* references an input range from the input component. Notice also that columns in the spreadsheet show figures on the same semantic level, enabling valid comparisons between figures to be made. For example, column D shows *net profit*, *unappropriated profits from last year* and *total appropriations*. These figures give a valid impression of the state of the *Trading and Profit and Loss Account* at this level of detail. If we were to include a figure from a different level, e.g. *purchases* (from column G), it would confuse the picture, since it has already been included in *net profit*.

Referring to *Figures 7.21 (a) or 7.21 (b)*, it is beyond any doubt that the use of *indentation* and *virtual columns* make it far more straight-forward to make sense of and comprehend the composition of formulae. In order to further enhance the comprehensibility and integrity of the spreadsheet model, each data value and formula in the input component and model schema is assigned a unique name. These names are then used as operands within formulae, instead of cell addresses. If this technique is applied, the formula view of the model schema will appear as displayed in *Figure 7.21 (c)*. It can be seen that in *Figure 7.21 (c)*, the formulae or references are more comprehensible as they are in natural language.

	B	C	D	E	F	G
		£	£	£	£	£
05						
06	Unappropriated profits carried to next year	=(NetProfit+UnappropriatedProfitsFromLastYear)-TotalAppropriations				
07	Net profit		=GrossProfit-TotalExpenses			
08	Gross Profit			=Sales-CostOfGoodsSold+ClosingStock		
09	Sales				=Sales	
10	Cost of goods sold				=SUM(OpeningStock;CarriageInwards)	
11	Opening stock					=OpeningStock
12	Purchases					=Purchases
13	Carriage inwards					=CarriageInwards
14	Closing stock				=ClosingStock	
15	Total expenses *			=SUM(Expenses)		
16	Unappropriated profits from last year		=UnappropriatedProfitsFromLastYear			
17	Total appropriations *		=SUM(Appropriations)			

Figure 7.21 (c): Model Schema

As mentioned in *Stage 5* of the methodology, a common error committed by users is the accidental overwriting of formulae. The technique proposed to overcome this problem is to put all inputs in a separate worksheet. This is called the *input section*. It is also protected as a precaution against any overwriting of data. There are reasons why cells for data input and assumptions should be grouped together in an input section, separate from the structured modules in the model schema. One reason is to do with the utmost importance of obtaining accurate data entry. Kee (Kee-88) supports this by stating that using a central data entry area makes data entry easier and helps to prevent input errors. A second reason is that input cells are often referred to by more than one calculated cell.

7.6 Summary

This chapter has described and presented the proposed structured methodology for the development and integrity control of spreadsheet models. This was preceded by a discussion of the development and synthesis of the methodology from the material considered in *Chapter 6*. The methodology's potential for enhancing the quality of spreadsheet models has also been explained.

It has been found that most of the software engineering principles discussed in *Chapter 6* are applicable to the development of spreadsheet models, and are therefore, used in the synthesis of the proposed methodology. Therefore, the methodology is based on a systematic approach that encompasses all the stages and activities of

spreadsheet model development. Various methods, tools and techniques are incorporated within the methodology, along with models, notations, rules and design advice.

In the proposed methodology, a diagrammatic representation of the logical design of the spreadsheet model is produced using *Jackson structures*. Other important software engineering principles and techniques have also been applied in the various stages of the methodology. They include *indentation and translation of data structure into structured form, virtual columns, separation of inputs, model schema and outputs, and modularisation*.

The proposed structured methodology consists of eight principal stages:

- Requirements Analysis and Development of *Output Structures*
- Conceptual Design of the *Model Schema*
- Logical Design of the *Model Schema*
- Physical Construction of the *Model Schema Layout* on the Spreadsheet
- Development of the *Input Component* and Entry of *Model Inputs*
- Implementation of *Formulae* and *Binding Relationships* in the *Model Schema*
- Implementation of *References* in the *Output Component*
- *Testing, Documentation and Administration* of the Spreadsheet Model

The application of the methodology is demonstrated using three different examples of spreadsheet models. They are a *Trading and Profit and Loss Account for a Particular Year*, a *Post-tax Income Distribution Model* and a *Trading and Profit and Loss Account for Several Years*.

The proposed methodology has various benefits in terms of quality improvement. It is based on a disciplined and standard approach to spreadsheet model development within a logical framework. The creation of standard model structures facilitates peer review, enabling the early detection of errors. The logic of the model can be easily understood from a clear representation of the dependencies between model elements. The structured format for spreadsheet models produced in the methodology can increase the comprehensibility, maintainability and accuracy of the models.

CHAPTER 8

EVALUATION STRATEGY AND EXPERIMENTS

8.1 Introduction

In order to evaluate the effectiveness of the proposed methodology presented in *Chapter 7*, a series of experiments had to be conducted. An analysis of the results of these experiments is used to assess the methodology's potential for integrity control of spreadsheet models.

This chapter begins by putting forth a plan for the evaluation of the proposed methodology based on experimental trials. The underlying evaluation strategy of the experiments is also discussed. The actual experiments undertaken are subsequently described in detail. At the end of the chapter, a tabulated summary of the experiments, their subjects, and their different aims is presented.

The experiments are aimed at testing the various features of the proposed structured methodology. The series of experiments involve a range of spreadsheet models used in educational institutions and industry. The elements of the methodology are tested on diverse groups of students. Two different strategies are formulated to evaluate the quality of the proposed methodology for spreadsheet model development. Various aspects of the experiments are meticulously studied in planning the experiments.

8.1.1 User Groups or Participants

Ideally, the methodology should be tested on spreadsheet users, of varying levels of spreadsheet literacy, in both business and academia. It has been virtually impossible to obtain consent to conduct trials in business organisations due to various reasons. In some cases, there were rules and policies in place against such experiments, conducted by external individuals or organisations. In others, staff were unwilling to participate in the trials due to the assumption that these experiments would consume considerable time and effort.

Referring to past experiments undertaken, as shown in *Figure 8.1* (Panko-96,98), it has been found that most of the participants of such tests were students at an institution associated with the author(s). In most cases where the subjects were industry or commercial users, the experiment was either conducted by the particular organisation or the information derived from the normal operations of the organisation and published by the company itself.

Study	Sample
<i>Field Audits</i>	
Davies & Ikin (Davies-87)	19 operational spreadsheets
Butler (Butler-92)	273 operational spreadsheets audited by 143 United Kingdom tax inspectors in 1992.
Cragg & King (Cragg-93)	20 operational spreadsheets from 10 firms.
Hicks (Hicks-95)	1 module with 19 submodules about to enter operation.
Coopers & Lybrand (Coopers-97)	23 spreadsheets from industry
KPMG (KPMG-97)	22 spreadsheets from industry
Butler (Butler-00)	7 spreadsheets for tax submissions
<i>Cell Entry Experiments</i>	
Olson & Nilsen (Olsen-87-88)	14 experienced Lotus 1-2-3 users.
Lerch (Lerch-88)	21 professionals with at least one year of Lotus 1-2-3 experience.
<i>Development Experiments</i>	
Brown & Gould (Brown-87)	9 highly experienced spreadsheet developers
Hassinen (Hassinen-88)	92 novice spreadsheet students developed 355 spreadsheets
Hassinen (Hassinen-88)	10 novice students developing 48 spreadsheets
Janvrin & Morrison (Janvrin-96,00)	61 upper division business and graduate accounting students
Janvrin & Morrison (Janvrin-96,00)	88 senior-level accounting students
Panko & Halverson (Panko-97)	35 undergraduate business students working alone
Panko & Halverson (Panko-97)	40 undergraduate business students working in groups of 4
Panko & Sprague (Panko-99)	102 undergraduate business students and 50 MBA students. 17 MBAs had more than 250 hours of experience
Teo & Tan (Teo-97)	168 undergraduate business students taking second-year IS course
(Unpublished)	80 undergraduate business students
<i>Code Inspection Experiments</i>	
Galletta et al (Galletta-93)	30 MBA students and 30 CPA accountants
Galletta et al (Galletta-97)	113 MBA students
Panko & Sprague (Panko-99)	23 undergraduate subjects with errors in initial model.
Panko (Panko-99a)	33 undergraduate MIS majors

Figure 8.1: Studies on Spreadsheet Errors

Based on these findings, it is clear that carrying out tests within academic institutions is the most feasible option. Tests are therefore, planned to be carried out at a London-based University, involving different groups of students. As learnt from previous experiments, an advantage of using students is that we are normally aware of their level of computer/spreadsheet literacy and the experiments can be better controlled.

Three different groups of students have been selected as participants for the proposed experiments. They are as follows:

- Undergraduates
- Post-graduate students
- Students on a short course designed primarily for professionals in industry

8.1.2 Types of Errors

Ideally, the tests should demonstrate the capacity of the proposed structured methodology to address all types of spreadsheet errors. The taxonomy or classification of spreadsheet errors presented in *Chapter 3* is used as a basis for organising tests for as many different types of errors as possible.

8.1.3 Spreadsheet Models

The spreadsheet models selected and used for experimental purposes should be common business and financial models. The models should address the different features of the proposed methodology. Moreover, the models should have the capacity to be used to test for as many different types of errors as possible.

The spreadsheet models selected for the experiments are as follows:

- *A Trading and Profit and Loss Account* for a particular year (Wood-96)
- *A Trading and Profit and Loss Account* for several years
- *A Post-tax Income Distribution Model* (Slater-90)
- *A Balance Sheet* (Read-99)

8.2 The Evaluation Strategies

Two different strategies have been developed to evaluate the quality of the proposed methodology for spreadsheet model development.

8.2.1 Error Prevention

The first strategy for testing the quality of the proposed methodology is based on error prevention. It involves comparing the occurrence of errors in spreadsheet models developed based on the proposed methodology to the occurrence of errors in models built using conventional unstructured methods. The aim of this strategy is to establish whether or not there is a material difference in error rates between spreadsheet models produced using the two different approaches. The hypothesis is that users commit significantly fewer errors by adopting the proposed structured methodology. The first experiment is based on this strategy while the subsequent three experiments are based on a different strategy (error detection).

8.2.2 Error Detection

The second strategy for evaluating the effectiveness of the proposed methodology is based on error detection. It involves comparing the probability of detecting errors in spreadsheet models developed based on the proposed methodology to the probability of detecting errors in models constructed based on conventional unstructured methods.

Errors are deliberately seeded into the spreadsheet models. The aim of this strategy is to establish whether or not there is a significant difference in the probabilities of error detection between spreadsheet models produced using the two different approaches. The hypothesis is that users are able to identify significantly more errors seeded into a model developed using the proposed structured methodology. This is a reflection of its comprehensibility. It is particularly important for audit, review and update purposes. Apart from the first experiment, the subsequent three experiments are based on this strategy.

8.3 The Experiments Undertaken

8.3.1 Experiment 1

This experiment was carried out in two different stages, each involving two groups of students at a University. The purpose of the experiment was to compare two different approaches to the development of a single-module spreadsheet model. The first approach was based on conventional unstructured methods for spreadsheet model development while the second approach was based on the proposed structured methodology. This experiment was based on the first testing strategy, described earlier. The spreadsheet model used is based on a *Trading and Profit and Loss Account* for a particular year (Wood-96).

Stage 1

The *first stage* of the experiment involved the development of a spreadsheet model without any guidance or support. The participants had to employ suitable methods based on personal experience or discretion, and carry out the exercise independently under time constraint.

Subjects were given the desired output of the model as shown in *Figure 8.2*, on a separate worksheet labelled *Output*. In order to create the spreadsheet model based on the required output, they were provided with all the formulae needed in a worksheet labelled *Formulae*, as displayed in *Appendix D: Figure D1 (a)*.

	B	C
03	Net profit	?
04	Unappropriated profits from last year	?
05	Appropriations	?
06	Unappropriated profits carried to next year	?
07		
08		
09		
10		
11	Gross Profit	?
12	Expenses	?
13	Net profit	?

Figure 8.2: Desired Output

They were required to systematically organise and perform the appropriate calculations in a worksheet called *SCHEMA*, which was blank. This was their principal task. The calculations were based on the elements in the desired *output*. The *schema* sheet should contain all required data labels (e.g. *Net profit*, *Purchases*, *Salaries*, *Appropriations*, etc.) as well as their associated numeric values, either as an *input* or a *formula*. They were asked to try and present all the calculations within the same structure, so that the relationships between them would be clearer.

In order to carry out some of the calculations, they would require certain inputs. All the inputs required were provided in a worksheet labelled *Input*. This is shown in *Appendix D: Figure D1 (b)*. Participants were allowed to reorganise or restructure the inputs.

Finally, participants of the experiment had to use the results of the calculations to replace the unknown values in the output, denoted by a question mark (?). They were told not to alter the structure of the output as this was assumed to be the output style required by the end-user. The relevant cells were therefore protected against accidental overwriting or alteration.

A total of **42 post-graduate students** and **26 short course students** (most of whom were working in industry) took part in this experiment. Two tests were carried out in *Stage 1*.

Test 1

The *first test* was carried out on a group of **22 post-graduate students**. The students were pursuing a taught masters programme. Most of them had graduated in other disciplines and had limited prior knowledge of information systems.

This test was split into two sessions. Both sessions involved the same set of participants carrying out the same task(s). Therefore, each participant had to build the same spreadsheet model twice, once in each session. The purpose of having the participants rebuild the same model was so that it can be used as a *control* in the experiment.

Test 2

The *second test* was performed on a group of **12 short course students**. Most of the students were employed on a full-time basis in industry. Each participant had to build the spreadsheet model without having had a lesson on the proposed methodology. Due to time constraint, participants were not asked to rebuild the same model for control purposes. However, in *Stage 2*, the results of this set of participants are compared against the results of another group of short course participants with a similar background.

Stage 2

The *second stage* of the experiment involved the development of the same spreadsheet model based on the *Trading and Profit and Loss Account*. However, before participants engaged in the experiment, they were given a tutorial/lesson on

employing the proposed structured methodology for building and structuring a single-module spreadsheet model. During the tutorial, no references were made to elements of the spreadsheet model used in the experiment. Instead, participants were taught the generic algorithm and steps involved. Where deemed necessary, other examples were used. This stage too was composed of two tests.

Test 1

The *first test* was carried out on a different group of **20 post-graduate students**. The students were pursuing the same taught masters programme. As carried out in *Stage 1*, this test was also divided into two sessions. Both sessions involved the same group of participants but they had to carry out a different set of tasks.

In the *first session*, each participant had to first build the spreadsheet model using a method they were familiar with. This was not based on any structured methodologies and was exactly the same as the experiment in *Stage 1*. None of these participants had however taken part in the previous tests. The purpose of this exercise was to ensure that the errors committed by this group of students were in fact consistent with those produced by the previous group (in *Stage 1*).

In the *second session*, the same group was first given a lesson/tutorial on using the proposed structured methodology to construct a single-module spreadsheet model. They subsequently had to re-construct the spreadsheet model based on the proposed methodology. Ideally, complying with the algorithm, steps and rules of the methodology, they were expected to produce a schema as shown in *Appendix D: Figure D1 (c)*.

Test 2

Test 2 was conducted on a group of **14 short course students**. This was a different group of students but who were pursuing a different offering of the same short course. Moreover, they had a similar background, in that they were also mainly holding professional positions in industry. The participants were asked to create the spreadsheet model, having had a lesson on building spreadsheet models using the proposed methodology. This was similar to *Session 2* of the previous test (*Test 1*).

8.3.2 Experiment 2

This experiment was based on the second evaluation strategy (error detection) and carried out in two stages. A total of **104 undergraduates** took part in this experiment. The students were in two different groups. The first group of 55 students took part in the *first stage* of the experiment while the second group of 49 students participated in the *second stage* of the experiment.

Both groups had to detect and correct a total of 12 errors that had been seeded into a spreadsheet model. They were given the same amount of time to complete the exercise. The model was based on a *Trading and Profit and Loss Account for several years*. However, there was a fundamental difference between the layout or structure of

the model used by the first group (in *Stage 1*) and the model used by the second group (in *Stage 2*).

Stage 1

In this stage, the participating group consisted of **55 students** and were presented the spreadsheet model based on a conventional layout as presented in *Figure 8.3*. Their task was to identify the twelve errors that had been seeded into the model. As most of the students did not possess adequate knowledge of accounting, all formulae related to the model were provided. This is shown in *Appendix D: Figure D2 (a)*. They were not aware of how many errors the model contained. *Appendix D: Figure D2 (b)* shows the model with the errors highlighted while *Appendix D: Figure D2 (c)* displays the formula view of the model. In *Appendix D: Figure D2 (d)*, the correct version of the model is given.

	1996	1997	1998	1999
Sales		135,486	212,670	180,900
Cost of goods sold				
Opening stock		40,360	22,766	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
		112,720	104,290	248,854
Closing stock	40,360	22,766	44,700	25,430
Gross Profit		45,532	108,380	404,324
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
		52,823	28,600	31,140
Net profit		98,355	124,480	149,760
Unappropriated profits from last year		15,286	140,821	113,070
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240
		11,800	11,410	9,990
Unappropriated profits carried to next year	15,286	101,841	113,070	507,404

Figure 8.3: Model with Hidden Seeded Errors (Conventional Layout)

Stage 2

In *Stage 2*, the group of participants was made up of **49 students**. The experiment involved the same model but it had been built and structured based on the proposed methodology. The same errors had been seeded into this model as well, and participants had to independently detect and correct them. They were given a brief and general lesson/tutorial on how to interpret a spreadsheet model based on the proposed methodology without any references to the particular model used.

The model given to the students (with the seeded errors) is shown in *Figure 8.4*. The input component of the model is displayed in *Appendix D: Figure D3 (a)*. *Appendix D: Figure D3 (b)* shows the same model with the errors highlighted while *Appendix D: Figure D3 (c)* displays the formula view of the model. In *Appendix D: Figure D3 (d)*, the correct version of the model is presented.

	£	£	£	£	£	£
Year						
1997						
Unappropriated profits carried to next year	101,841					
Net profit		98,355				
Gross Profit			45,532			
Sales				135,486		
Cost of goods sold				112,720		
Opening stock					40,360	
Closing stock from last year						40,360
Purchases					72,360	
Carriage inwards					1,570	
Closing stock				22,766		
Total expenses *			52,823			
Unappropriated profits from last year		15,286				
Total appropriations *		11,800				
1998						
Unappropriated profits carried to next year	113,070					
Net profit		124,480				
Gross Profit			108,380			
Sales				212,670		
Cost of goods sold				104,290		
Opening stock					22,766	
Closing stock from last year						22,766
Purchases					52,340	
Carriage inwards					890	
Closing stock				44,700		
Total expenses *			28,600			
Unappropriated profits from last year		140,821				
Total appropriations *		11,410				
1999						
Unappropriated profits carried to next year	507,404					
Net profit		149,760				
Gross Profit			404,324			
Sales				180,900		
Cost of goods sold				248,854		
Opening stock					44,700	
Closing stock from last year						44,700
Purchases					22,109	
Carriage inwards					1,145	
Closing stock				25,430		
Total expenses *			31,140			
Unappropriated profits from last year		113,070				
Total appropriations *		9,990				

Figure 8.4: Model with Hidden Seeded Errors (Structured Layout)

8.3.3 Experiment 3

This experiment was based on the second evaluation strategy (error detection) and carried out in two stages. A total of **41 post-graduate students** (pursuing the same course) and **23 short course students** (also on the same short course) participated in this experiment. Two identical tests were performed in each stage. Each test involved a different subset of students. Therefore 4 different groups of subjects/participants had to detect and correct a total of 10 errors that had been seeded into a spreadsheet model. The difference between the two models was their structure or layout. All participants were given the same amount of time to complete the exercise. The model used in this experiment was based on a *Post-tax Income Distribution Model* (Slater-90). The original model was modified slightly to decrease its size.

Stage 1

In the first stage of the experiment, the spreadsheet model was presented based on the original (conventional) layout (Slater-90). The model is shown in *Figure 8.5*. They had to identify a total of 10 errors that had been seeded into the model. The first test involved a group of **19 post-graduate students** while the second test was conducted on a group of **11 short course students**.

	B	C	D	E	F	G	H
02	Post-tax income distribution for 1975 and 1985						
03							
04	Income after tax	Number		Number		Income	
05	1975	(thousands)	Total income	%	Cumulative %	%	Cumulative %
06							
07	675 but under 750	357	255	4.97	23.78	2.84	2.84
08	750 but under 1000	1350	1190	18.81	42.59	13.62	16.46
09	1000 but under 1250	1780	2000	24.80	67.40	22.28	38.74
10	1250 but under 1500	1840	2530	25.64	93.03	28.19	10.55
11	1500 but under 1750	1850	3000	25.78	118.81	33.43	43.98
12							
13	Income after tax						
14	1985						
15							
16	1750 but under 2000	635	1190	9.26	9.26	17.36	17.36
17	2000 but under 2500	1470	3290	47.99	57.26	16.45	33.81
18	2500 but under 3000	1410	3850	20.57	105.25	19.25	36.61
19	3000 but under 3500	1670	5420	24.36	129.61	27.10	63.71
20	3500 but under 4000	1670	6250	26.72	156.33	31.25	94.96
21							

Figure 8.5: Model with Seeded Errors (Conventional Layout)

It was assumed that participants of the experiment did not have adequate knowledge of the mathematics required for calculations in the model. Therefore, all formulae needed to comprehend the various calculations were provided. This is shown in *Appendix D: Figure D4 (a)*. Subjects in both tests were unaware of the number of errors that had been seeded into the model. *Appendix D: Figure D4 (b)* contains the model with the errors highlighted while *Appendix D: Figure D4 (c)* displays the formula view of the same model. In *Appendix D: Figure D4 (d)*, the correct version of the model is given.

Stage 2

In the second stage of the experiment, the spreadsheet model was re-designed and re-structured according to the proposed methodology. The same 10 errors were then deliberately seeded into the model. The participants of the experiment at this stage were given a brief and general tutorial/lesson on how to interpret a spreadsheet model based on the proposed methodology without any specific references to the particular model used.

The first test was performed on a different group of **22 post-graduate students** while the second test involved a group of **12 short course students** (on a different offering of the same short course pursued by subjects of *Test 2* in *Stage 1*).

The model given to the students (with the seeded errors) is shown in *Figure 8.6*. The model has been created based on the proposed methodology. However, cell addresses are used in formulae/references instead of meaningful labels, as recommended by the methodology. The input component of the model is displayed in *Appendix D: Figure D5 (a)*. *Appendix D: Figure D5 (b)* shows the spreadsheet model with the errors highlighted while *Appendix D: Figure D5 (c)* contains the formula view of the model. In *Appendix D: Figure D5 (d)*, the correct version of the model is presented.

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008		Number Cumulative %	23.78		
009		<i>Previous Number Cumulative %</i>		0.00	
010		Number %		4.97	
011		Number (000)			357
012		<i>Total Number (000)</i>			7177
013		Income Cumulative %	2.84		
014		<i>Previous Income Cumulative %</i>		0.00	
015		Income %		2.84	
016		Total Range Income			255
017		<i>Total Year Income</i>			8975
018	750 but under 1000				
019		Number Cumulative %	42.59		
020		<i>Previous Number Cumulative %</i>		23.78	
021		Number %		18.81	
022		Number (000)			1350
023		<i>Total Number (000)</i>			7177
024		Income Cumulative %	16.46		
025		<i>Previous Income Cumulative %</i>		2.84	
026		Income %		13.62	
027		Total Range Income			1190
028		<i>Total Year Income</i>			8975
029	1000 but under 1250				
030		Number Cumulative %	67.40		
031		<i>Previous Number Cumulative %</i>		42.59	
032		Number %		24.80	
033		Number (000)			1780
034		<i>Total Number (000)</i>			7177
035		Income Cumulative %	38.74		
036		<i>Previous Income Cumulative %</i>		16.46	
037		Income %		22.28	
038		Total Range Income			2000
039		<i>Total Year Income</i>			8975
040	1250 but under 1500				
041		Number Cumulative %	93.03		
042		<i>Previous Number Cumulative %</i>		67.40	
043		Number %		25.64	
044		Number (000)			1840
045		<i>Total Number (000)</i>			7177
046		Income Cumulative %	10.55		
047		<i>Previous Income Cumulative %</i>		38.74	
048		Income %		28.19	
049		Total Range Income			2530
050		<i>Total Year Income</i>			8975
051	1500 but under 1750				
052		Number Cumulative %	118.81		
053		<i>Previous Number Cumulative %</i>		93.03	
054		Number %		25.78	
055		Number (000)			1850
056		<i>Total Number (000)</i>			7177
057		Income Cumulative %	43.98		
058		<i>Previous Income Cumulative %</i>		10.55	
059		Income %		33.43	
060		Total Range Income			3000
061		<i>Total Year Income</i>			8975
062	Total Number (000)				
063		Total Number (000) *	7177		
064	Total Year Income				
065		Total Year Income *	8975		

continued ...

	B	C	D	E	F
066	1985				
067	Range				
068	1750 but under 2000				
069		Number Cumulative %	9.26		
070		<i>Previous Number Cumulative %</i>		0.00	
071		Number %		9.26	
072		Number (000)			635
073		<i>Total Number (000)</i>			6855
074		Income Cumulative %	17.36		
075		<i>Previous Income Cumulative %</i>		0.00	
076		Income %		17.36	
077		Total Range Income			1190
078		<i>Total Year Income</i>			20000
079	2000 but under 2500				
080		Number Cumulative %	57.26		
081		<i>Previous Number Cumulative %</i>		9.26	
082		Number %		47.99	
083		Number (000)			1470
084		<i>Total Number (000)</i>			6855
085		Income Cumulative %	33.81		
086		<i>Previous Income Cumulative %</i>		17.36	
087		Income %		16.45	
088		Total Range Income			3290
089		<i>Total Year Income</i>			20000
090	2500 but under 3000				
091		Number Cumulative %	105.25		
092		<i>Previous Number Cumulative %</i>		57.26	
093		Number %		20.57	
094		Number (000)			1410
095		<i>Total Number (000)</i>			6855
096		Income Cumulative %	36.61		
097		<i>Previous Income Cumulative %</i>		33.81	
098		Income %		19.25	
099		Total Range Income			3850
100		<i>Total Year Income</i>			20000
101	3000 but under 3500				
102		Number Cumulative %	129.61		
103		<i>Previous Number Cumulative %</i>		105.25	
104		Number %		24.36	
105		Number (000)			1670
106		<i>Total Number (000)</i>			6855
107		Income Cumulative %	63.71		
108		<i>Previous Income Cumulative %</i>		36.61	
109		Income %		27.10	
110		Total Range Income			5420
111		<i>Total Year Income</i>			20000
112	3500 but under 4000				
113		Number Cumulative %	156.33		
114		<i>Previous Number Cumulative %</i>		129.61	
115		Number %		26.72	
116		Number (000)			1670
117		<i>Total Number (000)</i>			6855
118		Income Cumulative %	94.96		
119		<i>Previous Income Cumulative %</i>		63.71	
120		Income %		31.25	
121		Total Range Income			6250
122		<i>Total Year Income</i>			20000
123	Total Number (000)				
124	Total Number (000) *		6855		
125	Total Year Income				
126	Total Year Income *		20000		

Figure 8.6: Model with Seeded Errors (Structured Layout)

8.3.4 Experiment 4

This experiment was very similar to the previous experiment (*Experiment 3*). The only difference was that a different spreadsheet model was used. However, this was also a common business model, a *balance sheet* (Read-99). The model was abridged before creating it on a spreadsheet, to make it less time-consuming to work on. The original model is displayed in *Appendix D: Figure D6 (a)* while the abridged version of the model (spreadsheet view) is shown in *Appendix D: Figure D6 (b)*. This is, therefore, the correct, error-free version of the model.

The experiment was carried out in two stages and involved a total of **44 post-graduate students** (pursuing the same course) and **23 short course students** (also on the same short course). Two identical tests were performed in each stage. Each test involved a different subset of students. The task of the 4 different groups of participants was to detect a total of 10 errors that had been seeded into the spreadsheet model. All participants were given the same amount of time to complete the exercise.

Stage 1

In the first stage of the experiment, the spreadsheet model was presented to participants based on the conventional layout. 10 errors had been deliberately seeded into the model. This erroneous model is shown in *Figure 8.7*. For the benefit of students not familiar with the interpretation of balance sheets, a set of relevant formulae was provided. This is shown in *Appendix D: Figure D6 (c)*. In *Appendix D: Figure D6 (d)*, the errors are highlighted and in *Appendix D: Figure D6 (e)*, the formula view of the erroneous model is displayed, with the flaws highlighted.

Two identical tests were carried out on different sets of participants. The first test involved a group of **24 post-graduate students** while the second test was conducted on a group of **12 short course students**.

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s	1999/2000	2000/01	2001/02	2002/03
04					
05					
06	Fixed assets	56,742	55,622	54,481	53,318
07					
08	Current assets				
09	Cash	3,251	4,372	2,375	1,218
10	Debtors	3,088	3,328	3,602	3,762
11					
12	Total assets	59,993	63,322	5,977	58,298
13					
14	Current liabilities				
15	Debt	51,411	51,411	51,411	51,411
16	Creditors	3,122	2,619	2,541	2,648
17	Capital creditors	1,170	136	140	144
18	Tax creditor	132	352	1,358	1,729
19					
20	Total liabilities	55,835	54,166	52,769	55,932
21					
22	Net assets	4,581	9,156	58,746	1,907
23					
24	Share capital	1,000	1,000	1,000	1,000
25	P&L reserve	6,246	7,804	4,008	1,366
26					
27	Shareholders' funds	7,246	8,804	5,008	1,366

Figure 8.7: Erroneous Model

Stage 2

In the second stage of the experiment, the spreadsheet model was re-designed and re-structured according to the proposed methodology. The same 10 errors were then seeded into the model. However, cell addresses are used in formulae/references instead of meaningful labels, as recommended by the methodology. This model (with the seeded errors) is shown in *Figure 8.8*. The input component of the model is displayed in *Appendix D: Figure D7 (a)*.

As done in the previous experiment, the students taking part in the experiment at this stage were given a brief and general tutorial on how to interpret a spreadsheet model based on the proposed methodology without any direct or specific references to the *balance sheet* used in the experiment. The first test was performed on a different group of **20 post-graduate students** while the second test involved a group of **11 short course students** (on a different offering of the same short course pursued by subjects of *Test 2* in *Stage 1*).

Appendix D: Figure D7 (b) shows the spreadsheet model with the errors highlighted while *Appendix D: Figure D7 (c)* contains the formula view of the model. In *Appendix D: Figure D7 (d)*, the correct version of the model is displayed.

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s				
04					
05	Year				
06	1999/2000				
07		Net assets	4,581		
08		Total assets		59,993	
09		Fixed assets			56,742
10		Cash			3,251
11		Debtors			3,088
12		Total liabilities		55,835	
13		Debt			51,411
14		Creditors			3,122
15		Capital creditors			1,170
16		Tax creditor			132
17		Shareholders' funds	7,246		
18		Share capital		1,000	
19		P&L reserve		6,246	
20	2000/01				
21		Net assets	9,156		
22		Total assets		63,322	
23		Fixed assets			55,622
24		Cash			4,372
25		Debtors			3,328
26		Total liabilities		54,166	
27		Debt			51,411
28		Creditors			2,619
29		Capital creditors			136
30		Tax creditor			352
31		Shareholders' funds	8,804		
32		Share capital		1,000	
33		P&L reserve		7,804	
34	2001/02				
35		Net assets	58,746		
36		Total assets		5,977	
37		Fixed assets			54,481
38		Cash			2,375
39		Debtors			3,602
40		Total liabilities		52,769	
41		Debt			51,411
42		Creditors			2,541
43		Capital creditors			140
44		Tax creditor			1,358
45		Shareholders' funds	5,008		
46		Share capital		1,000	
47		P&L reserve		4,008	
48	2002/03				
49		Net assets	1,907		
50		Total assets		58,298	
51		Fixed assets			53,318
52		Cash			1,218
53		Debtors			3,762
54		Total liabilities		55,932	
55		Debt			51,411
56		Creditors			2,648
57		Capital creditors			144
58		Tax creditor			1,729
59		Shareholders' funds	1,366		
60		Share capital		1,000	
61		P&L reserve		1,366	

Figure 8.8: Structured Model with Seeded Errors

8.4 Summary

Figure 8.9 provides a tabulated summary of the experiments, their subjects, and their different aims. These are also cross-referred to the research questions specified in Chapter 1. Experiment 1 is different from the other three experiments. Therefore, it has a different aim and tries to address the research questions differently. On the other hand, as Experiments 2, 3 and 4 are very similar in nature, they have the same aim and attempt to address the research questions in the same way.

Experiment	Subjects	Aim	Research Questions Addressed
Experiment 1			
Stage 1			
Test 1			
Session 1	22 post-graduate students		
Session 2	22 post-graduate students		
Test 2	12 short course students		
Stage 2			
Test 1			
Session 1	20 post-graduate students		
Session 2	20 post-graduate students		
Test 2	14 short course students		

continued ...

Experiment	Subjects	Aim	Research Questions Addressed
Experiment 2 Stage 1 Stage 2	55 under-graduate students 49 under-graduate students	To determine whether significantly more seeded errors can be detected in a spreadsheet model built based on the proposed methodology, compared to a model constructed based on an unstructured or conventional approach.	<u>Primary question:</u> The experiment can be used to assess whether the proposed structured methodology can reduce the occurrence of user-generated errors by enhancing the comprehensibility of spreadsheet models. <u>Secondary question(s):</u> The experiment can show whether a software engineering based methodology can enhance the comprehensibility of spreadsheet models in a practical situation. A comparison of error detection rates can be used to assess the effectiveness of the framework. The experiment can reveal the usefulness of software engineering principles in increasing quality by making spreadsheet models more understandable.
Experiment 3 Stage 1 Test 1 Test 2 Stage 2 Test 1 Test 2	19 post-graduate students 11 short course students 22 post-graduate students 12 short course students	To determine whether significantly more seeded errors can be detected in a spreadsheet model built based on the proposed methodology, compared to a model constructed based on an unstructured or conventional approach.	<u>Primary question:</u> The experiment can be used to assess whether the proposed structured methodology can reduce the occurrence of user-generated errors by enhancing the comprehensibility of spreadsheet models. <u>Secondary question(s):</u> The experiment can show whether a software engineering based methodology can enhance the comprehensibility of spreadsheet models in a practical situation. A comparison of error detection rates can be used to assess the effectiveness of the framework. The experiment can reveal the usefulness of software engineering principles in increasing quality by making spreadsheet models more understandable.

continued ...

Experiment	Subjects	Aim	Research Questions Addressed
Experiment 4			
Stage 1			
Test 1	24 post-graduate students		
Test 2	12 short course students		
Stage 2			
Test 1	20 post-graduate students		
Test 2	11 short course students		
		To determine whether significantly more seeded errors can be detected in a spreadsheet model built based on the proposed methodology, compared to a model constructed based on an unstructured or conventional approach.	<p><u>Primary question:</u> The experiment can be used to assess whether the proposed structured methodology can reduce the occurrence of user-generated errors by enhancing the comprehensibility of spreadsheet models.</p> <p><u>Secondary question(s):</u> The experiment can show whether a software engineering based methodology can enhance the comprehensibility of spreadsheet models in a practical situation.</p> <p>A comparison of error detection rates can be used to assess the effectiveness of the framework.</p> <p>The experiment can reveal the usefulness of software engineering principles in increasing quality by making spreadsheet models more understandable.</p>

Figure 8.9: Tabulated Summary of Experiments

CHAPTER 9

ANALYSIS OF RESULTS

9.1 Introduction

This chapter presents a detailed analysis of the results of the experiments conducted. In order to assess and establish the quality of the proposed structured methodology, four different experiments were carried out. The results of these experiments are analysed and presented in this chapter. The experiments themselves are described in detail in *Chapter 8: Evaluation Strategy and Experiments*.

9.2 Results of Experiments

9.2.1 Experiment 1

As described in *Chapter 8*, the experiment was carried out in two stages. Each stage consisted of two tests. Each test was performed on a different set of subjects. The first test was composed of two sessions. *Figure 9.1* displays a summary of the results obtained.

Stage 1	Test 1		Test 2
	Session 1	Session 2	
	without lesson	without lesson	without lesson
Subject Type	post-graduate students	post-graduate students	short course students
Subject Size	22	22	12
Mean	3.91	3.77	3.33
Standard Deviation	1.97	1.82	1.07
Stage 2	Session 1	Session 2	
	without lesson	with lesson	with lesson
	post-graduate students	post-graduate students	short course students
Subject Type	post-graduate students	post-graduate students	short course students
Subject Size	20	20	14
Mean	3.55	1.40	1.21
Standard Deviation	1.15	1.14	1.25

Figure 9.1: Summary of Results from *Experiment 1*

The following abbreviations are used to refer to the various parts of this experiment:

S1 T1 S1	Stage 1 - Test 1 - Session 1
S1 T1 S2	Stage 1 - Test 1 - Session 2
S1 T2	Stage 1 - Test 2
S2 T1 S1	Stage 2 - Test 1 - Session 1
S2 T1 S2	Stage 2 - Test 1 - Session 2
S2 T2	Stage 2 - Test 2

The purpose of this experiment was to establish whether adoption of the proposed structured methodology could result in a significant reduction in the number of errors committed when producing a spreadsheet model. This is compared against development of the model using an unstructured or conventional approach.

The results of *Test 1* were analysed to test this hypothesis. Firstly, the two sets of correlated dependent samples [S1 T1 S1 and S1 T1 S2] and [S2 T1 S1 and S2 T1 S2] were assessed. In order to find out if subjects performed significantly better in *Session 2* of *Stage 2* using the proposed structured methodology, compared to their performance in *Session 1* of *Stage 2* (using an unstructured approach), a *Wilcoxon Signed Rank Test* was carried out. This is the non-parametric equivalent of a *Paired T-Test*. A *Paired T-Test* could not be done as normality tests showed that the data from S2 T1 S2 were not normally distributed. The results of the normality tests are shown in *Appendix E: Figures E1 (a) and E1 (b)*.

The *Wilcoxon Signed Rank Test* revealed that subjects did in fact commit significantly fewer errors in *Session 2* of *Stage 2* where the proposed structured methodology was adopted in creating the spreadsheet model. *Figure 9.2 (a)* shows the results of the test. *Figure 9.2 (b)* shows a plot of the data obtained from S2 T1 S1 and S2 T1 S2. It can be clearly seen that on the whole, subjects in S2 T1 S2 committed fewer errors compared to subjects in S2 T1 S1. The raw data from S2 T1 S1 and S2 T1 S2 are presented in *Appendix F*.

Ranks				
		N	Mean Rank	Sum of Ranks
S2T1S2 - S2T1S1	Negative Ranks	14 ^a	7.50	105.00
	Positive Ranks	0 ^b	.00	.00
	Ties	6 ^c		
	Total	20		

a. S2T1S2 < S2T1S1

b. S2T1S2 > S2T1S1

c. S2T1S1 = S2T1S2

Test Statistics ^b	
	S2T1S2 - S2T1S1
Z	-3.375 ^a
Asymp. Sig. (2-tailed)	.001

a. Based on positive ranks.

b. Wilcoxon Signed Ranks Test

The 1-tailed P-Value (0.0005) is less than 0.01. Therefore, there is a very significant decrease in the number of errors committed in S2 T1 S2.

Figure 9.2 (a): Wilcoxon Signed Rank Test (in SPSS)

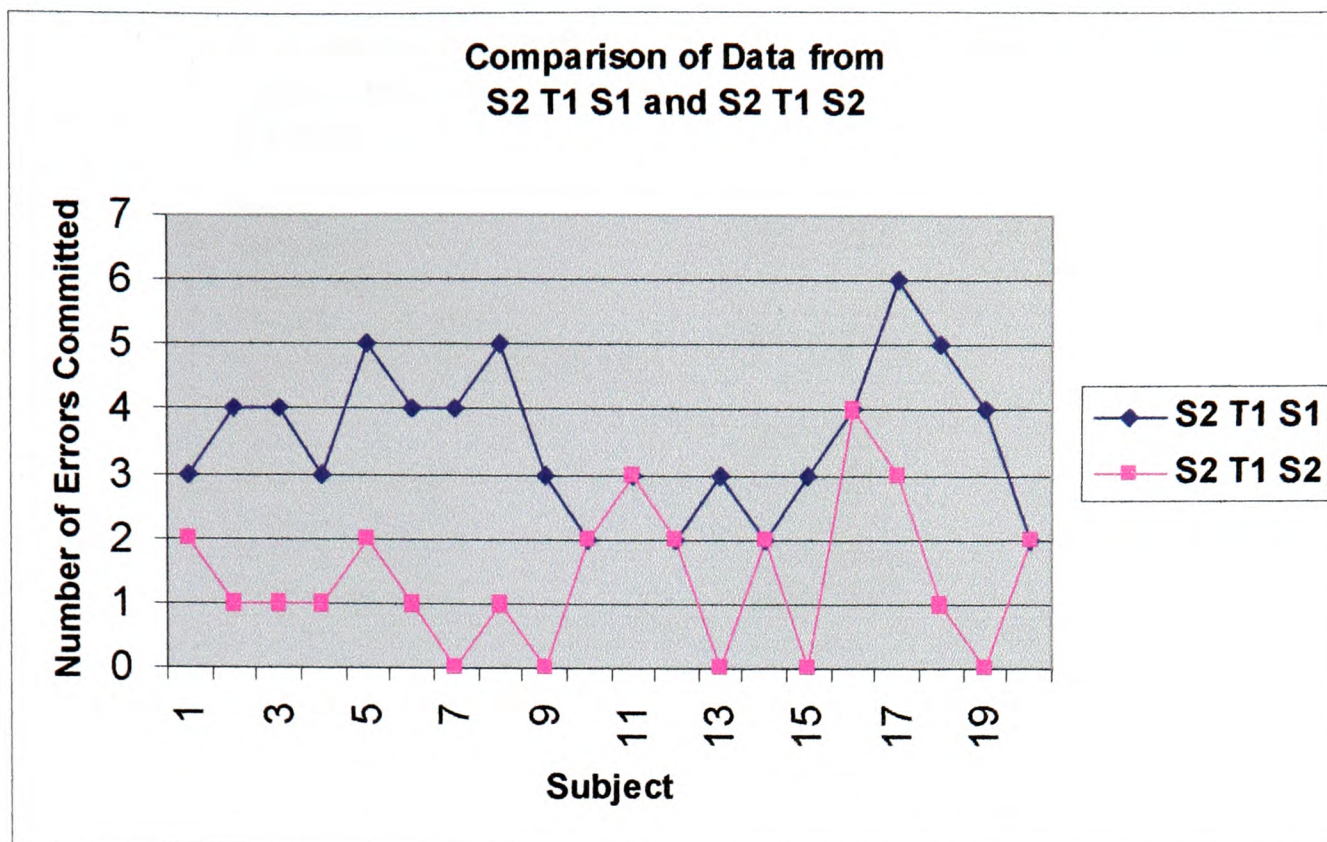


Figure 9.2 (b): Plot of Data from S2 T1 S1 and S2 T1 S2

An assessment of the results of the correlated dependent samples **S1 T1 S1** and **S1 T1 S2** was subsequently undertaken to establish whether there was a significant difference between the performance of participants in **S1 T1 S1** and **S1 T1 S2**. Any significant reduction in the number of errors could be attributable to the repeated task. A *Paired T-Test* was carried out as normality tests showed that the data from both samples were normally distributed. The results of the normality tests are displayed in *Appendix E: Figures E1 (c) and E1 (d)*.

The *Paired T-Test* revealed that there was no significant difference between the number of errors committed by subjects in **S1 T1 S1** and the same subjects in **S1 T1 S2**. The conclusion that can be drawn from this is that subjects who re-created the same model using the proposed methodology produced significantly fewer errors compared to participants who re-created the first model based on their own approach. The results of the *Paired T-Test* are shown in *Figure 9.3 (a)*. *Figure 9.3 (b)* shows a plot of the data obtained from **S1 T1 S1** and **S1 T1 S2**. It is obvious that there is no significant difference between the number of errors committed by subjects in **S1 T1 S1** and subjects in **S1 T1 S2**. The raw data from **S1 T1 S1** and **S1 T1 S2** are given in *Appendix F*.

t-Test: Paired Two Sample for Means		
5% level		
	<i>Variable 1</i>	<i>Variable 2</i>
Mean	3.909090909	3.772727273
Variance	3.896103896	3.326839827
Observations	22	22
Pearson Correlation	0.933079248	
Hypothesized Mean Difference	0	
df	21	
t Stat	0.900643547	
P(T<=t) one-tail	0.18899549	
t Critical one-tail	1.720743512	
P(T<=t) two-tail	0.377990979	
t Critical two-tail	2.079614205	

The test statistic 0.9 is NOT in the critical region. Therefore, there is NO evidence of a significant difference in the number of errors committed between S1 T1 S1 and S1 T1 S2.

Figure 9.3 (a): Paired T-Test Result (in MS Excel)

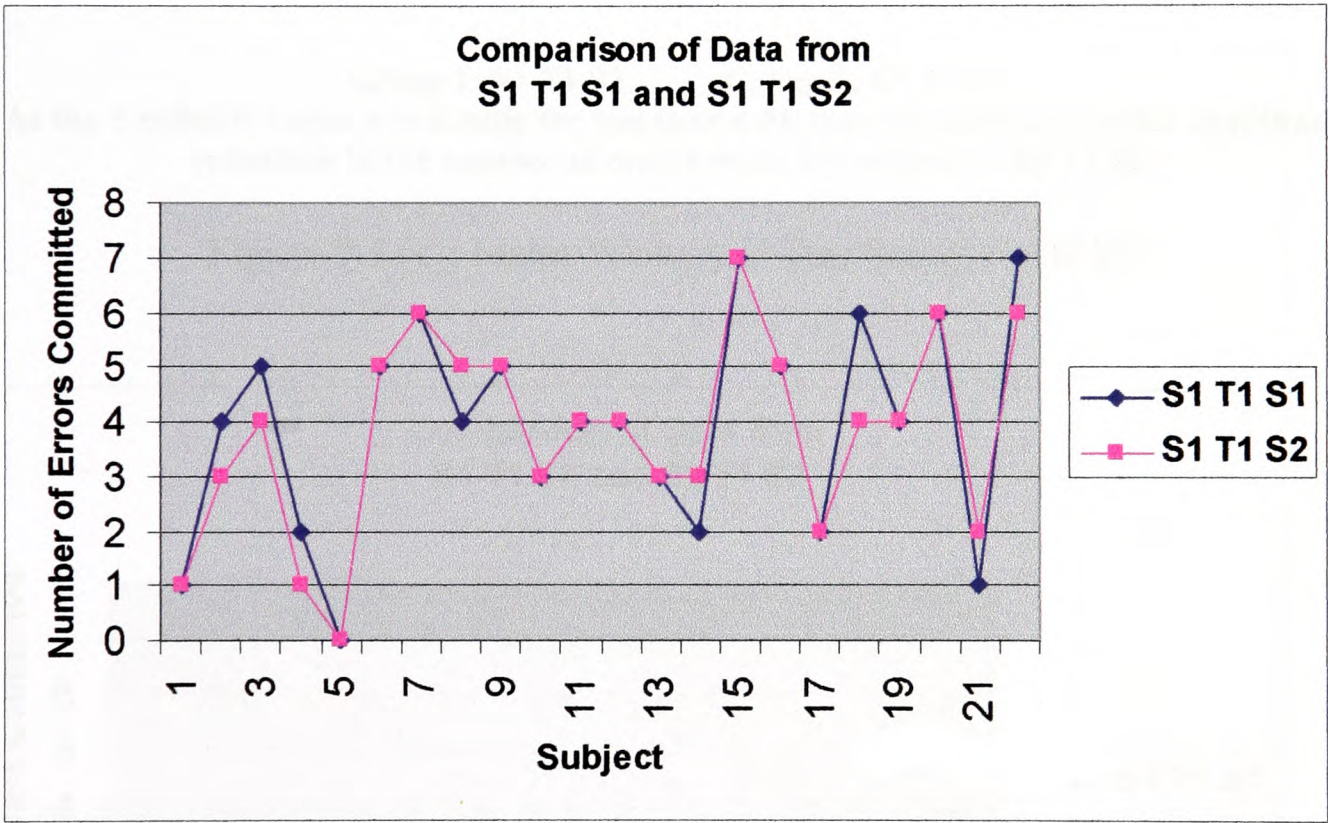
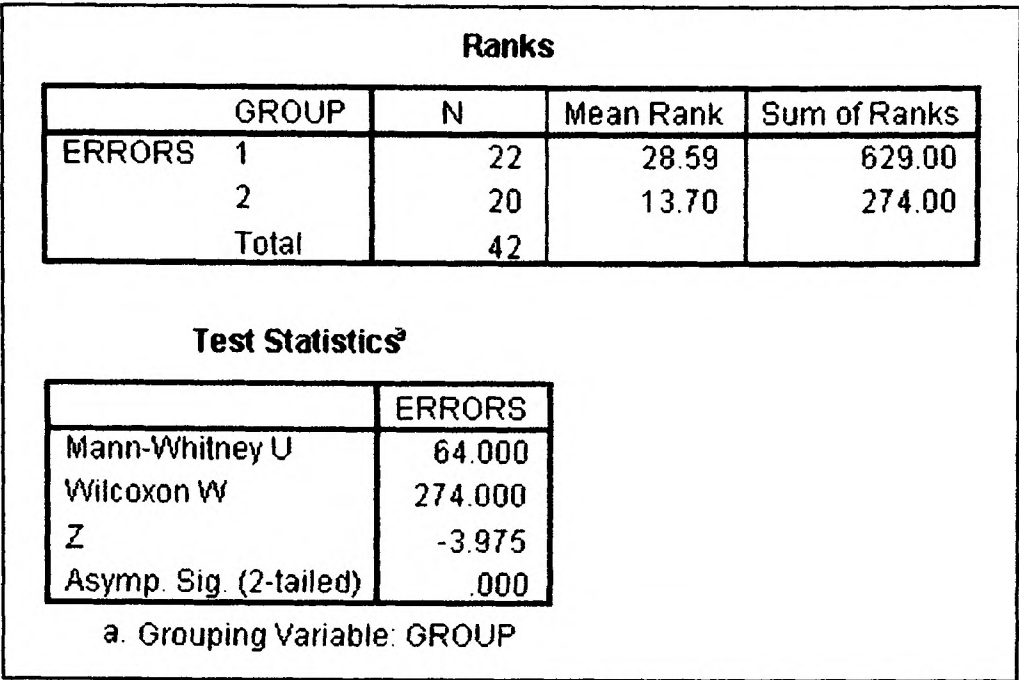


Figure 9.3 (b): Plot of Data from S1 T1 S1 and S1 T1 S2

The data from S1 T1 S2 and S2 T1 S2 were then compared. As mentioned previously, subjects in S1 T1 S2 repeated the creation of the first spreadsheet model using their own approaches and techniques. On the contrary, in S2 T1 S2 a different set of subjects repeated the construction of the first model having had a tutorial on the proposed structured methodology. A normality test has demonstrated that the data from S2 T1 S2 is not from a normal distribution. This is shown in *Appendix E: Figure E1 (b)*. As such, a non-parametric test for independent samples had to be performed. A *Mann-Whitney U Test* carried out distinctly showed that participants using the proposed methodology (S1 T1 S2) committed significantly fewer errors compared to subjects adopting their own methods (S2 T1 S2). The results of the *Mann-Whitney U*

Test can be seen in Figure 9.4 (a). Figure 9.4 (b) shows a plot of the data obtained from S1 T1 S2 and S2 T1 S2. It can be distinctly seen that on the whole, subjects in S1 T1 S2 committed fewer errors compared to subjects in S2 T1 S2. The raw data from S1 T1 S2 and S2 T1 S2 are presented in Appendix F.



Group 1: S1 T1 S2 Group 2: S2 T1 S2
As the 1-tailed P-Value is evidently far less than 0.01, there is proof of a highly significant reduction in the number of errors made by subjects in S2 T1 S2.

Figure 9.4 (a): Mann-Whitney U Test Result (in SPSS)

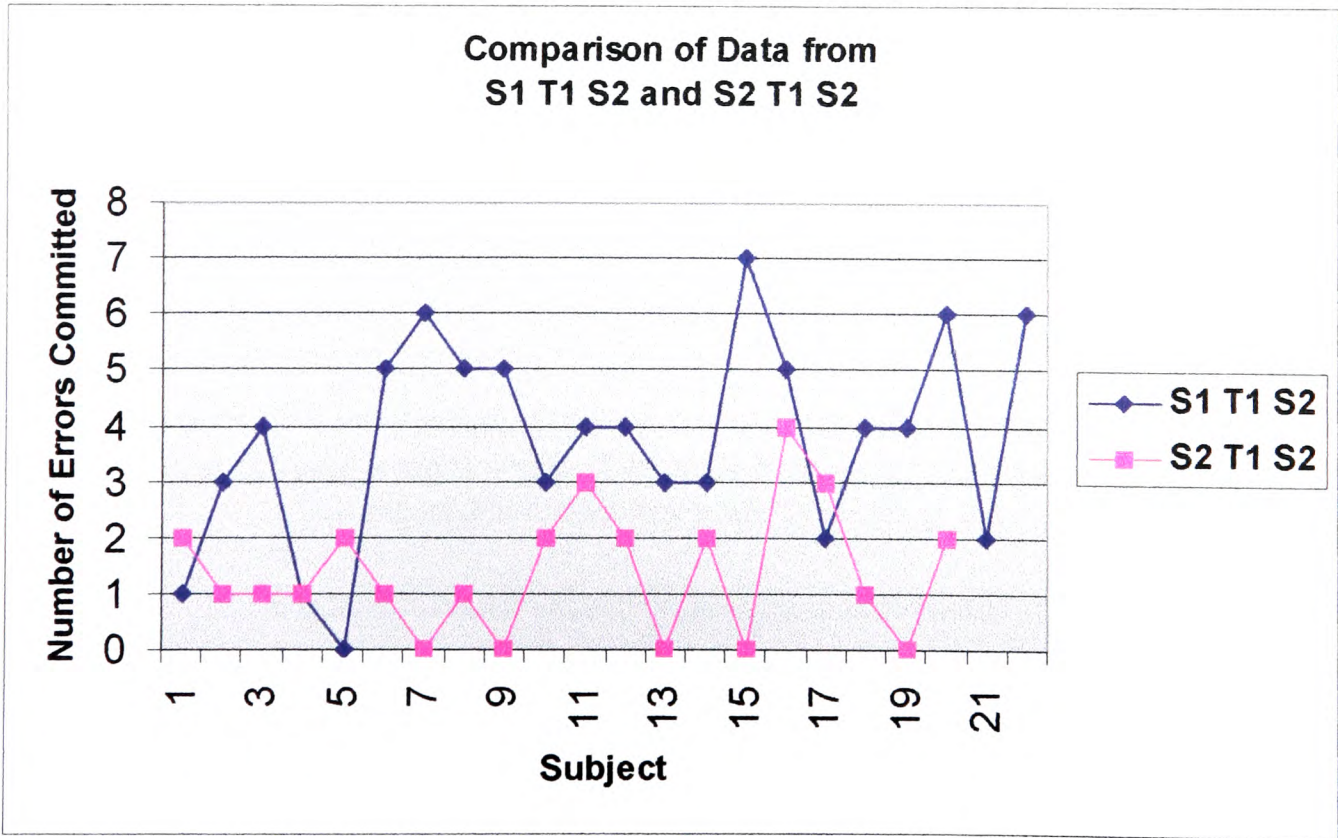


Figure 9.4 (b): Plot of Data from S1 T1 S2 and S2 T1 S2

Finally, the results of *Test 2* were analysed. *Test 2* was conducted in two stages. In *Stage 1*, a group of short course students had to build a spreadsheet model without any help or guidance. In *Stage 2*, a different group of short course students were given a tutorial on the proposed methodology prior to the creation of the spreadsheet model.

A non-parametric test for the independent samples had to be performed instead of a *T-Test* as the data from **S2 T2** (Stage 2 - *Test 2*) was found not to be normally distributed. A normality test, however, showed that the data from **S1 T2** (Stage 1 - *Test 2*) was in fact normally distributed, although this did not make a difference. The results of the normality tests can be seen in *Appendix E: Figures E1 (e) and E1 (f)*.

The *Mann-Whitney U Test* performed provided extremely strong evidence that subjects using the proposed methodology (**S2 T2**) committed significantly fewer errors compared to participants using their own methods and techniques (**S1 T2**). The results of the *Mann-Whitney U Test* are shown in *Figure 9.5 (a)*. *Figure 9.5 (b)* shows a plot of the data obtained from **S1 T2** and **S2 T2**. It can be clearly seen that on the whole, subjects in **S2 T2** committed fewer errors compared to subjects in **S1 T2**. The raw data from **S1 T2** and **S2 T2** are presented in *Appendix F*.

Ranks				
GROUP		N	Mean Rank	Sum of Ranks
ERRORS	1	12	19.00	228.00
	2	14	8.79	123.00
	Total	26		

Test Statistics ^b	
	ERRORS
Mann-Whitney U	18.000
Wilcoxon W	123.000
Z	-3.452
Asymp. Sig. (2-tailed)	.001
Exact Sig. [2*(1-tailed Sig.)]	.000 ^a

a. Not corrected for ties.

b. Grouping Variable: GROUP

Group 1: S1 T2

Group 2: S2 T2

As the 1-tailed P-Value is far less than 0.01, there is evidence of a highly significant reduction in the number of errors made by subjects in S2 T2.

Figure 9.5 (a): Mann-Whitney U test result (in SPSS)

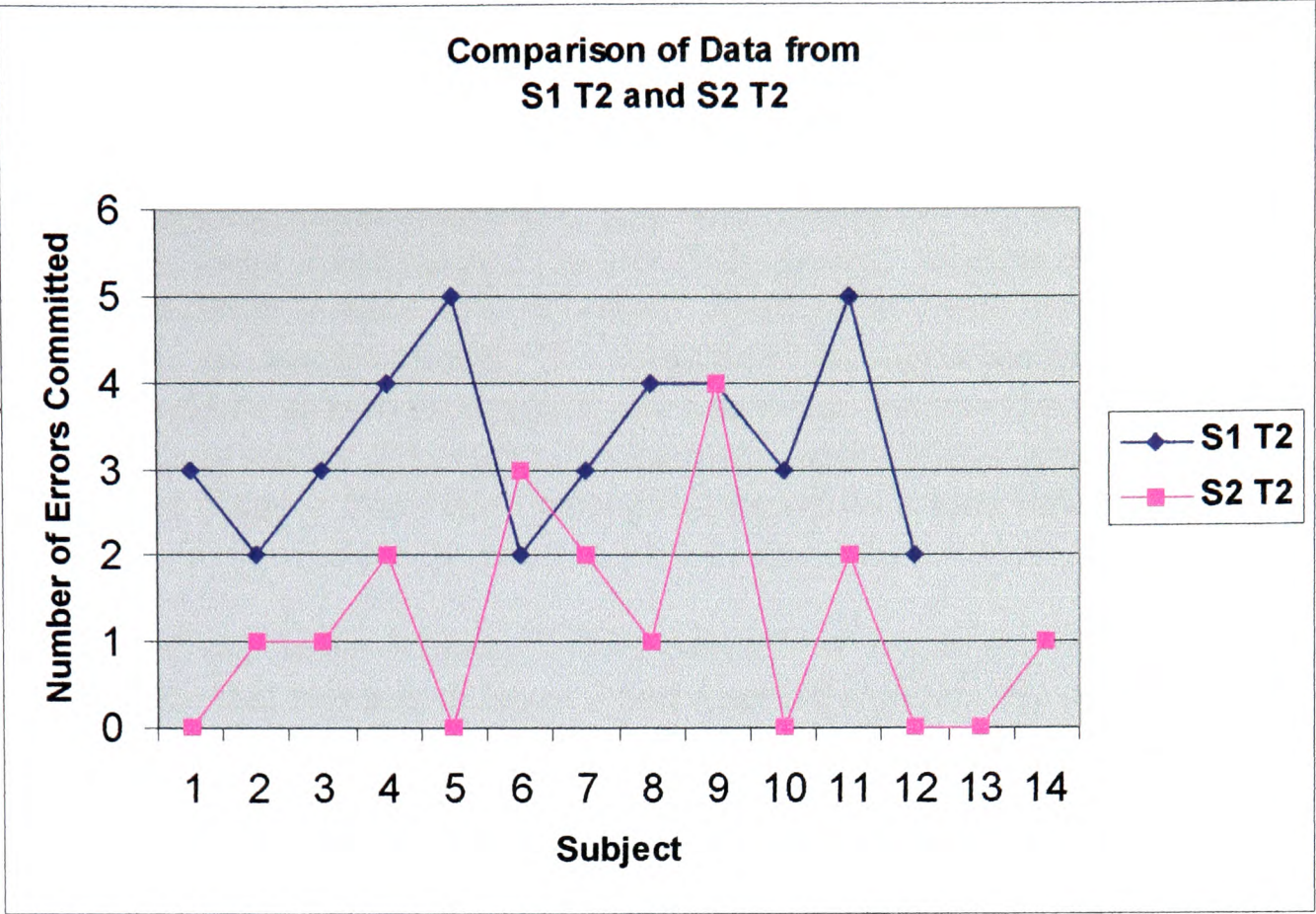


Figure 9.5 (b): Plot of Data from S1 T2 and S2 T2

9.2.2 Experiment 2

Experiment 2 is elaborately described in *Chapter 8: Evaluation Strategy and Experiments*. The experiment was conducted in two stages. Each involved a different group of under-graduate students. However, they were all from the same school and at the same academic stage. *Figure 9.6* displays a summary of the results obtained.

Stage 1	without lesson
Subject Type	under-graduate students
Subject Size	55
% of Errors Detected	33
Mean	3.91
Standard Deviation	1.78

Stage 2	with lesson
Subject Type	under-graduate students
Subject Size	49
% of Errors Detected	72
Mean	8.61
Standard Deviation	2.71

Figure 9.6: Summary of Results from *Experiment 2*

The first step was to perform normality tests on the two sets of data to determine whether each sample was from a normal distribution. The normality tests demonstrated that the data from both samples were NOT normally distributed.

Therefore, a *Mann-Whitney U test* for independent samples had to be carried out instead of a *two-sample t-test*. The results of the normality tests are shown in *Appendix E: Figures E2 (a) and E2 (b)*.

The non-parametric *Mann-Whitney U test* was carried out on the two independent samples, from *Stage 1* and *Stage 2*, to establish whether subjects participating in the experiment at *Stage 2* were able to detect significantly more errors that had been seeded into the spreadsheet model. The hypothesis was that subjects of the experiment at *Stage 2* would be able to accomplish this due to the fact that the spreadsheet model had been built based on the proposed structured methodology. On the contrary, the model used in *Stage 1* had been constructed based on a conventional, unstructured approach.

The results of the *Mann-Whitney U test* proved the hypothesis by providing very strong evidence that subjects in *Stage 2* had detected significantly more seeded errors compared to subjects in *Stage 1*. This can be regarded as testimony to the increased comprehensibility of the model built based on the proposed methodology. The results of the *Mann-Whitney U test* are displayed in *Figure 9.7 (a)*. *Figure 9.7 (b)* shows a plot of the data obtained from *Stage 1* and *Stage 2*. It can be distinctly seen that on the whole, subjects in *Stage 2* detected more errors compared to subjects in *Stage 1*. The raw data from *Stage 1* and *Stage 2* are given in *Appendix F*.

Ranks				
STAGE		N	Mean Rank	Sum of Ranks
ERRORS	1	55	32.77	1802.50
	2	49	74.64	3657.50
	Total	104		

Test Statistics ^a	
	ERRORS
Mann-Whitney U	262.500
Wilcoxon W	1802.500
Z	-7.105
Asymp. Sig. (2-tailed)	.000

a. Grouping Variable: STAGE

As the 1-tailed P-Value is far less than 0.01, there is evidence of a highly significant increase in the number of errors detected by subjects in Stage 2.

Figure 9.7 (a): Mann-Whitney U Test Result (in SPSS)

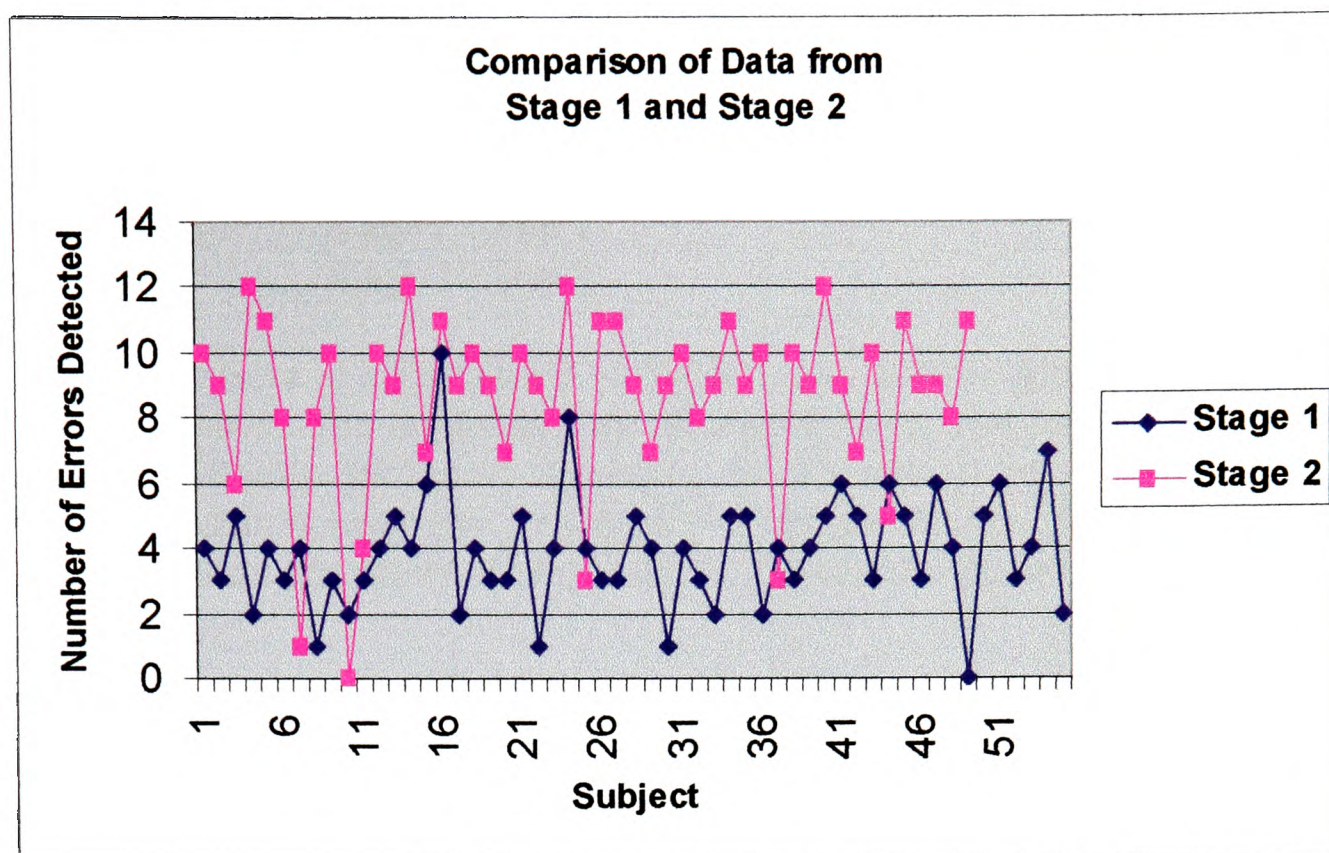


Figure 9.7 (b): Plot of Data from Stage 1 and Stage 2

9.2.3 Experiment 3

In *Experiment 3*, two tests were carried out. Each test involved a different group of subjects belonging to the same subject type, i.e. post-graduate students, undergraduate students, etc. In both *Tests 1* and *2*, subjects had to detect seeded errors in a spreadsheet model built using a conventional approach, in *Stage 1*. In *Stage 2*, a different set of participants would try and identify the same errors seeded into the model re-structured based on the proposed methodology. This is preceded by a brief tutorial. The results of the experiment are shown in *Figure 9.8*.

	Test 1	Test 2
Stage 1	<i>without lesson</i>	<i>without lesson</i>
Subject Type	post-graduate students	short course students
Subject Size	19	11
% of Errors Detected	23	24
Mean	2.26	2.36
Standard Deviation	1.63	1.69
Stage 2	<i>with lesson</i>	<i>with lesson</i>
Subject Type	post-graduate students	short course students
Subject Size	22	12
% of Errors Detected	52	58
Mean	5.23	5.83
Standard Deviation	2.89	2.44

Figure 9.8: Summary of Results from *Experiment 3*

The following abbreviations are used to refer to the various parts of this experiment:

S1 T1	Stage 1 - Test 1
S1 T2	Stage 1 - Test 2
S2 T1	Stage 2 - Test 1
S2 T2	Stage 2 - Test 2

Normality tests were initially carried out to ascertain whether each sample was from a normal distribution. The normality tests showed that the data from all four samples were indeed normally distributed. Therefore, *Two-Sample T-Tests* could be conducted on the independent samples [S1 T1 and S2 T1] and [S1 T2 and S2 T2]. The results of the normality tests are shown in *Appendix E: Figures E3 (a), E3 (b), E3 (c) and E3 (d)*.

Firstly, a *Two-Sample T-Test* was used to analyse the results of *Test 1*. It was carried out on the independent samples S1 T1 and S2 T1. The subjects of the test were post-graduate students. The results of the *T-test*, with equal variances not assumed, distinctly revealed that participants of the experiment in S2 T1 detected significantly more seeded errors compared to subjects in S1 T1. The spreadsheet model used in S2 T1 had been created based on the proposed methodology. The results can be seen in *Figure 9.9 (a)*. *Figure 9.9 (b)* shows a plot of the data obtained from S1 T1 and S2 T1. It can be clearly seen that on the whole, subjects in S2 T1 were able to detect more errors compared to subjects in S1 T1. The raw data from S1 T1 and S2 T1 are presented in *Appendix F*.

Group Statistics

	GROUP	N	Mean	Std. Deviation	Std. Error Mean
ERRORS	S1 T1	19	2.26	1.63	.37
	S2 T1	22	5.23	2.89	.62

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
ERRORS	Equal variances assumed	8.407	.006	-3.953	39	.000	-2.96	.75	-4.48	-1.45
	Equal variances not assumed			-4.110	33.896	.000	-2.96	.72	-4.43	-1.50

The P-Value of the F-test is 0.006. As this is less than 0.05, equal variances are not assumed. The 1-tailed P-Value of the T-test is evidently far less than 0.01. Therefore, there is an extremely significant increase in the number of errors detected by subjects in S2 T1.

Figure 9.9 (a): Two-Sample T-Test (in SPSS)

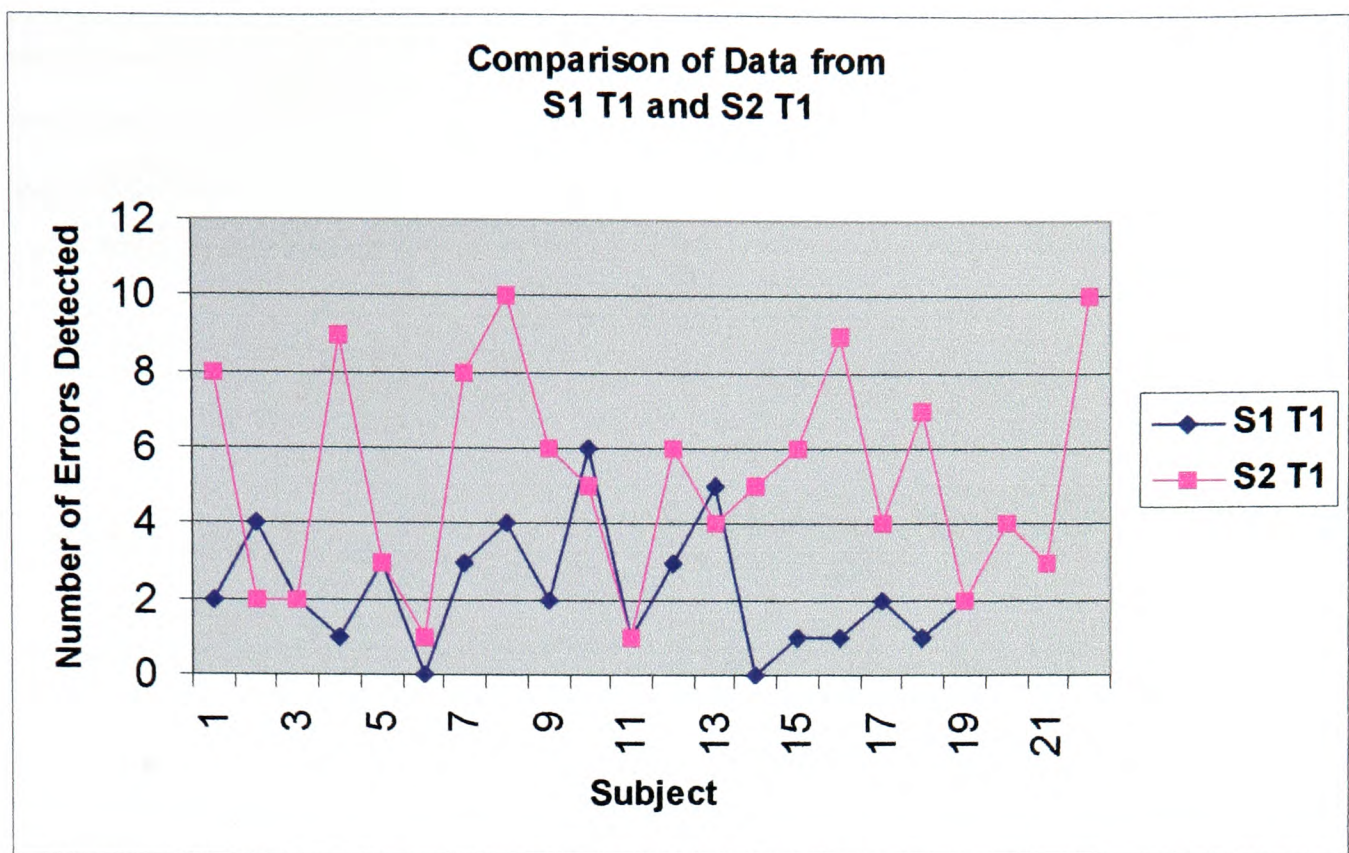


Figure 9.9 (b): Plot of Data from S1 T1 and S2 T1

Secondly, a *Two-Sample T-Test* was also used in the analysis of the results of *Test 2*. It was performed on the independent samples **S1 T2** and **S2 T2**. The subjects of *Test 2* were short course students. The results of the *T-test* provided very strong evidence that subjects of the experiment in **S2 T2** detected significantly more seeded errors compared to subjects in **S1 T2**. The spreadsheet model used in **S2 T2** had been created based on the proposed methodology while **S1 T2** involved a model built using a conventional approach. The results are shown in *Figure 9.10 (a)*. *Figure 9.10 (b)* shows a plot of the data obtained from **S1 T2** and **S2 T2**. It is clearly evident that on the whole, subjects in **S2 T2** detected more errors compared to subjects in **S1 T2**. The raw data from **S1 T2** and **S2 T2** are presented in *Appendix F*.

Group Statistics

	GROUP	N	Mean	Std. Deviation	Std. Error Mean
ERRORS	S1 T2	11	2.36	1.69	.51
	S2 T2	12	5.83	2.44	.71

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
ERRORS	Equal variances assumed	.903	.353	-3.924	21	.001	-3.47	.88	-5.31	-1.63
	Equal variances not assumed			-3.988	19.602	.001	-3.47	.87	-5.29	-1.65

The P-Value of the F-test is 0.353. As this is greater than 0.05, equal variances are assumed. The 1-tailed P-Value of the T-test (0.0005) is obviously far less than 0.01. Therefore, there is an extremely significant increase in the number of errors detected by subjects in S2 T2.

Figure 9.10 (a): Two-Sample T-Test (in SPSS)

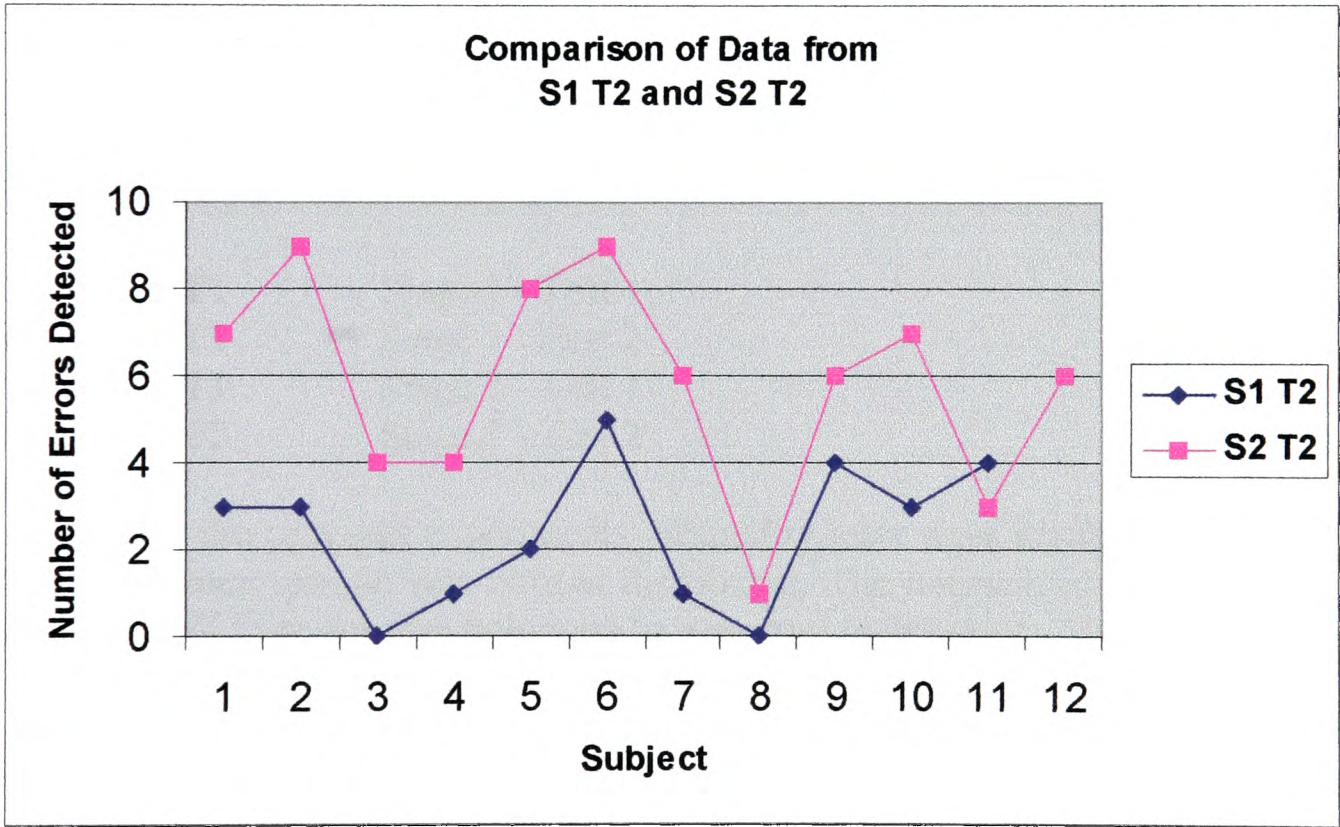


Figure 9.10 (b): Plot of Data from S1 T2 and S2 T2

9.2.4 Experiment 4

The nature and structure of *Experiment 4* are identical to those of *Experiment 3*. The main difference between the two experiments was that different spreadsheet models were used. As performed in *Experiments 2* and *3*, subjects had to detect seeded errors in a model that had been produced either based on a conventional approach or the proposed structured methodology. The results of *Experiment 4* are displayed in *Figure 9.11*.

	Test 1	Test 2
Stage 1	without lesson	without lesson
Subject Type	post-graduate students	short course students
Subject Size	24	12
% of Errors Detected	44	37
Mean	4.42	3.67
Standard Deviation	1.98	2.35
Stage 2	with lesson	with lesson
Subject Type	post-graduate students	short course students
Subject Size	20	11
% of Errors Detected	68	78
Mean	6.75	7.82
Standard Deviation	2.36	2.04

Figure 9.11: Summary of Results from *Experiment 4*

The following abbreviations are used to refer to the various parts of this experiment:

S1 T1	Stage 1 - Test 1
S1 T2	Stage 1 - Test 2
S2 T1	Stage 2 - Test 1
S2 T2	Stage 2 - Test 2

As done in the previous experiments, normality tests were carried out to check if the data from each sample was normal distributed. The normality tests revealed that the data from all four samples belonged to a normal distribution. Therefore, *Two-Sample T-Tests* were performed on the independent samples [S1 T1 and S2 T1] and [S1 T2 and S2 T2]. The results of the normality tests are displayed in *Appendix E: Figures E4 (a), E4 (b), E4 (c) and E4 (d)*.

A *Two-Sample T-Test* was used to analyse the results of *Test 1*. It was performed on the independent samples S1 T1 and S2 T1. The subjects were post-graduate students. The results of the *T-test* distinctly proved that subjects of the experiment in S2 T1 were able to identify significantly more seeded errors compared to subjects in S1 T1. This can be attributable to the fact that the spreadsheet model used in S2 T1 had been created based on the proposed methodology. The results can be seen in *Figure 9.12 (a)*. *Figure 9.12 (b)* shows a plot of the data obtained from S1 T1 and S2 T1. It is obvious that on the whole, subjects in S2 T1 identified more errors compared to subjects in S1 T1. The raw data from S1 T1 and S2 T1 are presented in *Appendix F*.

Group Statistics					
GROUP		N	Mean	Std. Deviation	Std. Error Mean
ERRORS	S1 T1	24	4.42	1.98	.40
	S2 T1	20	6.75	2.36	.53

Independent Samples Test										
		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
ERRORS	Equal variances assumed	1.455	.234	-3.571	42	.001	-2.33	.65	-3.65	-1.01
	Equal variances not assumed			-3.514	37.208	.001	-2.33	.66	-3.68	-.99

The P-Value of the F-test is 0.234. As this is greater than 0.05, equal variances are assumed. The 1-tailed P-Value of the T-test (0.0005) is far less than 0.01. Therefore, there is a highly significant increase in the number of errors detected by subjects in S2 T1.

Figure 9.12 (a): Two-Sample T-Test (in SPSS)

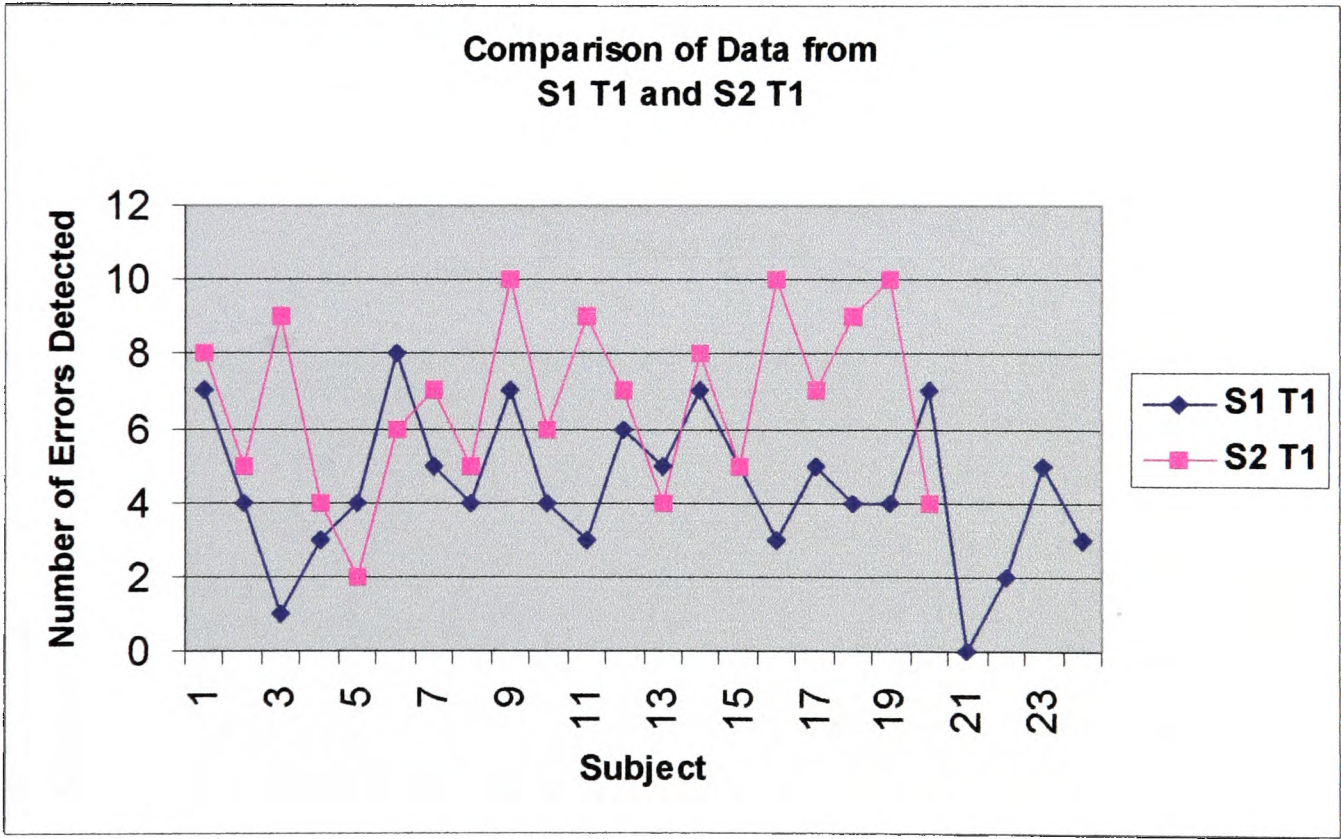


Figure 9.12 (b): Plot of Data from S1 T1 and S2 T1

A Two-Sample T-Test was subsequently conducted to analyse the results of Test 2. It was carried out on the independent samples S1 T2 and S2 T2. The subjects of Test 2 were short course students. The results of the T-test provided extremely strong evidence that subjects of the experiment in S2 T2 performed far better by detecting significantly more seeded errors compared to subjects in S1 T2. The spreadsheet model used in S2 T2 had been produced according to the proposed methodology while S1 T2 involved the model built based on a conventional approach. The results are shown in Figure 9.13 (a). Figure 9.13 (b) shows a plot of the data obtained from

S1 T2 and S2 T2. It is obvious that on the whole, subjects in S2 T2 detected more errors compared to subjects in S1 T2. The raw data from S1 T2 and S2 T2 are provided in *Appendix F*.

Group Statistics

	GROUP	N	Mean	Std. Deviation	Std. Error Mean
ERRORS	S1 T2	12	3.67	2.35	.68
	S2 T2	11	7.82	2.04	.62

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
ERRORS	Equal variances assumed	.352	.559	-4.506	21	.000	-4.15	.92	-6.07	-2.24
	Equal variances not assumed			-4.535	20.949	.000	-4.15	.92	-6.06	-2.25

The P-Value of the F-test is 0.559. As this is far greater than 0.05, equal variances are assumed. It is very obvious that the 1-tailed P-Value of the T-test is far less than 0.01. Therefore, there is a highly significant increase in the number of errors detected by subjects in S2 T2.

Figure 9.13 (a): Two-Sample T-Test (in SPSS)

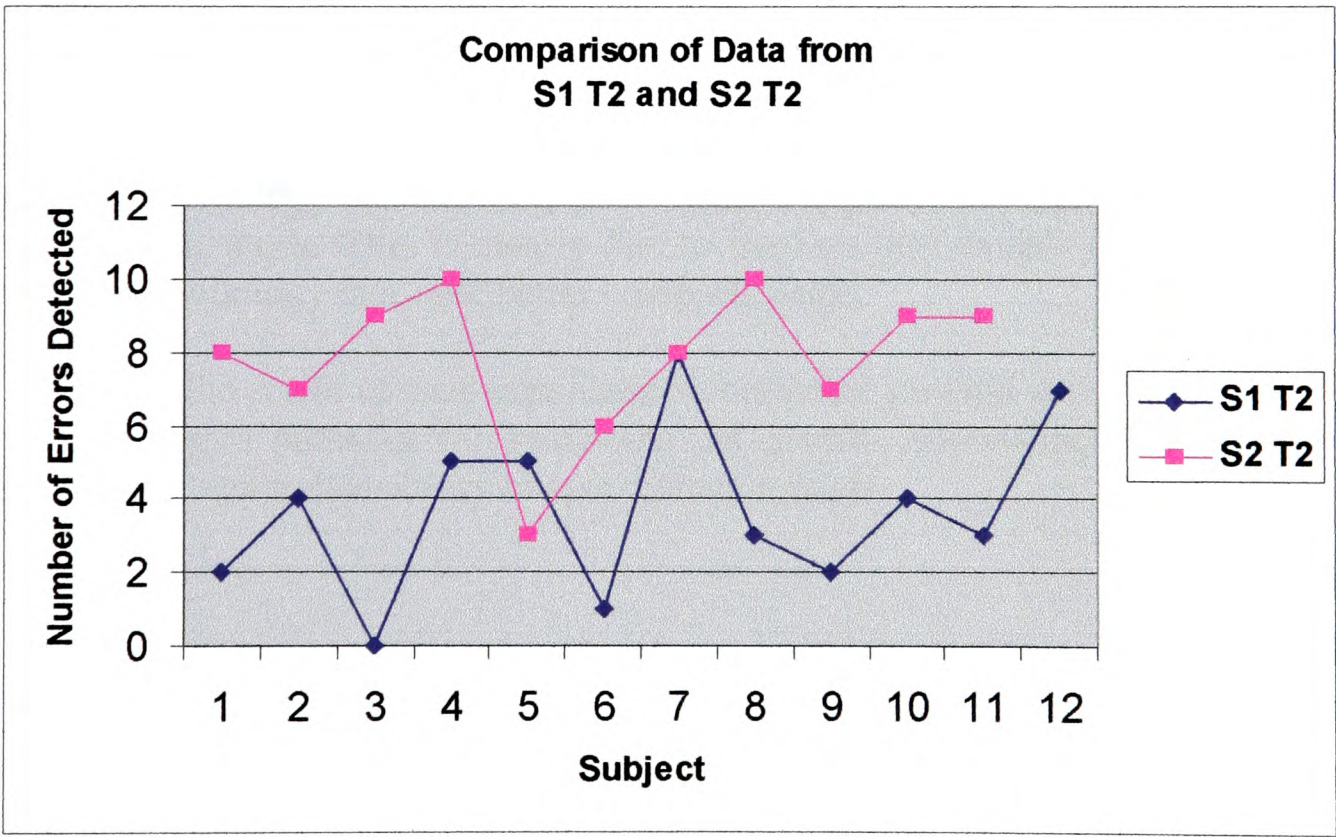


Figure 9.13 (b): Plot of Data from S1 T2 and S2 T2

9.3 Summary

The purpose of *Experiment 1* was to determine whether employing the proposed structured methodology could result in a significant reduction in the number of errors committed when producing a spreadsheet model. This inevitably involves a comparison with the development of the model using an unstructured or conventional approach. The principal objective of *Experiments 2, 3 and 4* was to establish whether spreadsheet models built based on the proposed structured methodology were more comprehensible by evaluating users' capacity to detect seeded errors.

The results obtained from *Experiment 1* have distinctly and consistently demonstrated that there is a drastic decrease in the number of user-generated errors committed by subjects adopting the proposed structured methodology to develop a spreadsheet model. On the contrary, subjects produced significantly more errors when their own methods and techniques were employed. A combination of several appropriate statistical tests and techniques, namely *Normality* tests, the *Wilcoxon Signed Rank* test, a *Paired T-Test* and *Mann-Whitney U* tests, were used to analyse the results of *Experiment 1*. The conclusion that can be drawn from this experiment is that the proposed methodology has the potential to reduce the occurrence of user-generated errors in spreadsheet models.

The results of *Experiments 2, 3 and 4* proved beyond any doubt that subjects were able to detect considerably more seeded errors in spreadsheet models built based on the proposed structured methodology, in comparison to models created using conventional unstructured or less structured methods. An analysis of the data using techniques such as *Normality* tests, *F-tests*, *Two-Sample T-tests* and *Mann-Whitney U* tests, revealed that all the results were statistically significant. They strongly supported the hypothesis that the error detection rate in models built based on the proposed methodology was considerably higher compared to those constructed using conventional methods. This is clearly due to the fact that models conforming to the proposed methodology facilitate better comprehension.

The results of the series of four experiments conducted provide adequate testimony to the methodology's potential for enhancing the quality, controlling the integrity and improving the comprehensibility of spreadsheet models.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

10.1 Conclusions

Important contributions have been made in this research programme. The primary question posed in the research is whether a structured methodology can be developed for the integrity control of spreadsheet models and if such a framework can reduce the occurrence of user-generated errors. The most significant contribution of this research programme is a structured methodology for the development and integrity control of spreadsheet models. Through the various experiments conducted and a meticulous analysis of their results, the proposed methodology's potential for integrity control has been demonstrated. The proposed methodology can reduce the occurrence of user-generated errors by ensuring consistency in the spreadsheet model development process and producing more comprehensible, reliable and maintainable models.

It is of utmost importance to gain a thorough insight into the nature and properties of spreadsheet errors in order to develop an effective methodology for controlling the integrity of spreadsheet models. A secondary research question concerns the possibility of developing a classification of user-generated spreadsheet errors based on a rational taxonomic scheme. The construction of the proposed methodology was preceded and inspired by a more comprehensive classification of user-generated errors, than presented before, based on systematic taxonomic methods. This is an immensely important by-product of the research and clearly establishes the possibility of developing a comprehensive classification or taxonomy of the different types of user-generated spreadsheet errors based on a rational taxonomic scheme. The provision of this taxonomy offers an extremely important means of comprehending, analysing and comparing the different types of spreadsheet errors.

The next secondary research question asks what framework for spreadsheet model development is most likely to be optimum in a practical situation. A thorough investigation of past work on the phenomenon of spreadsheet errors has revealed an urgent need to adopt a structured and software engineering based methodology as an optimum framework for spreadsheet development in a practical situation. The proposed methodology represents a new approach or paradigm to the provision of such a discipline for the development of spreadsheet models. As explained in Chapter 6 (Section 6.3), at the present time, the structured techniques based on *Jackson structural forms* appear to be the most approachable due to its simplicity in concept, maturity and likely acceptance by spreadsheet users. More sophisticated approaches such as *object-oriented methods* and *entity relationship modelling* might become feasible as the practice develops industrially.

Based on one of the secondary research questions, an investigation is conducted into the possibility of applying software engineering principles to the process of spreadsheet model building to help improve the quality of the models. Structured techniques and software engineering principles form the foundation and backbone to the proposed methodology. The rigorous application of structured methods and established software engineering principles makes this a novel structured

methodology for the development and quality control of spreadsheet models. The methodology consists of numerous software engineering based methods and techniques that are effectively applied to the process of spreadsheet model building. The experiments conducted clearly demonstrated that the principles and techniques advocated within the methodology have the potential to improve the quality of spreadsheet models.

In order to answer the secondary research question on how effective the proposed framework is, the various features of the proposed structured methodology are tested on a range of spreadsheet models through a series of experiments. The results of the various experiments performed have distinctly and consistently demonstrated that the proposed methodology has tremendous potential to drastically reduce the incidence of errors and enhance the comprehensibility of spreadsheet models. This provides a very strong testimony to the effectiveness of the proposed structured methodology.

In conclusion, the research programme has established that a structured methodology for the integrity control of spreadsheet models can indeed be produced. The framework, primarily based on software engineering principles, can be applied to the development of spreadsheet models and decrease the occurrence of user-generated errors. This represents a significant contribution of additional knowledge and novel methods to the area of integrity control of spreadsheet models and structured spreadsheet development.

10.2 Future Work

The aim of the research was to create a methodology for improving the quality of spreadsheet models. However, this should not impose considerable extra burden on model developers. To this end, it will be immensely beneficial to build an automatic support tool to assist in the design, structuring and implementation of the spreadsheet model. This will take the form of a CASE (Computer-Aided Software Engineering) tool.

Future work on this project is envisaged on two important issues. The first is to produce an automatic spreadsheet-engineering tool to assist in the production of new spreadsheet models based on the proposed methodology. This tool should encompass both the front-end and back-end phases of the spreadsheet building process. In the front-end, it should offer facilities to produce the conceptual and logical designs in the form of *Jackson structures*. In the back-end stages, the tool should be able to automatically map the logical design onto an implemented model schema. Apart from that, it should also have mechanisms to perform the various update operations without affecting the integrity of the model.

The second issue concerns the re-engineering of existing spreadsheet models built based on conventional or unstructured methods. The tool should have a reverse-engineering function to extract information on structure from existing spreadsheets, and translate these models into structured form, based on the proposed methodology.

REFERENCES

[Aktas-87]

Aktas, A.Z. (1987) *Structured analysis & design of information systems*. Englewood Cliffs, NJ: Prentice-Hall.

[Ayalew-00]

Ayalew, Y., Clermont, M. and Mittermeir, R.T. (2000) "Detecting errors in spreadsheets". In: Chadwick, D. (ed.) (2000) *EuSpRIG 2000 Symposium proceedings - spreadsheet risks, audit and development methods*. London: University of Greenwich, pp. 51-62.

[Batson-86]

Batson, J. (1986) "Spreadsheet good practice", *Accountants Digest*, 200 (Winter).

[Batson-91]

Batson, J. and Brown, A. (1991) "Spreadsheet modelling best practice", *Accountants Digest*, 2730 (Autumn).

[Bell-00]

Bell, D. (2000) *Software engineering – a programming approach (third edition)*. Harlow: Addison-Wesley.

[Benham-93]

Benham, H., Delaney, M. and Luzi, A. (1993) "Structured techniques for successful end user spreadsheets", *Journal of End User Computing*, 5(2), pp. 18-25.

[Bodily-86]

Bodily, S.E. (1986) "Spreadsheet modeling as a stepping stone", *Interfaces*, 16(5), pp. 34-52.

[Booch-94]

Booch, G. (1994) *Object-oriented analysis and design with applications*. Redwood City, CA: Benjamin Cummings.

[Britannica.com-99-00]

Britannica.com (1999-2000) *Website: <http://www.britannica.com>*. Britannica.com and Encyclopaedia Britannica, Inc.

[Brown-87]

Brown, P.S. and Gould, J.D. (1987) "An experimental study of people creating spreadsheets", *ACM Transactions on Office Information Systems*, 5(3), pp. 258-272.

[Burgess-87]

Burgess, R.S. (1987) *Structured program design using JSP*. London: Hutchinson.

[Business Week-84]

Business Week (1984) "How personal computers can trip up executives", *Business Week*, 2861 (September), pp. 94-102.

[Butler-92]

Butler, R. (1992) *Tax audit study*. United Kingdom: HM Customs & Excise.

[Butler-97]

Butler, R. (1997) "The subversive spreadsheet", *DataWatch (ISACA London, UK)*.

[Butler-00]

Butler, R.J. (2000) "Is this spreadsheet a tax evader? how H.M. Customs & Excise tax test spreadsheet applications". In: Sprague, R.H., Jr. (ed.) (2000) *Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences 2000 – abstracts and CD-ROM of full papers*. California: IEEE Computer Society.

[Butler-00a]

Butler, R. (2000) "Risk assessment for spreadsheet developments". In: Chadwick, D. (ed.) (2000) *EuSpRIG 2000 Symposium proceedings - spreadsheet risks, audit and development methods*. London: University of Greenwich, pp. 65-74.

[Cameron-83]

Cameron, J.R. (1983) *JSP & JSD: The Jackson approach to software development*. Silver Spring, MD: IEEE Computer Society.

[Cale-94]

Cale, E.G., Jr. (1994). "Quality issues for end-user developed software", *Journal of Systems Management*, 45(1), pp. 36-39.

[Carlsson-89]

Carlsson, S.V. (1989) "Why Johnny can't or won't spreadsheet", *Scandinavian Journal of Information Systems*, 1, pp. 118-142.

[Chadwick-97]

Chadwick, D., Knight, J. and Clipsham, P. (1997) "Information integrity in end-user systems". In: Jajodia, S., List, W., McGregor, G. and Strous, L. (eds) (1997) *Integrity and internal control in information systems, volume 1: increasing the confidence in information systems*. London: Chapman and Hall, pp. 273-292.

[Chadwick-97a]

Chadwick, D. (1997) "Auditing and the three A's", *CASG Journal*, 7(4), p. 7.

[Chadwick-99]

Chadwick, D., **Rajalingham, K.**, Knight, B. and Edwards, D. (1999) "A methodology for spreadsheet development based on data structure", *CMS Press*, 99/IM/50.

[Chadwick-99a]

Chadwick, D., **Rajalingham, K.**, Knight, B. and Edwards, D. (1999) "An approach to the teaching of spreadsheets using software engineering concepts", *Proceedings of the Fourth International Conference on Software Process Improvement, Research, Education and Training, INSPIRE '99, 9-11 September 1999, Crete, Greece*. Great Britain: British Computer Society, pp. 261-273.

[Chadwick-00]

Chadwick, D., Knight, B. and **Rajalingham, K.** (2000) "Quality control in spreadsheets: a visual approach using color codings to reduce errors in formulae", *Proceedings of the Eighth International Conference on Software Quality Management, SQM2000, 10-13 April 2000, Greenwich, London*. Great Britain: British Computer Society.

[Chadwick-00a]

Chadwick, D., Knight, B. and **Rajalingham, K.** (2000) "Quality control in spreadsheets: a visual approach using color codings to reduce errors in formulae", *Software Quality Journal*, 9(2), pp. 133-143.

[Chadwick-00b]

Chadwick, D. (2000) "Stop the subversive spreadsheet", *Internal Auditing*, May, pp. 26-27.

[Chen-76]

Chen, P. (1976) "The entity relationship model – toward a unified view of data", *ACM Transactions on Data Base Systems*, 1 (1), pp. 6-36.

[Coopers-97]

Coopers & Lybrand (1997) *Website: <http://www.planningobjects.com/jungle1.htm>*. London: Coopers & Lybrand.

[Cragg-92]

Cragg, P.B. and King, M. (1992) *A review and research agenda for spreadsheet based systems in end-user computing*. Working Paper (September), Not formally published.

[Cragg-93]

Cragg, P.G. and King, M. (1993) "Spreadsheet modelling abuse: an opportunity for OR?", *Journal of the Operational Research Society*, 44(8), pp. 743-752.

[Creeth-85]

Creeth, R. (1985) "Microcomputer spreadsheets: their uses and abuses", *Journal of Accountancy*, 159, pp. 90-93.

[Dahl-72]

Dahl, O., Dijkstra, E. and Hoare, C. (1972) *Structured programming*. London: Academic Press.

[Davies-87]

Davies, N. and Ikin, C. (1987) "Auditing spreadsheets", *Australian Accountant*, December, pp. 54-56.

[Davis-96]

Davis, S.J. (1996) "Tools for spreadsheet auditing", *International Journal of Human-Computer Studies*, 45, pp. 429-442.

[Dent-95]

Dent, A. (1995) Dent's personal communication with Prof. Panko, R. (University of Hawaii) via electronic mail (2 April 1995).

[Dhebar-93]

Dhebar, A. (1993) "Managing the quality of quantitative analysis", *Sloan Management Review*, 34, pp. 69-75.

[DiAntonio-86]

DiAntonio, A.E. (1986) *Spreadsheet applications*. Englewood Cliffs, NJ: Prentice-Hall.

[Ditlea-87]

Ditlea, S. (1987) "Spreadsheets can be hazardous to your health", *Personal Computing*, 11, pp. 60-69.

[Floyd-87]

Floyd, B.D. and Pyun, J. (1987) *Errors in spreadsheet use*. Working Paper 167 (October), New York: Center for Research on Information Systems, Information Systems Department, New York University.

[Floyd-95]

Floyd, B.D., Walls, J. and Marr, K. (1995) "Managing spreadsheet model development", *Journal of Systems Management*, 46(1), pp. 38-43.

[Freeman-86]

Freeman, R.M. (1986) "A slip of the chip on computer spreadsheets can cause millions", *The Wall Street Journal*, July, p. 8.

[Freeman-96]

Freeman, D. (1996) "How to make spreadsheets error-proof", *Journal of Accountancy*, May, pp. 75-77.

[Galletta-92]

Galletta, D.F. and Hufnagel, E.M. (1992) "A model of end-user computing policy: context, process, content and compliance", *Information and Management*, 22(1), pp. 1-28.

[Galletta-93]

Galletta, D.F., Abraham, D., El Louadi, M., Lekse, W., Pollailis, Y.A. and Sampler, J.L. (1993) "An empirical study of spreadsheet error performance", *Journal of Accounting, Management, and Information Technology*, 5(3-4), pp. 79-95.

[Galletta-97]

Galletta, D.F., Hartzel, K.S., Johnson, S. and Joseph, J.L. (1997) "Spreadsheet presentation and error detection: an experimental study", *Journal of Management Information Systems*, 13(2), pp. 45-63.

[Hall-96]

Hall, M.J.J. (1996) "A risk and control oriented study of the practices of spreadsheet application developers", *Proceedings of the Twenty-Ninth Hawaii International Conference on Systems Sciences*, 2-6 January 1996, Maui, Hawaii. California: IEEE Computer Society, pp. 364-373.

[Hassinen-88]

Hassinen, K. (1988) *An experimental study of spreadsheet errors made by novice spreadsheet users*. Finland: Department of Computer Science, University of Joensuu.

[Hayen-89]

Hayen, R.L. and Peters, R.M. (1989) "How to ensure spreadsheet integrity", *Management Accounting*, 60(9), pp. 30-33.

[Hendry-94]

Hendry, D.G. and Green, T.R.G. (1994) "Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model", *International Journal of Human-Computer Studies*, 40(6), pp. 1033-1065.

[Hicks-95]

Hicks, L. (1995) *Audit of capital budgeting spreadsheet at NYNEX*. Hicks' personal communication with Prof. Panko, R. (University of Hawaii) via electronic mail (21 June 1995).

[Howitt-85]

Howitt, D. (1985) "Avoiding bottom-line disaster – increased use of electronic worksheets heightens risk of serious errors", *Infoworld*, 7(6), pp. 26-30.

[IEEE-83]

IEEE (1983) *IEEE standard glossary of software engineering terminology*. IEEE Std 729.

[Igarashi-98]

Igarashi, T., Mackinlay, J.D., Chang, B.W. and Zellweger, P.T. (1998) "Fluid visualization of spreadsheet structures", *IEEE Symposium on Visual Languages*, pp. 118-125.

[Ingevaldsson-86]

Ingevaldsson, L. (1986). *JSP - A practical method of program design*. Sweden: Leif Ingevaldsson and Studentlitteratur.

[Ingevaldsson-90]

Ingevaldsson, L. (1990) *Software engineering fundamentals – the Jackson approach*. Sweden: Leif Ingevaldsson and Studentlitteratur.

[Isakowitz-95]

Isakowitz, T., Schocken, S. and Lucas, H.C.J. (1995) “Toward a logical/physical theory of spreadsheet modeling”, *ACM Transactions on Information Systems*, 13(1), pp. 1-37.

[Jackson-75]

Jackson, M.A. (1975) *Principles of program design*. New York: Academic Press.

[Janvrin-96]

Janvrin, D. and Morrison, J. (1996) “Factors influencing risks and outcomes in end-user development”, *Proceedings of the Twenty-Ninth Hawaii International Conference on Systems Sciences, 2-6 January 1996, Maui, Hawaii*. California: IEEE Computer Society.

[Janvrin-00]

Janvrin, D. and Morrison, J. (2000) “Using a structured design approach to reduce risks in end user spreadsheet development”, *Information & Management*, 37(1), pp. 1-12.

[Jones-90]

Jones, G.W. (1990) *Software engineering*. Singapore: John Wiley & Sons.

[Kantaris-94]

Kantaris, N. and Oliver, P.R.M. (1994) *Excel 5 explained*. London: Bernard Babani (publishing).

[Kavanagh-97]

Kavanagh, J. (1997) “Shoddy business models breed financial disaster”, *Computer Weekly*, 19 June 1997.

[Kee-88]

Kee, R. (1988) “Programming standards for spreadsheet software”, *CMA Magazine*, 62(3), pp. 55-60.

[Knight-00]

Knight, B., Chadwick, D. and **Rajalingham, K.** (2000) “A structured methodology for spreadsheet modelling”. In: Chadwick, D. (ed.) (2000) *EuSpRIG 2000 Symposium proceedings - spreadsheet risks, audit and development methods*. London: University of Greenwich, pp. 43-50.

[KPMG-97]

KPMG (1997) *Executive summary: financial model review survey*. London: KPMG Management Consulting.

[KPMG-98]

KPMG (1998) *Review of client spreadsheet models*. London: KPMG Management Consulting.

[KPMG-98a]

KPMG (1998). *KPMG Best practice guide - financial modelling handbook draft 6*. London: KPMG Management Consulting.

[KPMG-98b]

KPMG (1998) *Supporting the decision maker - a guide to the value of business modeling*, Press release (30 July 1998). London: KPMG Management Consulting.

[Lerch-88]

Lerch, F.J. (1988) *Computerized financial planning: discovering cognitive difficulties in knowledge building*, Unpublished doctoral dissertation. Michigan: University of Michigan.

[Mason-89]

Mason, D. and Keane, D. (1989) "Spreadsheets: solution or problem", *New Zealand Computer Interface*, October, pp. 82-84.

[McMickle-89]

McMickle, P. (1989) "Troubleshooting spreadsheets", *Journal of Accounting*, 3(2), pp. 60-71.

[Nardi-90]

Nardi, B.A. and Miller, J. (1990) *The spreadsheet interface: a basis for end user programming*, Technical report HPL-90-08 (March). California: HP Software Technology Laboratory.

[Nardi-91]

Nardi, B.A. and Miller, J. (1991) "Twinkling lights and nested loops: distributed problem solving and spreadsheet development", *International Journal of Man-Machine Studies*, 34(2), pp. 161-184.

[Nixon-01]

Nixon, D. and O'Hara, M. (2001) "Spreadsheet auditing". In: Chadwick, D. and Strous, L. (eds) (2001) *Controlling the subversive spreadsheet – risks, audit and development methods*. The Netherlands: EuSpRIG, pp. 79-93.

[Olsen-87-88]

Olsen, J.R. and Nilsen, E. (1987-1988) "Analysis of the cognition involved in spreadsheet interaction", *Human-Computer Interaction*, 3(4), pp. 309-349.

[Orr-81]

Orr, K.T. (1981) *Structured requirements definition*. Topeka, KS: Ken Orr and Associates.

[Panko-94]

Panko, R.R. and Halverson, R.P., Jr. (1994) "Individual and group spreadsheet design: patterns of errors", *Proceedings of the Twenty-Seventh Hawaii International Conference on Systems Sciences*, 4-7 January 1994, Maui, Hawaii. California: IEEE Computer Society.

[Panko-96]

Panko, R.R. and Halverson, R.P., Jr. (1996) "Spreadsheets on trial: a survey of research on spreadsheet risks", *Proceedings of the Twenty-Ninth Hawaii International Conference on Systems Sciences*, 2-6 January 1996, Maui, Hawaii. California: IEEE Computer Society.

[Panko-96a]

Panko, R.R. and Halverson, R.P., Jr. (1996) "Understanding spreadsheet risks", *Office Systems Research Journal*, 14(2), pp. 1-11.

[Panko-97]

Panko, R.R. and Halverson R.P., Jr. (1997) "Are two heads better than one? (at reducing errors in spreadsheet modeling)", *Office Systems Research Journal*, 15(1), pp. 21-23.

[Panko-98]

Panko, R.R. (1998) "What we know about spreadsheet errors", *Journal of End User Computing*, 10(2), pp. 15-21.

[Panko-99]

Panko, R.R. & Sprague, R.H., Jr. (1999) "Hitting the wall: errors in developing and code-inspecting a 'simple' spreadsheet model", *Decision Support Systems*, 22, pp. 337-353.

[Panko-99a]

Panko, R.R. (1999) "Applying code inspection to spreadsheet testing", *Journal of Management Information Systems*, 16(2), pp. 159-176.

[Parnas-72]

Parnas, D.L. (1972) "On the criteria to be used in decomposing systems into modules", *CACM*, 14(1), pp. 221-227.

[Pearson-88]

Pearson, R. (1988) "Lies, damned lies, and spreadsheets", *Byte*, 13, pp. 299-304.

[Rajalingham-98]

Rajalingham, K. and Chadwick, D. (1998) "Integrity control of spreadsheets: organisation & tools". In: Jajodia, S., List, W., McGregor, G.W. and Strous, L. (eds) (1998) *Integrity and internal control in information systems*. Massachusetts: Kluwer Academic Publishers, pp. 147-168.

[Rajalingham-99]

Rajalingham, K., Chadwick, D., Knight, B. and Edwards, D. (1999) "An approach to improving the quality of spreadsheet models". In: Hawkins, C., King, G., Ross, M. and Staples, G. (eds) (1999) *Software quality management VII – managing quality*. Great Britain: British Computer Society, pp. 117-131.

[Rajalingham-99a]

Rajalingham, K., Chadwick, D., Knight, B. and Edwards, D. (1999) "Efficient methods for checking integrity: an integrated spreadsheet engineering methodology (ISEM)". In: van Biene-Hershey, M.E. and Strous, L. (eds) (1999) *Integrity and internal control in information systems – strategic views on the need for control*. Massachusetts: Kluwer Academic Publishers, pp. 41-58.

[Rajalingham-00]

Rajalingham, K., Chadwick, D., Knight, B. and Edwards, D. (2000) "Quality control in spreadsheets: a software engineering-based approach to spreadsheet development". In: Sprague, R.H., Jr. (ed.) (2000) *Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences 2000 – abstracts and CD-ROM of full papers*. California: IEEE Computer Society.

[Rajalingham-00a]

Rajalingham, K., Chadwick, D. and Knight, B. (2000) "Classification of spreadsheet errors". In: Chadwick, D. (ed.) (2000) *EuSpRIG 2000 Symposium proceedings – spreadsheet risks, audit and development methods*. London: University of Greenwich, pp. 23-34.

[Rajalingham-00b]

Rajalingham, K., Chadwick, D. and Knight, B. (2000) "Classification of spreadsheet errors", *British Computer Society (BCS) Computer Audit Specialist Group (CASG) Journal*, 10(4), pp. 5-10.

[Rajalingham-01]

Rajalingham, K., Chadwick, D. and Knight, B. (2001) "An evaluation of the quality of a structured spreadsheet development methodology". In: Chadwick, D. and Strous, L. (eds) (2001) *Controlling the subversive spreadsheet – risks, audit and development methods*. The Netherlands: EuSpRIG, pp. 39-59.

[Rajalingham-02]

Rajalingham, K., Chadwick, D. and Knight, B. (2002) "Efficient methods for checking integrity: a structured spreadsheet engineering methodology", *Informatica: An International Journal of Computing and Informatics*, 26(1).

[Read-99]

Read, N. and Batson, J. (1999) *Spreadsheet modelling best practice*. London: Business Dynamics, PricewaterhouseCoopers.

[Roberts-88]

Roberts, R.S. (1988) "Problems with user programmed applications: errors in spreadsheets", *Proceedings of the Association of Computer Educators National Conference, June 1988, Dallas, USA*.

[Ronen-89]

Ronen, B., Palley, M.A. and Lucas, H.C., Jr. (1989) "Spreadsheet analysis and design", *Communications of the ACM*, 32, pp. 84-93.

[Rumbaugh-91]

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) *Object-oriented modeling and design*. Englewood Cliffs, NJ: Prentice-Hall.

[Saariluoma-91]

Saariluoma, P. and Sajaniemi, J. (1991) "Extracting implicit tree structures in spreadsheet calculation", *Ergonomics*, 34(8), pp. 1027-1046.

[Saariluoma-94]

Saariluoma, P. and Sajaniemi, J. (1994) "Transforming verbal descriptions into mathematical formulas in spreadsheet calculation", *International Journal of Human-Computer Studies*, 41(6), pp. 915-948.

[Savitz-94]

Savitz, E.J. (1994) "Magellan loses its compass", *Barron's*, 84(50).

[Simkin-87]

Simkin, M.G. (1987) "How to validate spreadsheets", *Journal of Accountancy*, November, pp. 130-138.

[Slater-90]

Slater, R. and Ascroft, P. (1990) *Quantitative techniques in a business context*. London: Chapman and Hall.

[Sommerville-01]

Sommerville, I. (2001) *Software engineering (sixth edition)*. Harlow: Addison-Wesley.

[Speier-96]

Speier, C. and Brown, C.V. (1996) "Perceived risks and management actions: differences in end-user application development across functional groups", *Proceedings of the Twenty-Ninth Hawaii International Conference on Systems Sciences, 2-6 January 1996, Maui, Hawaii*. California: IEEE Computer Society.

[Sprague-82]

Sprague, D.H., Jr. and Carlson, E.D. (1982) *Building effective decision support systems*. Englewood Cliffs, NJ: Prentice-Hall.

[Stang-87]

Stang, D. (1987) "Spreadsheet disasters I have known", *Information Center*, 3(11), pp. 26-30.

[Stevens-74]

Stevens, W., Myers, G. and Constantine, L. (1974) "Structured design", *IBM Systems Journal*, 13(2), pp. 115-139.

[Steward-87]

Steward, D.V. (1987) *Software engineering with systems analysis and design*. California: Brooks/Cole Publishing Company.

[Teo-97]

Teo, T.S.H. and Tan, M. (1997) "Quantitative and qualitative errors in spreadsheet development", *Proceedings of the Thirtieth Hawaii International Conference on Systems Sciences, Maui, Hawaii (Vol 3)*, California: IEEE Computer Society, pp. 149-155.

[van Vliet-96]

Van Vliet, H. (1996) *Software engineering – principles and practice*. Chichester: John Wiley & Sons.

[Ward-97]

Ward, M. (1997) "Fatal addition", *New Scientist*, 16 August 1997.

[Warnier-81]

Warnier, J.D. (1981) *Logical construction of systems*. New York: Van Nostrand Reinhold.

[Weaver-98]

Weaver, P.L., Lambrou, N. and Walkley, M. (1998) *Practical SSADM version 4+ - a complete tutorial guide (second edition)*. London: Financial Times Pitman Publishing.

[Weaver-02]

Weaver, P.L., Lambrou, N. and Walkley, M. (2002) *Practical business systems development using SSADM - a complete tutorial guide (third edition)*. Harlow: Financial Times Prentice Hall.

[Whittaker-99]

Whittaker, D. (1999) "Spreadsheet errors and techniques for finding them", *Management Accounting (UK)*, 77(9), pp. 50-51.

[Wood-96]

Wood, F. (1996) *Business accounting 1 (seventh edition)*. London: Pitman Publishing.

[Woodbury-89]

Woodbury, G.G. (1989) "Re: 'Computer Error' in Durham N.C. election results", *The Risks Digest*, 9(42).

APPENDIX A

FREQUENCY OF SPREADSHEET ERRORS

The information in this table is presented by Panko and Halverson (Panko-96,98,00).

Study	Sample	Study	Cell Error Rate (CER)	% of Models with Errors
Field Audits				
Davies & Ikin (Davies-87)	19 operational spreadsheets	14 had qualitative errors. Methodology unspecified.		21%
Butler (Butler-92)	273 operational spreadsheets audited by 143 United Kingdom tax inspectors in 1992.	Only counted "material" errors. Inspectors used a spreadsheet analysis program designed to identify suspicious parts of a model. Such programs identify only some errors.		10.7%
Cragg & King (Cragg-93)	20 operational spreadsheets from 10 firms.	150 to 10,000 cells. Had been in use for median of 6 months		25%
Hicks (Hicks-95)	1 module with 19 submodules about to enter operation.	Part of a capital budgeting system for NYNEX. Checked heavily ahead of time. Code inspection by three analysts. Found 45 errors in 3,856 cells.	1.2%	100% (1)
Coopers & Lybrand (Coopers-97)	23 spreadsheets from industry	Spreadsheets off by at least 5%		91%
KPMG (KPMG-97)	22 spreadsheets from industry	Spreadsheets containing major errors.		91%
Butler (Butler-00)	7 spreadsheets for tax submissions	Spreadsheets audited with enhanced methodology and software. Single auditor.		86%
Total	367 spreadsheets	Weighted average		24%
Since 1997	54 spreadsheets	Weighted average		91%
Cell Entry Experiments		Errors counted when made, even if corrected later		
Olson & Nilsen (Olsen-87-88)	14 experienced Lotus 1-2-3 users.	Filled in formulas in skeleton model. 4 formula cells per person. 56 total.	21% (2)	
Floyd & Pyun (Floyd-87)		Reanalysed Olson & Nilsen (Olsen, 85,87-88) for errors in text cells.	12.5%	
Lerch (Lerch-88)	21 professionals with at least one year of Lotus 1-2-3 experience.	Filled in formulas in template. CER based on formulas only. CER especially high for formulas referring to cells in both different rows and different columns.	11.3% (2)	

continued ...

Study	Sample	Study	Cell Error Rate (CER)	% of Models with Errors
Development Experiments		Errors counted at end of development process		
Brown & Gould (Brown-87)	9 highly experienced spreadsheet developers	Developed 3 models apiece. All made an error in at least one model. Minimum definition of errors, excluding the omission of requirements.		44%
Brown & Gould (Brown-87)		Broader definition of errors including the omission of requirements.		63%
Hassinen (Hassinen-88)	92 novice spreadsheet students developed 355 spreadsheets	Paper and pencil exercise. Subjects filled in formulas in a skeleton model containing organization and numbers.	4.3% (2)	55%
Hassinen (Hassinen-88)	10 novice students developing 48 spreadsheets	Computer exercise for same task.		48%
Janvrin & Morrison (Janvrin-96,00)	61 upper division business and graduate accounting students	Study 1: Developed model with multiple worksheets. Had template with filled-in values. Measured incorrect links between worksheets. Model had 51 links. Students had 16 days to do task. Worked an average of 8.8 hours	7%-14% (3)	84%-95%
Janvrin & Morrison (Janvrin-96,00)	88 senior-level accounting students	Study 2: Developed model with multiple worksheets. 66 links between worksheets. No templates to work from; only 1 check figure. CER is percent of incorrect links between spreadsheets.	8%-17% (3)	
Panko & Halverson (Panko-97)	35 undergraduate business students working alone	Developed pro forma income statement based on the Galumpke task.	5.4%	80%
Panko & Halverson (Panko-97)	40 undergraduate business students working in groups of 4	Developed pro forma income statement based on the Galumpke task	2.0%	60%
Panko & Sprague (Panko-99)	102 undergraduate business students and 50 MBA students. 17 MBAs had more than 250 hours of experience	Developed a model based on the Wall task designed to be relatively simple and free of domain knowledge. No difference in errors across groups.	2.0%	35%
Teo & Tan (Teo-97)	168 undergraduate business students taking second-year IS course	Developed a model based on the Wall task, then did a what-if analysis	2.0%	42%
(Unpublished)	80 undergraduate business students	Developed spreadsheet working alone or in triad (group of 3). Error rates for individual, triad.	4.6%, 1.0%	86%, 27%

continued ...

Study	Sample	Study	Cell Error Rate (CER)	% of Models with Errors
Code Inspection Experiments		Subjects looked for errors in a model		
Galletta et al (Galletta-93)	30 MBA students and 30 CPA accountants	Examined 6 small model with 1 seeded error in each. Subjects with 250 hours or more of spreadsheet experience did not find more errors than those without experience.	34%-54% (4)	
Galletta et al (Galletta-97)	113 MBA students	Finding eight seeded errors in a student budgeting model.	45%-55% (4)	
Panko & Sprague (Panko-99)	23 undergraduate subjects with errors in initial model.	Study described above. Code inspected own models. Fixed 18% of errors. Only 13% of spreadsheets were fixed completely. One was made worse.	81% (4)	
Panko (Panko-99a)	33 undergraduate MIS majors	Code inspected 11 spreadsheet models, individually and then in groups. Missed errors for individuals, groups.	40%, 17% (4)	

APPENDIX B

SURVEYS ABOUT CORPORATE CONTROL POLICIES

The information in this table is presented by Panko and Halverson (Panko-96).

Study	Method/Sample	Selected Findings
Cale (Cale-94)	52 IS & non-IS managers in 25 firms	Less than 1/10 of companies had written policies on testing spreadsheets; about ¼ had unwritten polices; about 6/10 had no policies. Documentation standards similar. About 70% strongly agreed that lack of testing standards produced serious problems. None disagreed. Reluctant to impose standards if developed by person for own use. More likely as used by others, if updates database, or as size grows. 90% would require testing for development lasting one week; 100% if one month.
Cragg & King (Cragg-93)	FTF interviews with 17, questionnaire survey of 14, N=31.	1/10 said firm had formal policies. 9/10 said no one in firm responsible for SS development.
Floyd, Walls & Marr (Floyd-95)	72 end users in 4 corporations	1/7 had development policies, 2/5 implementation policies; 2/3 development policies. 1/3 required approvals only for important models. Almost all: any policy existing initiated by workgroup. All functions had some policies; modification policies most common. None reported comprehensive standards for all models. None knew of disasters in their firms. Clan-based control policy: socialization.
Galletta & Hufnagel (Galletta-92)	107 MIS executives in mail survey	End user computing, not just spreadsheeting. Restrictions on application development? 23% rule, 58% guideline, 28% don't address; compliance level if address: 27% full compliance, 58% partial compliance; 15% ignore. Post-development audit requirement? 15% rule, 34% guideline, 52% don't address; compliance level if address: 10% full compliance, 49% partial compliance; 41% ignore.
Hall (Hall-96)		Only 11% new of a comprehensive corporate policy; only 1/3 of these could be located it in written form.

continued ...

Study	Method/Sample	Selected Findings
Hendry & Green (Hendry-94)	Ethnographic interviews with 11 SS developers	Modeled after Nardi & Miller (Nardi-91) but added a part in which the interviewer went over a specific SS with the developer. Generally repeated Nardi & Miller, but noted pattern of difficulty in comprehending parts of spreadsheets. Describing efforts to build error-free models by taking specific actions.
Nardi & Miller (Nardi-91)	Ethnographic interviews with 11 SS developers	Extensive joint development mixed programming & domain expertise; sometimes other built difficult parts; other times other checked for reasonableness, gave guidance. Considerable evidence of taking care in development; conscious of errors; spend considerable time debugging. Reasonableness, cross-footings, spot-checking of values, examining formulas. Gave one example of comprehensive code inspection—the subject was taking over a SS developed by another.
Speier & Brown (Speier-96)	Study of 3 departments	Study of overall EUC, not just SS. Interviewed managers of 3 departments in a firm: financial operations, marketing and sales. Questionnaire interviews of 22 end users. Company has few corporate rules beyond backup, which is not enforced. Managers differed in concerns. End users differed by department in awareness of norms and perceptions of benefits. Mostly unwritten norms. Underscores importance of department perspective.

APPENDIX C

EXAMPLES OF THE APPLICATION OF THE PROPOSED METHODOLOGY

Example 1: *Trading and Profit and Loss Account for a Particular Year*

STAGE 1: Requirements Analysis and Development of Output Structures

Based on the example of a *Trading and Profit and Loss Model*, the model developer would first examine the desired output(s). A typical desired output is shown in *Figure C1*. Each column and row is labelled with either an alphabet (column) or a number (row). A cell address is made up of a column label and a row number. Based on *Figure C1*, **Gross Profit** occupies cell **B11** while the value corresponding to **Appropriations** is in cell **C5**.

	B	C
03	Net profit	?
04	Unappropriated profits from last year	?
05	Appropriations	?
06	Unappropriated profits carried to next year	?
07		
08		
09		
10		
11	Gross Profit	?
12	Expenses	?
13	Net profit	?

Figure C1: Desired Output

Based on *Figure C1*, among the required output values, those that take the form of a formula are identified. They are *net profit*, *appropriations*, *unappropriated profits carried to next year*, *gross profit* and *expenses*.

The underlying logic and structure of each formula is determined at the logical level:

- Net profit = Gross profit - Total expenses
- Total appropriations = $\sum (appropriations)$
- Unappropriated profits carried to next year = Net profit +
Unappropriated profits from last year - Total appropriations
- Gross profit = Sales - Cost of goods sold + Closing stock
- Total expenses = $\sum (expenses)$

STAGE 2:
Conceptual Design of the Model Schema

Based on the desired outputs of the *Trading and Profit and Loss Model*, shown in *Figure C1*, the model developer would firstly determine the operands of each output formula. This step was performed in the previous stage and therefore does not have to be repeated. The precedent-dependant relationships between the output formulae are then established. As described in *Chapter 5 (Section 5.2)*, if **B** is a precedent of **A**, this is represented by an arrow pointing from **B** to **A**, i.e. $B \rightarrow A$, or $A \leftarrow B$. The dependencies between the formulae are shown in *Figure C2*.

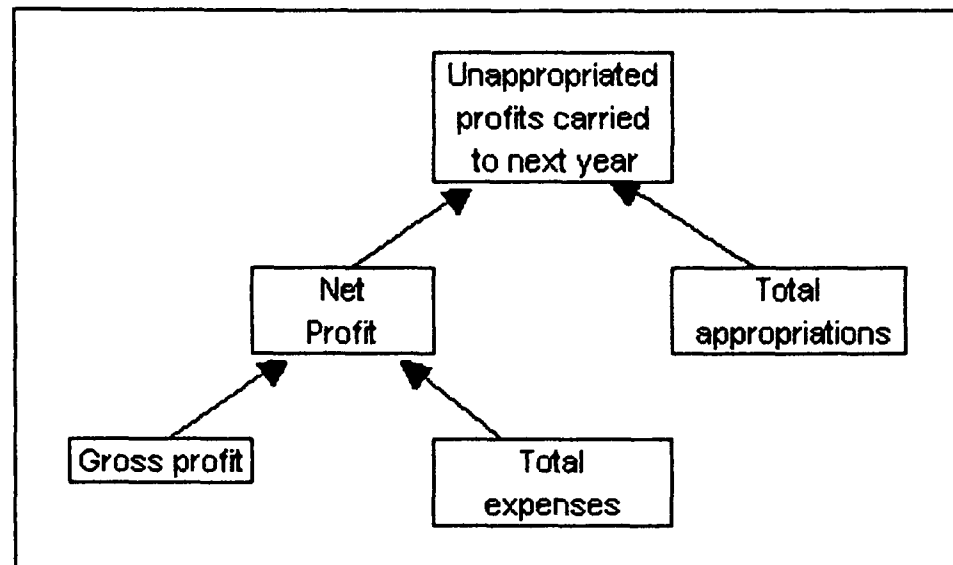


Figure C2: Dependencies Between Formulae

From *Figure C2*, the only root formula is easily identifiable, i.e. *Unappropriated profits carried to next year*. This is because it has no dependants.

The Jackson structure representation of the direct and indirect precedents of the root formula is shown in *Figure C3*. This diagram also represents the conceptual design of the spreadsheet model.

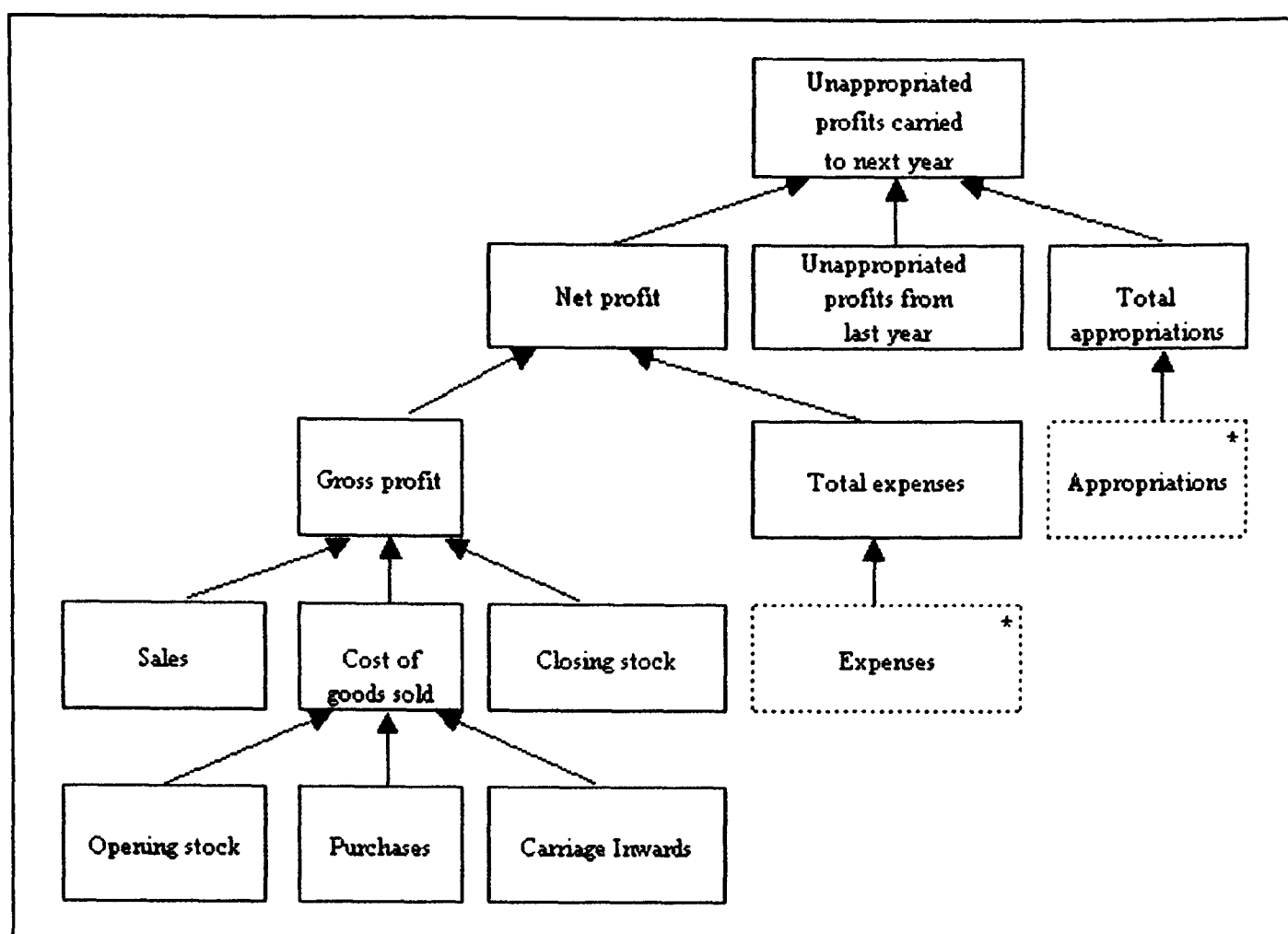


Figure C3: Conceptual Design of the *Model Schema*

This model distinctly shows the precedents of the various functions. The leaves *expenses* and *appropriations* are represented as iterations in *Figure C3*. This is because each of them refers to a group of related inputs, defined as a range. The elements of a range are always operated on or manipulated as a set rather than individually. These iterations also indicate that the contents of the defined set (or input range) can change frequently as a result of data entry operations. It is also assumed that *unappropriated profits from last year* is an input.

STAGE 3:

Logical Design of the Model Schema

Observing the conceptual design of the model, shown in *Figure C3*, it is found that there are no nodes or formulae with multiple dependants. As a result, there are no graph sub-structures within the model that need to be resolved. Therefore, in this particular example, the logical design of the model schema is said to be identical to the conceptual design.

STAGE 4:

Physical Construction of the Model Schema Layout on the Spreadsheet

Applying the steps and techniques associated with this stage, the logical design of the *Trading and Profit and Loss* model shown in *Figure C3* can be mapped onto a physical spreadsheet structure as shown in *Figure C4*. *Indentation* is used to show the different levels within the model. An asterisk (*) is placed next to a row label (in column A) to denote that the formula operates on an *input range*.

	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year					
07	Net profit					
08	Gross Profit					
09	Sales					
10	Cost of goods sold					
11	Opening stock					
12	Purchases					
13	Carriage inwards					
14	Closing stock					
15	Total expenses *					
16	Unappropriated profits from last year					
17	Total appropriations *					

Figure C4: Outline of *Model Schema*

STAGE 5:

Development of the Input Component and Entry of Model Inputs

The input component for the *Trading and Profit and Loss Model* can now be created and all inputs entered in order to provide the model schema with the values required. This is done on a separate worksheet. The worksheet should be labelled *input*. Based on the logical design for the spreadsheet model, the end-leaves can be implemented within an *input* component. This is shown in *Figure C5*.

	B	C	D
05		Sales	135,486.00
06		Opening stock	40,360.00
07		Closing stock	52,360.00
08		Purchases	72,360.00
09		Carriage inwards	1,570.00
10		Unappropriated profits from last year	15,286.00
11		Expenses	
12		Salaries	18,310.00
13		Rates and occupancy	4,515.00
14		Carriage outwards	1,390.00
15		Office expenses	3,212.00
16		Sundry expenses	1,896.00
17		Depreciation: Buildings	5,000.00
18		Depreciation: Equipment	9,000.00
19		Directors' remuneration	9,500.00
20		Appropriations	
21		Proposed dividend	10,000.00
22		General reserve	1,000.00
23		Foreign exchange	800.00

Figure C5: Input Component

The elements of a group of inputs, which are always operated on or manipulated as a set, are indented and a common name assigned to it. In order to define such a set of related inputs, the input values corresponding to the set elements are defined as a range, which is given a distinct name in the model.

The input data corresponding to the input sets *expenses* (D12 to D19) and *appropriations* (D21 to D23) are defined as ranges, and assigned the range names *expenses* and *appropriations* respectively.

STAGE 6:
Implementation of Formulae and Binding Relationships in the Model Schema

The formulae and binding relationships of the model can now be physically implemented or programmed in the model schema. The structure of the model schema has already been produced and the appropriate formulae will be entered into the right cells.

References to inputs are first entered into the relevant cells in the model schema. This includes functions of input ranges, such as *total expenses* and *total appropriations*.

A bottom-up approach is taken in the implementation of formulae and relationships in the model schema. *Figures C6 (a) and C6 (b)* show the final state of the model schema of the *Trading and Profit and Loss Model*. In *Figure C6 (a)*, the structure of the underlying functions are shown as entered by the model developer. *Figure C6 (b)*, on the other hand, shows the surface values of the formulae based on the current state of inputs shown in *Figure C5*.

	B	C	D	E	F	G
		£	£	£	£	£
05						
06	Unappropriated profits carried to next year	= (D7 + D16) - D17				
07	Net profit		= E8 - E15			
08	Gross Profit			= F9 - F10 + F14		
09	Sales				= Input!D5	
10	Cost of goods sold				= G11 + G12 + G13	
11	Opening stock					= Input!D6
12	Purchases					= Input!D8
13	Carriage inwards					= Input!D9
14	Closing stock				= Input!D7	
15	Total expenses *			= SUM(Expenses)		
16	Unappropriated profits from last year		= Input!D10			
17	Total appropriations *		= SUM(Appropriations)			

Figure C6 (a): Model Schema

	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	24,219				
07	Net profit		20,733			
08	Gross Profit			73,556		
09	Sales				135,486	
10	Cost of goods sold				114,290	
11	Opening stock					40,360
12	Purchases					72,360
13	Carriage inwards					1,570
14	Closing stock				52,360	
15	Total expenses *			52,823		
16	Unappropriated profits from last year		15,286			
17	Total appropriations *		11,800			

Figure C6 (b): Model Schema

Each data value and formula in the input component and model schema should be assigned a unique name. These names should then be used as operands within formulae, instead of cell addresses. The exception to this rule applies to a data value which is part of a related set of data that is always treated and operated on as a set, in which case it will be defined as a range along with the other related inputs. If this technique is applied, the formula view of the model schema will appear as displayed in *Figure C7*.

	B	C	D	E	F	G
05		£	£	£	£	£
06	Unappropriated profits carried to next year	=NetProfit+UnappropriatedProfitsFromLastYear-TotalAppropriations				
07	Net profit	=GrossProfit-TotalExpenses				
08	Gross Profit		=Sales-CostOfGoodsSold+ClosingStock			
09	Sales			=Sales		
10	Cost of goods sold			=SUM(OpeningStock;CarriageInwards)		
11	Opening stock				=OpeningStock	
12	Purchases				=Purchases	
13	Carriage inwards				=CarriageInwards	
14	Closing stock			=ClosingStock		
15	Total expenses *		=SUM(Expenses)			
16	Unappropriated profits from last year	=UnappropriatedProfitsFromLastYear				
17	Total appropriations *	=SUM(Appropriations)				

Figure C7: Model Schema

STAGE 7:

Implementation of References in the Output Component

The final state of the output component is shown in *Figures C8 (a)*, *C8 (b)* and *C8 (c)*. *Figure C8 (a)* displays the surface values of formulae (references) in the output component, while *Figures C8 (b)* and *C8 (c)* show the underlying structure of the formulae (references) as they are entered. In *Figure C8 (c)*, the formulae or references are more comprehensible as they are in natural language. This can be done if every formula in the *model schema* and every piece of data in the *input component* is assigned a unique name.

	B	C
03	Net profit	20,733
04	Unappropriated profits from last year	15,286
05	Appropriations	11,800
06	Unappropriated profits carried to next year	24,219
07		
08		
09		
10		
11	Gross Profit	73,556
12	Expenses	52,823
13	Net profit	20,733

Figure C8 (a): Output Component

	B	C
03	Net profit	=Schema!D7
04	Unappropriated profits from last year	=Schema!D16
05	Appropriations	=Schema!D17
06	Unappropriated profits carried to next year	=Schema!C6
07		
08		
09		
10		
11	Gross Profit	=Schema!E8
12	Expenses	=Schema!E15
13	Net profit	=Schema!D7

Figure C8 (b): Output Component

	B	C
03	Net profit	=NetProfit
04	Unappropriated profits from last year	=UnappropriatedProfitsFromLastYear
05	Appropriations	=TotalAppropriations
06	Unappropriated profits carried to next year	=UnappropriatedProfitsCarriedToNextYear
07		
08		
09		
10		
11	Gross Profit	=GrossProfit
12	Expenses	=TotalExpenses
13	Net profit	=NetProfit

Figure C8 (c): Output Component

STAGE 8:

Testing, Documentation and Administration of the Spreadsheet Model

Refer to *Section 7.3: Stage 8*.

Example 2: *Post-tax Income Distribution Model*

STAGE 1:

Requirements Analysis and Development of Output Structures

The desired output of the model is shown in *Figure C9 (a)*. Its representation in spreadsheet form is shown in *Figure C9 (b)*.

Post-tax income distribution for 1975 and 1985						
Income after tax 1975	Number (thousands)	Total income	Number		Income	
			%	Cumulative %	%	Cumulative %
675 but under 750	750	357	255			
750 but under 1000	1000	1350	1190			
1000 but under 1250	1250	1780	2000			
1250 but under 1500	1500	1840	2530			
1500 but under 1750	1750	1850	3000			
Income after tax 1985						
1750 but under 2000	2000	635	1190			
2000 but under 2500	2500	1470	3290			
2500 but under 3000	3000	1410	3850			
3000 but under 3500	3500	1670	5420			
3500 but under 4000	4000	1670	6250			

Figure C9 (a): Output Structure

	B	C	D	E	F	G	H
02	Post-tax income distribution for 1975 and 1985						
03							
04	Income after tax	Number		Number		Income	
05	1975	(thousands)	Total income	%	Cumulative %	%	Cumulative %
06							
07	675 but under 750	357	255	?	?	?	?
08	750 but under 1000	1350	1190	?	?	?	?
09	1000 but under 1250	1780	2000	?	?	?	?
10	1250 but under 1500	1840	2530	?	?	?	?
11	1500 but under 1750	1850	3000	?	?	?	?
12							
13	Income after tax						
14	1985						
15							
16	1750 but under 2000	635	1190	?	?	?	?
17	2000 but under 2500	1470	3290	?	?	?	?
18	2500 but under 3000	1410	3850	?	?	?	?
19	3000 but under 3500	1670	5420	?	?	?	?
20	3500 but under 4000	1670	6250	?	?	?	?
21							

Figure C9 (b): Output Structure on Spreadsheet

Based on *Figures C9 (a) and C9 (b)*, the required formulae that can be identified are *number %*, *number cumulative %*, *income %* and *income cumulative %* for each range of each year.

The logical mathematical equations for the formulae are as follows:

- {Number %} = Range number as a percentage of Total number for the year
- {Number cumulative %} = Number cumulative % of previous range + number %
- {Income %} = Total range income as a percentage of Total year income
- {Income cumulative %} = Income cumulative % of previous range + Income %

A formula within curly brackets represents multiple iterations/instances.

STAGE 2:
Conceptual Design of the Model Schema

The desired output of the model is shown in *Figure C9 (a)*. Its representation in spreadsheet form is shown in *Figure C9 (b)*.

Based on the desired outputs of the *Post-tax Income Distribution Model*, the operands of each output formula have already been determined. The precedent-dependant relationships between the output formulae are now established. This is shown in *Figure C10*.

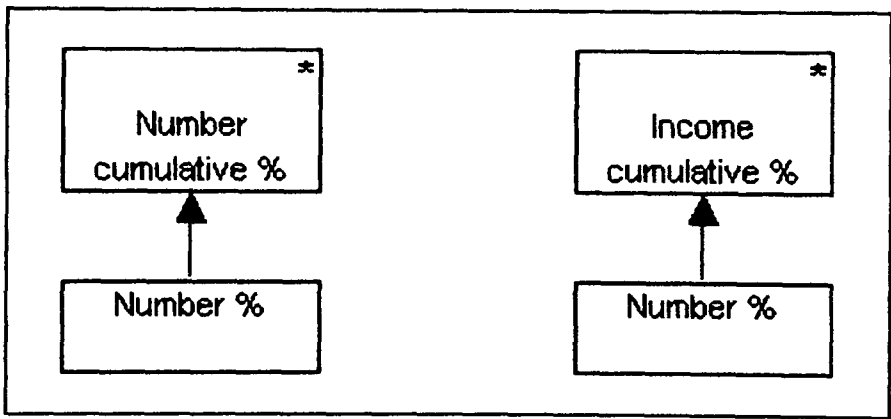


Figure C10: Precedent-Dependant Relationships Between the Output Formulae

From *Figure C10*, it can be observed that there are two sets of root formulae, namely, an *iteration of number cumulative %'s* and an *iteration of income cumulative %'s*. The next step is to use a Jackson structure to represent the direct and indirect precedents of each set of root formulae.

As each root formula is an iterated component, the iteration group(s) of each root formula has to be identified. According to the logic of the model, the *number cumulative %* and *income cumulative %* are to be calculated for each range of each year. This is shown in *Figure C11*.

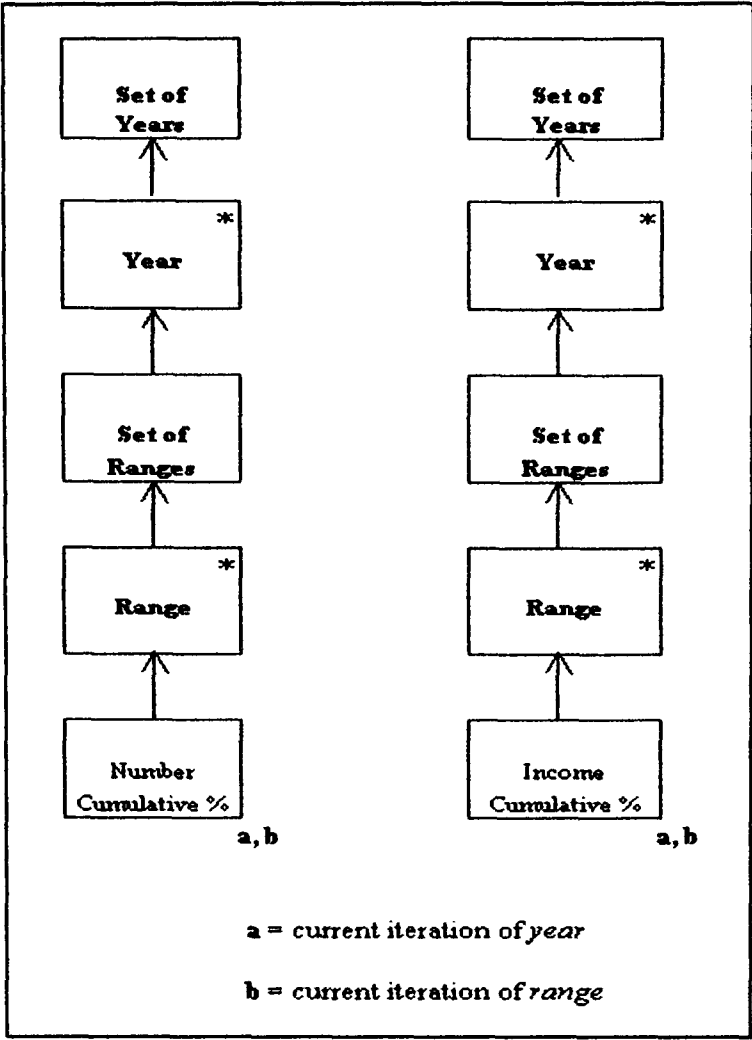


Figure C11: Relationship Between *Iterations*

Each iterated component is associated with an index that is used to indicate which iteration each of its precedents belongs to. The diagram in *Figure C11*, constructed using Jackson notation, represents the conceptual design of the spreadsheet model. The Jackson structures of both sets of root formulae have been merged into a single structure representing the entire spreadsheet model.

Based on *Figure C12*, by adopting a top-down approach without duplicating model elements, a graph-like structure, as opposed to a tree structure, is produced. The model distinctly shows the direct and indirect precedents of each formula (represented by a *node* that is not an *end-leaf*).

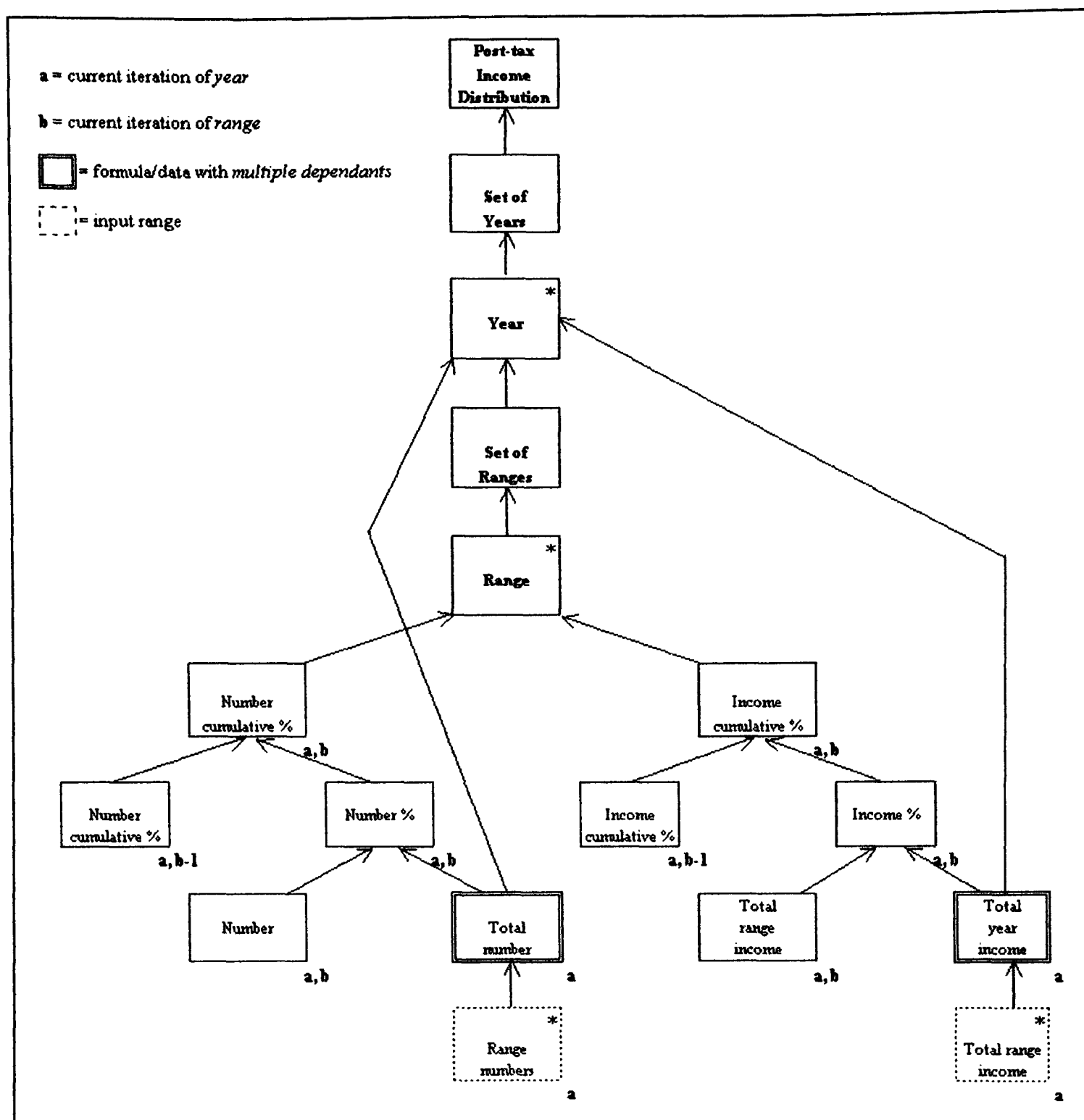


Figure C12: Conceptual Design of the *Post-tax Income Distribution Model*

STAGE 3:

Logical Design of the Model Schema

In *Example 1*, the conceptual design took the form of a tree. Not all spreadsheet conceptual models are of this simple form, but have underlying structures in the form of a more general graph.

Figure C12 shows the conceptual design of the *Post-tax Income Distribution Model*. There are two nodes with multiple dependants, namely, *total number* and *total year income*. Applying *Steps 1* and *2*, the conceptual design is transformed into a logical design of a pure tree structure by resolving the irregularities. The logical design of the model schema is presented in *Figure C13*.

Looking at *Figure C13*, the duplicated nodes *total number* and *total year income* are defined as separate modules.

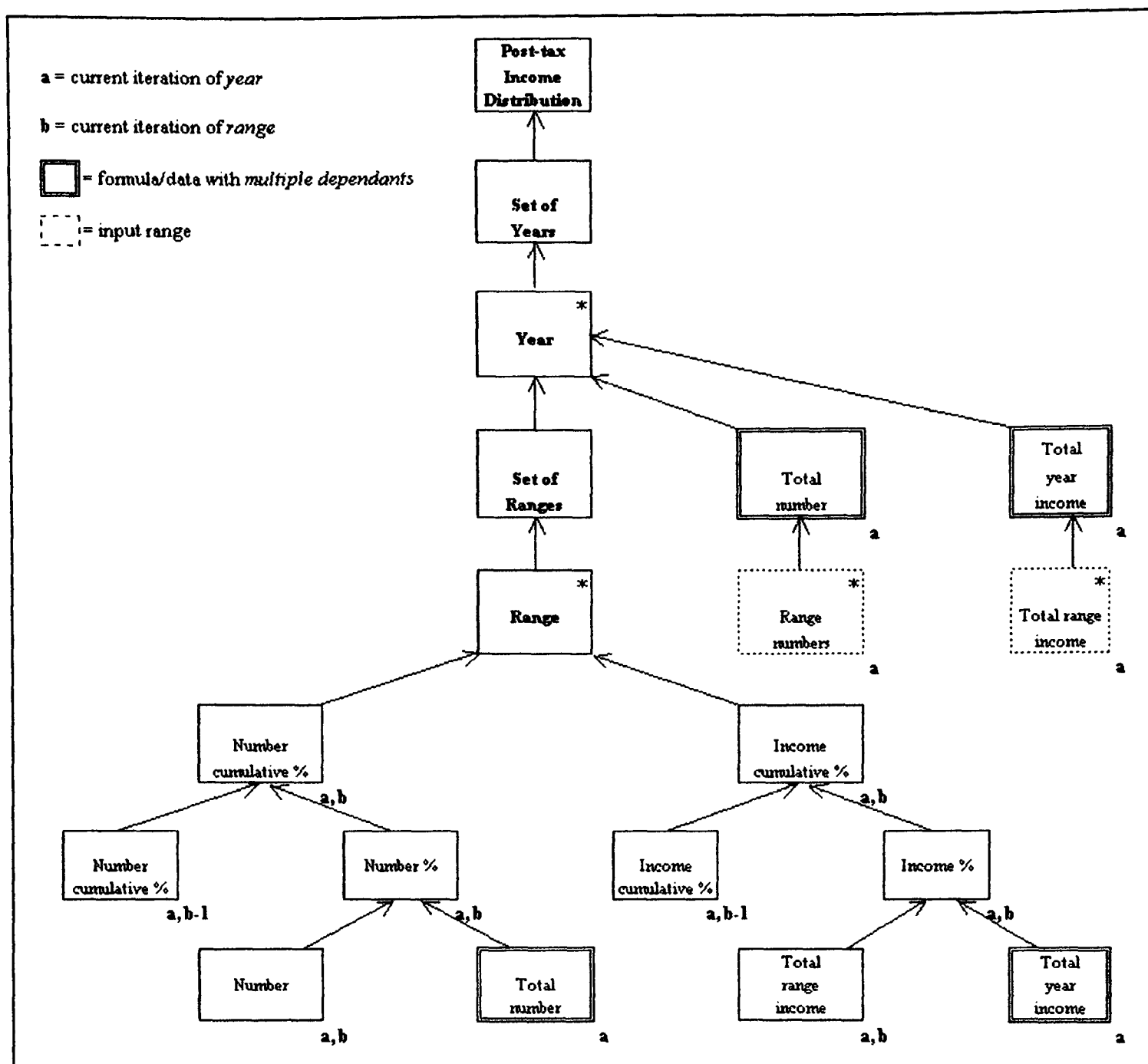


Figure C13: Logical Design of the Model Schema

STAGE 4:

Physical Construction of the Model Schema Layout on the Spreadsheet

The structure or layout of the *model schema* can now be created by mapping the logical model onto the physical spreadsheet according to the rules and steps of this stage. The first column contains module headings and names of iterated components. These are appropriately indented based on the logical design. The second column contains the names of formulae and data used in the spreadsheet model schema, also systematically indented. The resulting structure of the model schema is illustrated in *Figures C14 (a), C14 (b) and C14 (c)*.

Post-tax income distribution for 1975 & 1985					
	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008	Number Cumulative %				
009	Previous Number Cumulative %				
010	Number %				
011	Number (000)				
012	Total Number (000)				
013	Income Cumulative %				
014	Previous Income Cumulative %				
015	Income %				
016	Total Range Income				
017	Total Year Income				
018	750 but under 1000				
019	Number Cumulative %				
020	Previous Number Cumulative %				
021	Number %				
022	Number (000)				
023	Total Number (000)				
024	Income Cumulative %				
025	Previous Income Cumulative %				
026	Income %				
027	Total Range Income				
028	Total Year Income				
029	1000 but under 1250				
030	Number Cumulative %				
031	Previous Number Cumulative %				
032	Number %				
033	Number (000)				
034	Total Number (000)				
035	Income Cumulative %				
036	Previous Income Cumulative %				
037	Income %				
038	Total Range Income				
039	Total Year Income				
040	1250 but under 1500				
041	Number Cumulative %				
042	Previous Number Cumulative %				
043	Number %				
044	Number (000)				
045	Total Number (000)				
046	Income Cumulative %				
047	Previous Income Cumulative %				
048	Income %				
049	Total Range Income				
050	Total Year Income				
051	1500 but under 1750				
052	Number Cumulative %				
053	Previous Number Cumulative %				
054	Number %				
055	Number (000)				
056	Total Number (000)				
057	Income Cumulative %				
058	Previous Income Cumulative %				
059	Income %				
060	Total Range Income				
061	Total Year Income				
062	Total Number (000)				
063	Total Number (000) *				
064	Total Year Income				
065	Total Year Income *				
066	1985				
067	Range				
068	1750 but under 2000				
069	Number Cumulative %				
070	Previous Number Cumulative %				
071	Number %				
072	Number (000)				
073	Total Number (000)				
074	Income Cumulative %				
075	Previous Income Cumulative %				
076	Income %				
077	Total Range Income				
078	Total Year Income				
079	2000 but under 2500				
080	Number Cumulative %				
081	Previous Number Cumulative %				
082	Number %				
083	Number (000)				
084	Total Number (000)				
085	Income Cumulative %				
086	Previous Income Cumulative %				
087	Income %				
088	Total Range Income				
089	Total Year Income				
090	2500 but under 3000				
091	Number Cumulative %				
092	Previous Number Cumulative %				
093	Number %				
094	Number (000)				
095	Total Number (000)				
096	Income Cumulative %				
097	Previous Income Cumulative %				
098	Income %				
099	Total Range Income				
100	Total Year Income				
101	3000 but under 3500				
102	Number Cumulative %				
103	Previous Number Cumulative %				
104	Number %				
105	Number (000)				
106	Total Number (000)				
107	Income Cumulative %				
108	Previous Income Cumulative %				
109	Income %				
110	Total Range Income				
111	Total Year Income				
112	3500 but under 4000				
113	Number Cumulative %				
114	Previous Number Cumulative %				
115	Number %				
116	Number (000)				
117	Total Number (000)				
118	Income Cumulative %				
119	Previous Income Cumulative %				
120	Income %				
121	Total Range Income				
122	Total Year Income				
123	Total Number (000)				
124	Total Number (000) *				
125	Total Year Income				
126	Total Year Income *				

Figure C14 (a): The Entire Structure of the Model Schema

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008		Number Cumulative %			
009		<i>Previous Number Cumulative %</i>			
010		Number %			
011		Number (000)			
012		<i>Total Number (000)</i>			
013		Income Cumulative %			
014		<i>Previous Income Cumulative %</i>			
015		Income %			
016		Total Range Income			
017		<i>Total Year Income</i>			
018	750 but under 1000				
019		Number Cumulative %			
020		<i>Previous Number Cumulative %</i>			
021		Number %			
022		Number (000)			
023		<i>Total Number (000)</i>			
024		Income Cumulative %			
025		<i>Previous Income Cumulative %</i>			
026		Income %			
027		Total Range Income			
028		<i>Total Year Income</i>			

Figure C14 (b): A Segment of the Model Schema

112	3500 but under 4000			
113	Number Cumulative %			
114	<i>Previous Number Cumulative %</i>			
115	Number %			
116	Number (000)			
117	<i>Total Number (000)</i>			
118	Income Cumulative %			
119	<i>Previous Income Cumulative %</i>			
120	Income %			
121	Total Range Income			
122	<i>Total Year Income</i>			
123	Total Number (000)			
124	Total Number (000) *			
125	Total Year Income			
126	Total Year Income *			

Figure C14 (c): A Segment of the Model Schema

STAGE 5:
Development of the Input Component and Entry of Model Inputs

Based on the steps and guidelines discussed under this stage of the proposed methodology, the input component of the *Post-tax Income Distribution* model is built and all model inputs are entered. The input component or section can be structured either as seen in *Figure C15 (a)* or *Figure C15 (b)*.

	B	C	D	E
04	Year		Number (000)	Total Income
05	1975			
06	Range			
07		675 but under 750	357	255
08		750 but under 1000	1350	1190
09		1000 but under 1250	1780	2000
10		1250 but under 1500	1840	2530
11		1500 but under 1750	1850	3000
12	1985			
13	Range			
14		1750 but under 2000	635	1190
15		2000 but under 2500	1470	3290
16		2500 but under 3000	1410	3850
17		3000 but under 3500	1670	5420
18		3500 but under 4000	1670	6250

Figure C15 (a): Input Component

Number (000)		
Year		
1975		
Range		
	675 but under 750	357
	750 but under 1000	1350
	1000 but under 1250	1780
	1250 but under 1500	1840
	1500 but under 1750	1850
1985		
Range		
	1750 but under 2000	635
	2000 but under 2500	1470
	2500 but under 3000	1410
	3000 but under 3500	1670
	3500 but under 4000	1670
Total Income		
Year		
1975		
Range		
	675 but under 750	255
	750 but under 1000	1190
	1000 but under 1250	2000
	1250 but under 1500	2530
	1500 but under 1750	3000
1985		
Range		
	1750 but under 2000	1190
	2000 but under 2500	3290
	2500 but under 3000	3850
	3000 but under 3500	5420
	3500 but under 4000	6250

Figure C15 (b): Input Component

STAGE 6:
Implementation of Formulae and Binding Relationships in the Model Schema

The formulae and binding relationships of the *model schema* of the *Post-tax Income Distribution* model are implemented. *Figures C16 (a)* and *C16 (b)* show the final state of the model schema of the *Post-tax Income Distribution* model. In *Figure C16 (a)*, the structure of the underlying formulae is shown, while *Figure C16 (b)*, on the other hand, shows the surface values of the formulae based on a segment of the model. References to modules within this model segment are displayed in *Figure C16 (c)*.

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008	Number Cumulative %		4.97		
009	Previous Number Cumulative %			0.00	
010	Number %			4.97	
011	Number (000)				357
012	Total Number (000)				7177
013	Income Cumulative %		2.84		
014	Previous Income Cumulative %			0.00	
015	Income %			2.84	
016	Total Range Income				255
017	Total Year Income				8975
018	750 but under 1000				
019	Number Cumulative %		23.78		
020	Previous Number Cumulative %			4.97	
021	Number %			18.81	
022	Number (000)				1350
023	Total Number (000)				7177
024	Income Cumulative %		16.10		
025	Previous Income Cumulative %			2.84	
026	Income %			13.26	
027	Total Range Income				1190
028	Total Year Income				8975
029	1000 but under 1250				
030	Number Cumulative %		48.59		
031	Previous Number Cumulative %			23.78	
032	Number %			24.80	
033	Number (000)				1780
034	Total Number (000)				7177
035	Income Cumulative %		36.38		
036	Previous Income Cumulative %			16.10	
037	Income %			22.28	
038	Total Range Income				2000
039	Total Year Income				8975
040	1250 but under 1500				
041	Number Cumulative %		74.22		
042	Previous Number Cumulative %			48.59	
043	Number %			25.64	
044	Number (000)				1840
045	Total Number (000)				7177
046	Income Cumulative %		66.57		
047	Previous Income Cumulative %			36.38	
048	Income %			29.19	
049	Total Range Income				2530
050	Total Year Income				8975
051	1500 but under 1750				
052	Number Cumulative %		100.00		
053	Previous Number Cumulative %			74.22	
054	Number %			25.78	
055	Number (000)				1850
056	Total Number (000)				7177
057	Income Cumulative %		100.00		
058	Previous Income Cumulative %			66.57	
059	Income %			33.43	
060	Total Range Income				3000
061	Total Year Income				8975
062	Total Number (000)				
063	Total Number (000) *		7177		
064	Total Year Income				
065	Total Year Income *		8975		
066	1985				
067	Range				
068	1750 but under 2000				
069	Number Cumulative %		9.26		
070	Previous Number Cumulative %			0.00	
071	Number %			9.26	
072	Number (000)				636
073	Total Number (000)				6855
074	Income Cumulative %		5.95		
075	Previous Income Cumulative %			0.00	
076	Income %			5.95	
077	Total Range Income				1190
078	Total Year Income				20000
079	2000 but under 2500				
080	Number Cumulative %		30.71		
081	Previous Number Cumulative %			9.26	
082	Number %			21.44	
083	Number (000)				1470
084	Total Number (000)				6855
085	Income Cumulative %		22.40		
086	Previous Income Cumulative %			5.95	
087	Income %			16.45	
088	Total Range Income				3290
089	Total Year Income				20000
090	2500 but under 3000				
091	Number Cumulative %		51.28		
092	Previous Number Cumulative %			30.71	
093	Number %			20.57	
094	Number (000)				1410
095	Total Number (000)				6855
096	Income Cumulative %		41.65		
097	Previous Income Cumulative %			22.40	
098	Income %			19.25	
099	Total Range Income				3850
100	Total Year Income				20000
101	3000 but under 3500				
102	Number Cumulative %		75.64		
103	Previous Number Cumulative %			51.28	
104	Number %			24.36	
105	Number (000)				1670
106	Total Number (000)				6855
107	Income Cumulative %		68.75		
108	Previous Income Cumulative %			41.65	
109	Income %			27.10	
110	Total Range Income				5420
111	Total Year Income				20000
112	3500 but under 4000				
113	Number Cumulative %		100.00		
114	Previous Number Cumulative %			75.64	
115	Number %			24.36	
116	Number (000)				1670
117	Total Number (000)				6855
118	Income Cumulative %		100.00		
119	Previous Income Cumulative %			68.75	
120	Income %			31.25	
121	Total Range Income				6250
122	Total Year Income				20000
123	Total Number (000)				
124	Total Number (000) *		6855		
125	Total Year Income				
126	Total Year Income *		20000		

Figure C16 (a): Model Schema

101	3000 but under 3500			
102	Number Cumulative %	=E103+E104		
103	Previous Number Cumulative %		=D91	
104	Number %		=(F105/F106)*100	
105	Number (000)			=Yr85_3000to3500_No
106	Total Number (000)			=Yr85_TotalNo
107	Income Cumulative %	=E108+E109		
108	Previous Income Cumulative %		=D96	
109	Income %		=(F110/F111)*100	
110	Total Range Income			=Yr85_3000to3500_Inc
111	Total Year Income			=Yr85_TotalYrInc
112	3500 but under 4000			
113	Number Cumulative %	=E114+E115		
114	Previous Number Cumulative %		=D102	
115	Number %		=(F116/F117)*100	
116	Number (000)			=Yr85_3500to4000_No
117	Total Number (000)			=Yr85_TotalNo
118	Income Cumulative %	=E119+E120		
119	Previous Income Cumulative %		=D107	
120	Income %		=(F121/F122)*100	
121	Total Range Income			=Yr85_3500to4000_Inc
122	Total Year Income			=Yr85_TotalYrInc
123	Total Number (000)			
124	Total Number (000) *	=SUM(Yr85_Number)		
125	Total Year Income			
126	Total Year Income *	=SUM(Yr85_Income)		

Figure C16 (b): Model Schema

101	3000 but under 3500			
102	Number Cumulative %	75.64		
103	Previous Number Cumulative %		51.28	
104	Number %		24.36	
105	Number (000)			1670
106	Total Number (000)			6855
107	Income Cumulative %	68.75		
108	Previous Income Cumulative %		41.65	
109	Income %		27.10	
110	Total Range Income			5420
111	Total Year Income			20000
112	3500 but under 4000			
113	Number Cumulative %	100.00		
114	Previous Number Cumulative %		75.64	
115	Number %		24.36	
116	Number (000)			1670
117	Total Number (000)			6855
118	Income Cumulative %	100.00		
119	Previous Income Cumulative %		68.75	
120	Income %		31.25	
121	Total Range Income			6250
122	Total Year Income			20000
123	Total Number (000)			
124	Total Number (000) *	6855		
125	Total Year Income			
126	Total Year Income *	20000		

Figure C16 (c): References to Modules

STAGE 7:

Implementation of References in the Output Component

The output component of the *Post-tax Income Distribution* model is completed by entering references to formulae in the model schema and data in the input component, in the appropriate cells. *Figure C17 (a)* displays the *surface* level of the output component while *Figures C17 (b)* and *C17 (c)* show segments of its *deep* level or *formula view*. As highly recommended by the proposed methodology, the *references*, ideally, should be in natural language.

	B	C	D	E	F	G	H
02	Post-tax income distribution for 1975 and 1985						
03							
04	Income after tax	Number		Number		Income	
05	1975	(thousands)	Total income	%	Cumulative %	%	Cumulative %
06							
07	675 but under 750	357	255	4.97	4.97	2.84	2.84
08	750 but under 1000	1350	1190	18.81	23.78	13.26	16.10
09	1000 but under 1250	1780	2000	24.80	48.59	22.28	38.38
10	1250 but under 1500	1840	2530	25.64	74.22	28.19	66.57
11	1500 but under 1750	1850	3000	25.78	100.00	33.43	100.00
12							
13	Income after tax						
14	1985						
15							
16	1750 but under 2000	635	1190	9.26	9.26	5.95	5.95
17	2000 but under 2500	1470	3290	21.44	30.71	16.45	22.40
18	2500 but under 3000	1410	3850	20.57	51.28	19.25	41.65
19	3000 but under 3500	1670	5420	24.36	75.64	27.10	68.75
20	3500 but under 4000	1670	6250	24.36	100.00	31.25	100.00
21							

Figure C17 (a): Output Component

	B	C	D
04	Income after tax	Number	
05	1975	(thousands)	Total income
06			
07	675 but under 750	=Yr75_675to750_No	=Yr75_675to750_Inc
08	750 but under 1000	=Yr75_750to1000_No	=Yr75_750to1000_Inc
09	1000 but under 1250	=Yr75_1000to1250_No	=Yr75_1000to1250_Inc
10	1250 but under 1500	=Yr75_1250to1500_No	=Yr75_1250to1500_Inc
11	1500 but under 1750	=Yr75_1500to1750_No	=Yr75_1500to1750_Inc
12			
13	Income after tax		
14	1985		
15			
16	1750 but under 2000	=Yr85_1750to2000_No	=Yr85_1750to2000_Inc
17	2000 but under 2500	=Yr85_2000to2500_No	=Yr85_2000to2500_Inc
18	2500 but under 3000	=Yr85_2500to3000_No	=Yr85_2500to3000_Inc
19	3000 but under 3500	=Yr85_3000to3500_No	=Yr85_3000to3500_Inc
20	3500 but under 4000	=Yr85_3500to4000_No	=Yr85_3500to4000_Inc
21			

References to the *input component*

Figure C17 (b): Output Component

Example 3: *Trading and Profit and Loss Account for Several Years*

STAGE 1:
Requirements Analysis and Development of Output Structures

A typical desired output for this model is shown in *Figure C18*.

	B	C	D	E
02		1997	1998	1999
03	Net profit	?	?	?
04	Unappropriated profits from last year	?	?	?
05	Appropriations	?	?	?
06	Unappropriated profits carried to next year	?	?	?
07				
08				
09				
10		1997	1998	1999
11	Gross Profit	?	?	?
12	Expenses	?	?	?
13	Net profit	?	?	?

Figure C18: Structure of Output

The underlying logic and structure of each formula is determined at the logical level:

- Net profit = Gross profit - Total expenses
- Total appropriations = $\sum (appropriations)$
- Unappropriated profits carried to next year = Net profit +
Unappropriated profits from last year - Total appropriations
- Gross profit = Sales - Cost of goods sold + Closing stock
- Total expenses = $\sum (expenses)$

STAGE 2:
Conceptual Design of the Model Schema

The Jackson structure representation of the conceptual design of the spreadsheet model is displayed in *Figure C19*.

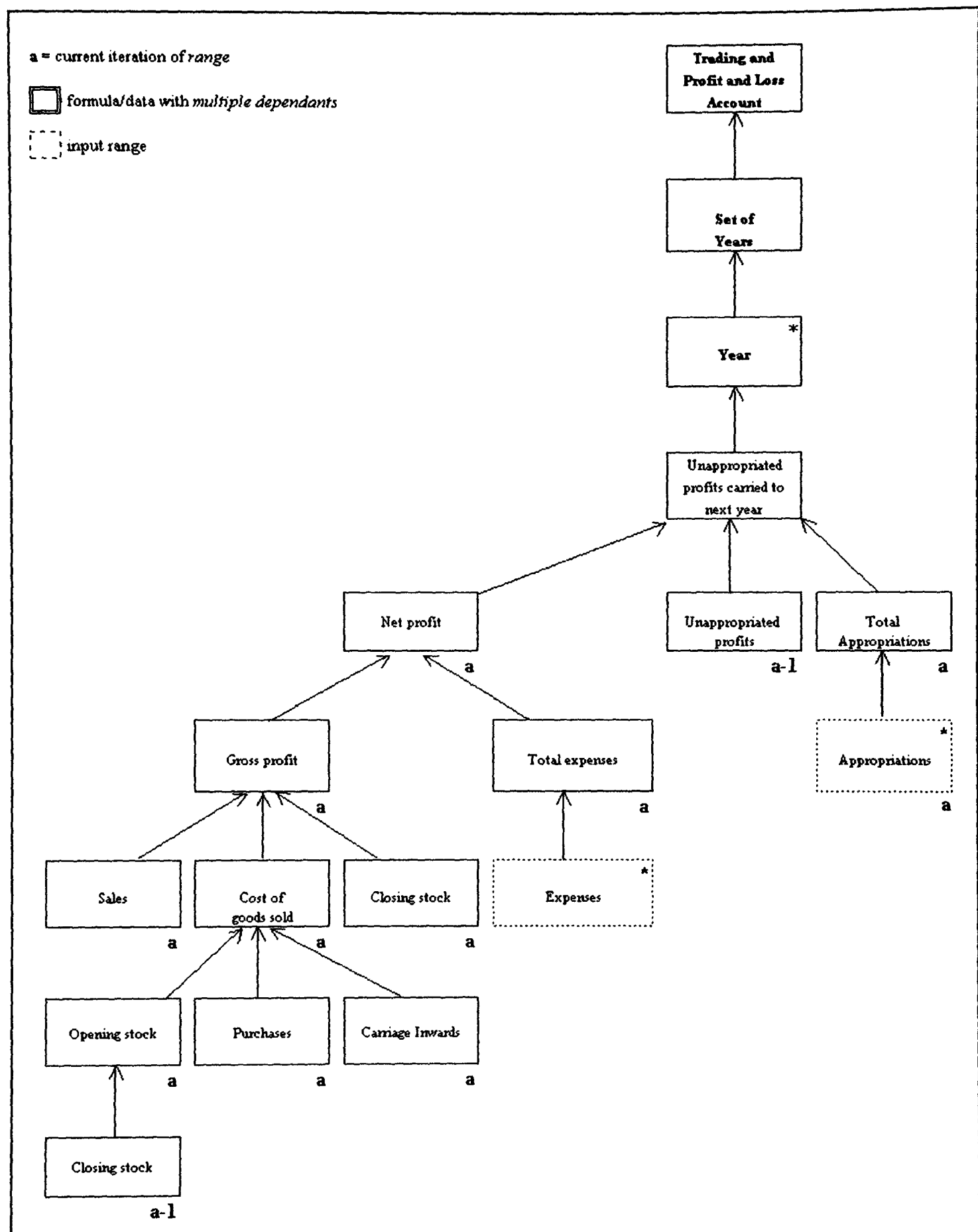


Figure C19: Conceptual Design

STAGE 3:
Logical Design of the Model Schema

In this example, the logical design of the model schema is identical to its conceptual design.

STAGE 4:
Physical Construction of the Model Schema Layout on the Spreadsheet

The logical design of the model is mapped onto a physical spreadsheet structure as displayed in *Figure C20*.

		£	£	£	£	£	£
Year							
1997							
	Unappropriated profits carried to next year						
	Net profit						
	Gross Profit						
	Sales						
	Cost of goods sold						
	Opening stock						
	Closing stock from last year						
	Purchases						
	Carriage inwards						
	Closing stock						
	Total expenses *						
	Unappropriated profits from last year						
	Total appropriations *						
1998							
	Unappropriated profits carried to next year						
	Net profit						
	Gross Profit						
	Sales						
	Cost of goods sold						
	Opening stock						
	Closing stock from last year						
	Purchases						
	Carriage inwards						
	Closing stock						
	Total expenses *						
	Unappropriated profits from last year						
	Total appropriations *						
1999							
	Unappropriated profits carried to next year						
	Net profit						
	Gross Profit						
	Sales						
	Cost of goods sold						
	Opening stock						
	Closing stock from last year						
	Purchases						
	Carriage inwards						
	Closing stock						
	Total expenses *						
	Unappropriated profits from last year						
	Total appropriations *						

Figure C20: Outline of Model Schema

STAGE 5:**Development of the Input Component and Entry of Model Inputs**

The input component for this model is shown in *Figure C21*.

	1996	1997	1998	1999
Unappropriated Profits	15,286			
Sales		135,486	212,670	180,900
Opening stock		40,360	52,360	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
Closing stock	40,360	52,360	44,700	25,430
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240

Figure C21: Input Component

STAGE 6:**Implementation of Formulae and Binding Relationships in the Model Schema**

Figures C22 (a) and C22 (b) show the final state of the model schema. Figure C22 (a) presents the formula view of the model schema while Figure C22 (b) shows the surface values based on the current state of inputs shown in Figure C21.

		£	£	£	£	£	£
Year							
1997							
	Unappropriated profits carried to next year	=E9+E19-E20					
	Net profit		=F10-F18				
	Gross Profit			=G11-G12+G17			
	Sales				=Input!E7		
	Cost of goods sold				=H13+H15+H16		
	Opening stock					=I14	
	Closing stock from last year						=Input!D11
	Purchases					=Input!E9	
	Carriage inwards					=Input!E10	
	Closing stock				=Input!E11		
	Total expenses *			=SUM(Expenses_1997)			
	Unappropriated profits from last year		=Input!D6				
	Total appropriations *		=SUM(Appropriations_1997)				
1998							
	Unappropriated profits carried to next year	=E23+E33-E34					
	Net profit		=F24-F32				
	Gross Profit			=G25-G26+G31			
	Sales				=Input!F7		
	Cost of goods sold				=H27+H29+H30		
	Opening stock					=I28	
	Closing stock from last year						=G17
	Purchases					=Input!F9	
	Carriage inwards					=Input!F10	
	Closing stock				=Input!F11		
	Total expenses *			=SUM(Expenses_1998)			
	Unappropriated profits from last year		=D8				
	Total appropriations *		=SUM(Appropriations_1998)				
1999							
	Unappropriated profits carried to next year	=E37+E47-E48					
	Net profit		=F38-F46				
	Gross Profit			=G39-G40+G45			
	Sales				=Input!G7		
	Cost of goods sold				=H41+H43+H44		
	Opening stock					=I42	
	Closing stock from last year						=G31
	Purchases					=Input!G9	
	Carriage inwards					=Input!G10	
	Closing stock				=Input!G11		
	Total expenses *			=SUM(Expenses_1999)			
	Unappropriated profits from last year		=D22				
	Total appropriations *		=SUM(Appropriations_1999)				

Figure C22 (a): Model Schema (*Formula View*)

	£	£	£	£	£	£
Year						
1997						
Unappropriated profits carried to next year	24,219					
Net profit		20,733				
Gross Profit			73,556			
Sales				135,486		
Cost of goods sold				114,290		
Opening stock					40,360	
Closing stock from last year						40,360
Purchases					72,360	
Carriage inwards					1,570	
Closing stock				52,360		
Total expenses *			52,823			
Unappropriated profits from last year		15,286				
Total appropriations *		11,800				
1998						
Unappropriated profits carried to next year	135,989					
Net profit		123,180				
Gross Profit			151,780			
Sales				212,670		
Cost of goods sold				105,590		
Opening stock					52,360	
Closing stock from last year						52,360
Purchases					52,340	
Carriage inwards					890	
Closing stock				44,700		
Total expenses *			28,600			
Unappropriated profits from last year		24,219				
Total appropriations *		11,410				
1999						
Unappropriated profits carried to next year	233,235					
Net profit		107,236				
Gross Profit			138,376			
Sales				180,900		
Cost of goods sold				67,954		
Opening stock					44,700	
Closing stock from last year						44,700
Purchases					22,109	
Carriage inwards					1,145	
Closing stock				25,430		
Total expenses *			31,140			
Unappropriated profits from last year		135,989				
Total appropriations *		9,990				

Figure C22 (b): Model Schema

STAGE 7:
Implementation of References in the Output Component

The final state of the output component is shown in *Figure C23*.

	B	C	D	E
02		1997	1998	1999
03	Net profit	20,733	123,180	107,236
04	Unappropriated profits from last year	15,286	24,219	135,989
05	Appropriations	11,800	11,410	9,990
06	Unappropriated profits carried to next year	24,219	135,989	233,235
07				
08				
09				
10		1997	1998	1999
11	Gross Profit	73,556	151,780	138,376
12	Expenses	52,823	28,600	31,140
13	Net profit	20,733	123,180	107,236

Figure C23: Model Output

STAGE 8:
Testing, Documentation and Administration of the Spreadsheet Model

Refer to *Section 7.3: Stage 8*.

APPENDIX D SPREADSHEET MODELS USED IN THE EXPERIMENTS

Experiment 1

Item	Formula
Cost of goods sold	= OpeningStock + Purchases + CarriageInwards
Net profit	= GrossProfit - Expenses
Closing stock	{Input}
Unappropriated profits from last year	{Input}
Gross Profit	= Sales - CostOfGoodsSold + ClosingStock
Unappropriated profits carried to next year	= NetProfit + UnappropriatedProfitsFromLastYear - Appropriations
Net profit	= GrossProfit - Expenses
Appropriations	= ProposedDividend + GeneralReserve + ForeignExchange
Sales	{Input}
Expenses	= Salaries + Rates&Occupancy + CarriageOutwards + OfficeExpenses + SundryExpenses + DepreciationBuildings + DepreciationEquipment + DirectorsRemuneration

Figure D1 (a): Formulae

Sales	135,486.00
Opening stock	40,360.00
Closing stock	52,360.00
Purchases	72,360.00
Carriage inwards	1,570.00
Unappropriated profits from last year	15,286.00
Expenses	
Salaries	18,310.00
Rates and occupancy	4,515.00
Carriage outwards	1,390.00
Office expenses	3,212.00
Sundry expenses	1,896.00
Depreciation: Buildings	5,000.00
Depreciation: Equipment	9,000.00
Directors' remuneration	9,500.00
Appropriations	
Proposed dividend	10,000.00
General reserve	1,000.00
Foreign exchange	800.00

Figure D1 (b): Inputs

	B	C	D	E	F	G
		£	£	£	£	£
05						
06	Unappropriated profits carried to next year	24,219				
07	Net profit		20,733			
08	Gross Profit			73,556		
09	Sales				135,486	
10	Cost of goods sold				114,290	
11	Opening stock					40,360
12	Purchases					72,360
13	Carriage inwards					1,570
14	Closing stock				52,360	
15	Total expenses *			52,823		
16	Unappropriated profits from last year		15,286			
17	Total appropriations *		11,800			

Figure D1 (c): Expected Schema

Experiment 2

<u>Item</u>	<u>Formula</u>
Cost of goods sold	$= \text{OpeningStock} + \text{Purchases} + \text{CarriageInwards}$
Net profit	$= \text{GrossProfit} - \text{Expenses}$
Opening stock	$= \text{Previous Year's Closing Stock}$
Closing stock	$\{\text{Input}\}$
Unappropriated profits from last year	$= \text{Previous Year's Unappropriated Profits}$
Gross Profit	$= \text{Sales} - \text{CostOfGoodsSold} + \text{ClosingStock}$
Unappropriated profits carried to next year	$= \text{NetProfit} + \text{UnappropriatedProfitsFromLastYear} - \text{Appropriations}$
Net profit	$= \text{GrossProfit} - \text{Expenses}$
Appropriations	$= \text{ProposedDividend} + \text{GeneralReserve} + \text{ForeignExchange}$
Sales	$\{\text{Input}\}$
Expenses	$= \text{Salaries} + \text{Rates\&Occupancy} + \text{CarriageOutwards}$ $+ \text{OfficeExpenses} + \text{SundryExpenses}$ $+ \text{DepreciationBuildings} + \text{DepreciationEquipment}$ $+ \text{DirectorsRemuneration}$

Figure D2 (a): Formulae

	1996	1997	1998	1999
Sales		135,486	212,670	180,900
Cost of goods sold				
Opening stock		40,360	22,766	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
		112,720	104,290	248,854
Closing stock	40,360	22,766	44,700	25,430
Gross Profit		45,532	108,380	404,324
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
		52,823	28,600	31,140
Net profit		98,355	124,480	149,760
Unappropriated profits from last year		15,286	140,821	113,070
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240
		11,800	11,410	9,990
Unappropriated profits carried to next year	15,286	101,841	113,070	507,404

Figure D2 (b): Seeded Errors Highlighted

	1996	1997	1998	1999
Sales		135,486	212,670	180,900
Cost of goods sold				
Opening stock		=C15	=D15	=E15
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
		=D10+D11	104,290	=F7+F10+F11+F12
Closing stock	40,360	=D7-D13	44,700	25,430
Gross Profit		=D7-D13+D15	=E7-E13	=F7+F13-F15
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
		=SUM(D20:D27)	=SUM(E20:E27)	=SUM(F20:F27)
Net profit		=D17+D28	=E15+E17-E28	=F7-F28
Unappropriated profits from last year		=C40	=140821	=E40
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240
		=SUM(D35:D37)	=SUM(E35:E37)	=SUM(F35:F37)
Unappropriated profits carried to next year	15,286	=D30+D32-D38	=E30-E38	=F17+F32-F38

Figure D2 (c): Seeded Errors Highlighted (Formula View)

	1996	1997	1998	1999
Sales		135,486	212,670	180,900
Cost of goods sold				
Opening stock		40,360	52,360	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
		114,290	105,590	67,954
Closing stock	40,360	52,360	44,700	25,430
Gross Profit		73,556	151,780	138,376
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
		52,823	28,600	31,140
Net profit		20,733	123,180	107,236
Unappropriated profits from last year		15,286	24,219	135,989
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240
		11,800	11,410	9,990
Unappropriated profits carried to next year	15,286	24,219	135,989	233,235

Figure D2 (d): Correct Model

	1996	1997	1998	1999
Unappropriated Profits	15,286			
Sales		135,486	212,670	180,900
Opening stock		40,360	52,360	44,700
Purchases		72,360	52,340	22,109
Carriage inwards		1,570	890	1,145
Closing stock	40,360	52,360	44,700	25,430
Expenses				
Salaries		18,310	10,980	9,800
Rates and occupancy		4,515	3,156	6,540
Carriage outwards		1,390	1,221	1,290
Office expenses		3,212	890	2,875
Sundry expenses		1,896	1,003	890
Depreciation: Buildings		5,000	3,050	900
Depreciation: Equipment		9,000	6,000	1,245
Directors' remuneration		9,500	2,300	7,600
Appropriations				
Proposed dividend		10,000	8,600	7,800
General reserve		1,000	2,050	950
Foreign exchange		800	760	1,240

Figure D3 (a): Inputs

Year	£	£	£	£	£	£
1997						
Unappropriated profits carried to next year	101,841					
Net profit		98,355				
Gross Profit			45,532			
Sales				135,486		
Cost of goods sold				112,720		
Opening stock					40,360	
Closing stock from last year						40,360
Purchases					72,360	
Carriage inwards					1,570	
Closing stock				22,766		
Total expenses *			52,823			
Unappropriated profits from last year		15,286				
Total appropriations *		11,800				
1998						
Unappropriated profits carried to next year	113,070					
Net profit		124,480				
Gross Profit			108,380			
Sales				212,670		
Cost of goods sold				104,290		
Opening stock					22,766	
Closing stock from last year						22,766
Purchases					52,340	
Carriage inwards					890	
Closing stock				44,700		
Total expenses *			28,600			
Unappropriated profits from last year		140,821				
Total appropriations *		11,410				
1999						
Unappropriated profits carried to next year	507,404					
Net profit		149,760				
Gross Profit			404,324			
Sales				180,900		
Cost of goods sold				248,854		
Opening stock					44,700	
Closing stock from last year						44,700
Purchases					22,109	
Carriage inwards					1,145	
Closing stock				25,430		
Total expenses *			31,140			
Unappropriated profits from last year		113,070				
Total appropriations *		9,990				

Figure D3 (b): Seeded Errors Highlighted

Year		£	£	£	£	£	£
1997							
	Unappropriated profits carried to next year	=E9+E19-E20					
	Net profit	=F10•F18					
	Gross Profit		=G11-G12+G17				
	Sales			=Input!E7			
	Cost of goods sold			=H13•H15			
	Opening stock				=I14		
	Closing stock from last year					=Input!D11	
	Purchases				=Input!E9		
	Carriage inwards				=Input!E10		
	Closing stock			=G11-G12			
	Total expenses *		=SUM(Expenses 1997)				
	Unappropriated profits from last year	=Input!D6					
	Total appropriations *	=SUM(Appropriations 1997)					
1998							
	Unappropriated profits carried to next year	=E23-E34					
	Net profit	=G31+F24-F32					
	Gross Profit		=G25-G26				
	Sales			=Input!F7			
	Cost of goods sold			104290			
	Opening stock				=I28		
	Closing stock from last year					=G17	
	Purchases				=Input!F9		
	Carriage inwards				=Input!F10		
	Closing stock			=Input!F11			
	Total expenses *		=SUM(Expenses 1998)				
	Unappropriated profits from last year	140821					
	Total appropriations *	=SUM(Appropriations 1998)					
1999							
	Unappropriated profits carried to next year	=F38+E47-E48					
	Net profit	=G39-F46					
	Gross Profit		=G39+G40-G45				
	Sales			=Input!G7			
	Cost of goods sold			=G39+H41+H43+H44			
	Opening stock				=I42		
	Closing stock from last year					=G31	
	Purchases				=Input!G9		
	Carriage inwards				=Input!G10		
	Closing stock			=Input!G11			
	Total expenses *		=SUM(Expenses 1999)				
	Unappropriated profits from last year	=D22					
	Total appropriations *	=SUM(Appropriations 1999)					

Figure D3 (c): Seeded Errors Highlighted (Formula View)

	£	£	£	£	£	£
Year						
1997						
Unappropriated profits carried to next year	24,219					
Net profit		20,733				
Gross Profit			73,556			
Sales				135,486		
Cost of goods sold				114,290		
Opening stock					40,360	
Closing stock from last year						40,360
Purchases					72,360	
Carriage inwards					1,570	
Closing stock				52,360		
Total expenses *			52,823			
Unappropriated profits from last year		15,286				
Total appropriations *		11,800				
1998						
Unappropriated profits carried to next year	135,989					
Net profit		123,180				
Gross Profit			151,780			
Sales				212,670		
Cost of goods sold				105,590		
Opening stock					52,360	
Closing stock from last year						52,360
Purchases					52,340	
Carriage inwards					890	
Closing stock				44,700		
Total expenses *			28,600			
Unappropriated profits from last year		24,219				
Total appropriations *		11,410				
1999						
Unappropriated profits carried to next year	233,235					
Net profit		107,236				
Gross Profit			138,376			
Sales				180,900		
Cost of goods sold				67,954		
Opening stock					44,700	
Closing stock from last year						44,700
Purchases					22,109	
Carriage inwards					1,145	
Closing stock				25,430		
Total expenses *			31,140			
Unappropriated profits from last year		135,989				
Total appropriations *		9,990				

Figure D3 (d): Correct Model

Experiment 3

<u>Item</u>	<u>Formula</u>
Number %	$= (\text{Number ('000)} / \text{Total Number ('000)}) * 100$
Number Cumulative %	$= \text{Number \%} + \text{Number Cumulative \% of previous range}$
Total Number ('000)	$= \text{SUM of Total Number ('000) for each range (within a year)}$ $= \text{Total Number ('000)}_{\text{RANGE 1}} + \text{Total Number ('000)}_{\text{RANGE 2}}$ $+ \text{Total Number ('000)}_{\text{RANGE n}}$ $= \text{SUM (Total Number ('000)}_{\text{RANGE 1}}; \text{Total Number ('000)}_{\text{RANGE n}})$
Number ('000) for each range	$\{input\}$
Income %	$= (\text{Total Range Income} / \text{Total Year Income}) * 100$
Income Cumulative %	$= \text{Income \%} + \text{Income Cumulative \% of previous range}$
Total Year Income	$= \text{SUM of Total Range Income for each range (within a year)}$ $= \text{Total Range Income}_{\text{RANGE 1}} + \text{Total Range Income}_{\text{RANGE 2}}$ $+ \text{Total Range Income}_{\text{RANGE n}}$ $= \text{SUM (Total Range Income}_{\text{RANGE 1}}; \text{Total Range Income}_{\text{RANGE n}})$
Total Range Income for each range	$\{input\}$

Figure D4 (a): Formulae

	B	C	D	E	F	G	H
02	Post-tax income distribution for 1975 and 1985						
03							
04	Income after tax	Number					
05	1975	(thousands)	Total income	Number		Income	
06				%	Cumulative %	%	Cumulative %
07	675 but under 750	357	255	4.97	23.78	2.84	2.84
08	750 but under 1000	1350	1190	18.81	42.59	13.62	16.46
09	1000 but under 1250	1780	2000	24.80	67.40	22.28	38.74
10	1250 but under 1500	1840	2530	25.64	93.03	28.19	44.65
11	1500 but under 1750	1850	3000	25.78	118.81	33.43	78.08
12							
13	Income after tax						
14	1985						
15							
16	1750 but under 2000	635	1190	9.26	9.26	5.95	5.95
17	2000 but under 2500	1470	3290	47.99	57.26	16.45	22.40
18	2500 but under 3000	1410	3850	20.57	15.28	19.25	3.15
19	3000 but under 3500	1670	5420	24.36	39.64	2.71	5.86
20	3500 but under 4000	1670	6250	0.24	39.89	31.25	37.11
21							

Figure D4 (b): Model with Seeded Errors (Conventional Layout)

	B	E	F	G	H
04	Income after tax	Number	Number	Income	Income
05	1975	%	Cumulative %	%	Cumulative %
06					
07	675 but under 750	$=(C7/SUM(C7:C11))*100$	$=E7+E8$	$=(D7/SUM(D7:D11))*100$	$=G7$
08	750 but under 1000	$=(C8/SUM(C7:C11))*100$	$=F7+E8$	13.62	$=H7+G8$
09	1000 but under 1250	$=(C9/SUM(C7:C11))*100$	$=F8+E9$	$=(D9/SUM(D7:D11))*100$	$=16.46+22.28$
10	1250 but under 1500	$=(C10/SUM(C7:C11))*100$	$=F9+E10$	$=(D10/SUM(D7:D11))*100$	$=H9+G10$
11	1500 but under 1750	$=(1850/SUM(C7:C11))*100$	$=F10+E11$	$=(D11/SUM(D7:D11))*100$	$=H10+G11$
12					
13	Income after tax				
14	1985				
15					
16	1750 but under 2000	$=(C16/SUM(C16:C20))*100$	$=E16$	$=(D16/SUM(D16:D20))*100$	$=G16$
17	2000 but under 2500	$=(D17/SUM(C16:C20))*100$	$=F16+E17$	$=(D17/SUM(D16:D20))*100$	$=H16+G17$
18	2500 but under 3000	$=(C18/SUM(C16:C20))*100$	$=F17+E17$	$=(D18/SUM(D16:D20))*100$	$=H16+G18$
19	3000 but under 3500	$=(C19/SUM(C16:C20))*100$	$=F18+E19$	$=(D19/SUM(D16:D20))*100$	$=H18+G19$
20	3500 but under 4000	$=(C20/D20)*100$	$=F19+E20$	$=(D20/SUM(D16:D20))*100$	$=H19+G20$
21					

Figure D4 (c): Model with Seeded Errors (Conventional Layout - Formula View)

	B	C	D	E	F	G	H
02	Post-tax income distribution for 1975 and 1985						
03							
04	<i>Income after tax</i>	<i>Number</i>		<i>Number</i>		<i>Income</i>	
05	1975	<i>(thousands)</i>	<i>Total income</i>	<i>%</i>	<i>Cumulative %</i>	<i>%</i>	<i>Cumulative %</i>
06							
07	675 but under 750	357	255	4.97	4.97	2.84	2.84
08	750 but under 1000	1350	1190	18.81	23.78	13.26	16.10
09	1000 but under 1250	1780	2000	24.80	48.59	22.28	38.38
10	1250 but under 1500	1840	2530	25.64	74.22	28.19	66.57
11	1500 but under 1750	1850	3000	25.78	100.00	33.43	100.00
12							
13	<i>Income after tax</i>						
14	1985						
15							
16	1750 but under 2000	635	1190	9.26	9.26	5.95	5.95
17	2000 but under 2500	1470	3290	21.44	30.71	16.45	22.40
18	2500 but under 3000	1410	3850	20.57	51.28	19.25	41.65
19	3000 but under 3500	1670	5420	24.36	75.64	27.10	68.75
20	3500 but under 4000	1670	6250	24.36	100.00	31.25	100.00
21							

Figure D4 (d): Correct Version of the Model (Conventional Layout)

	B	C	D	E
04	Year		Number ('000)	Total Income
05	1975			
06	Range			
07		675 but under 750	357	255
08		750 but under 1000	1350	1190
09		1000 but under 1250	1780	2000
10		1250 but under 1500	1840	2530
11		1500 but under 1750	1850	3000
12	1985			
13	Range			
14		1750 but under 2000	635	1190
15		2000 but under 2500	1470	3290
16		2500 but under 3000	1410	3850
17		3000 but under 3500	1670	5420
18		3500 but under 4000	1670	6250

Figure D5 (a): Input Component

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008		Number Cumulative %	23.78		
009		Previous Number Cumulative %		0.00	
010		Number %		4.97	
011		Number (000)			357
012		Total Number (000)			7177
013		Income Cumulative %	2.84		
014		Previous Income Cumulative %		0.00	
015		Income %		2.84	
016		Total Range Income			255
017		Total Year Income			8975
018	750 but under 1000				
019		Number Cumulative %	42.59		
020		Previous Number Cumulative %		23.78	
021		Number %		18.81	
022		Number (000)			1350
023		Total Number (000)			7177
024		Income Cumulative %	16.46		
025		Previous Income Cumulative %		2.84	
026		Income %		13.62	
027		Total Range Income			1190
028		Total Year Income			8975
029	1000 but under 1250				
030		Number Cumulative %	67.40		
031		Previous Number Cumulative %		42.59	
032		Number %		24.80	
033		Number (000)			1780
034		Total Number (000)			7177
035		Income Cumulative %	38.74		
036		Previous Income Cumulative %		16.46	
037		Income %		22.28	
038		Total Range Income			2000
039		Total Year Income			8975
040	1250 but under 1500				
041		Number Cumulative %	93.03		
042		Previous Number Cumulative %		67.40	
043		Number %		25.64	
044		Number (000)			1840
045		Total Number (000)			7177
046		Income Cumulative %	10.55		
047		Previous Income Cumulative %		38.74	
048		Income %		28.19	
049		Total Range Income			2530
050		Total Year Income			8975
051	1500 but under 1750				
052		Number Cumulative %	118.81		
053		Previous Number Cumulative %		93.03	
054		Number %		25.78	
055		Number (000)			1850
056		Total Number (000)			7177
057		Income Cumulative %	43.98		
058		Previous Income Cumulative %		10.55	
059		Income %		33.43	
060		Total Range Income			3000
061		Total Year Income			8975
062	Total Number (000)				
063	Total Number (000) *		7177		
064	Total Year Income				
065	Total Year Income *		8975		

continued ...

	B	C	D	E	F
066	1985				
067	Range				
068	1750 but under 2000				
069		Number Cumulative %	9.26		
070		Previous Number Cumulative %		0.00	
071		Number %		9.26	
072		Number (000)			635
073		Total Number (000)			6855
074		Income Cumulative %	17.36		
075		Previous Income Cumulative %		0.00	
076		Income %		17.36	
077		Total Range Income			1190
078		Total Year Income			20000
079	2000 but under 2500				
080		Number Cumulative %	57.26		
081		Previous Number Cumulative %		9.26	
082		Number %		47.99	
083		Number (000)			1470
084		Total Number (000)			6855
085		Income Cumulative %	33.81		
086		Previous Income Cumulative %		17.36	
087		Income %		16.45	
088		Total Range Income			3290
089		Total Year Income			20000
090	2500 but under 3000				
091		Number Cumulative %	105.25		
092		Previous Number Cumulative %		57.26	
093		Number %		20.57	
094		Number (000)			1410
095		Total Number (000)			6855
096		Income Cumulative %	36.61		
097		Previous Income Cumulative %		33.81	
098		Income %		19.25	
099		Total Range Income			3850
100		Total Year Income			20000
101	3000 but under 3500				
102		Number Cumulative %	129.61		
103		Previous Number Cumulative %		105.25	
104		Number %		24.36	
105		Number (000)			1670
106		Total Number (000)			6855
107		Income Cumulative %	63.71		
108		Previous Income Cumulative %		36.61	
109		Income %		27.10	
110		Total Range Income			5420
111		Total Year Income			20000
112	3500 but under 4000				
113		Number Cumulative %	156.33		
114		Previous Number Cumulative %		129.61	
115		Number %		26.72	
116		Number (000)			1670
117		Total Number (000)			6855
118		Income Cumulative %	94.96		
119		Previous Income Cumulative %		63.71	
120		Income %		31.25	
121		Total Range Income			6250
122		Total Year Income			20000
123	Total Number (000)				
124	Total Number (000) *		6855		
125	Total Year Income				
126	Total Year Income *		20000		

Figure D5 (b): Model with Seeded Errors (Structured Layout)

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008		Number Cumulative %	=E10+E21		
009		Previous Number Cumulative %		=0	
010		Number %		=(F11/F12)*100	
011		Number (000)			=Input!D7
012		Total Number (000)			=D63
013		Income Cumulative %	=E14+E15		
014		Previous Income Cumulative %		=0	
015		Income %		=(F16/F17)*100	
016		Total Range Income			=Input!E7
017		Total Year Income			=D65
018	750 but under 1000				
019		Number Cumulative %	=E20+E21		
020		Previous Number Cumulative %		=D8	
021		Number %		=(F22/F23)*100	
022		Number (000)			=Input!D8
023		Total Number (000)			=D63
024		Income Cumulative %	=E25+E26		
025		Previous Income Cumulative %		=D13	
026		Income %		13.62	
027		Total Range Income			=Input!E8
028		Total Year Income			=D65
029	1000 but under 1250				
030		Number Cumulative %	=E31+E32		
031		Previous Number Cumulative %		=D19	
032		Number %		=(F33/F34)*100	
033		Number (000)			=Input!D9
034		Total Number (000)			=D63
035		Income Cumulative %	=16.46+22.28		
036		Previous Income Cumulative %		=D24	
037		Income %		=(F38/F39)*100	
038		Total Range Income			=Input!E9
039		Total Year Income			=D65
040	1250 but under 1500				
041		Number Cumulative %	=E42+E43		
042		Previous Number Cumulative %		=D30	
043		Number %		=(F44/F45)*100	
044		Number (000)			=Input!D10
045		Total Number (000)			=D63
046		Income Cumulative %	=E47-E48		
047		Previous Income Cumulative %		=D35	
048		Income %		=(F49/F50)*100	
049		Total Range Income			=Input!E10
050		Total Year Income			=D65
051	1500 but under 1750				
052		Number Cumulative %	=E53+E54		
053		Previous Number Cumulative %		=D41	
054		Number %		=(1850/F56)*100	
055		Number (000)			=Input!D11
056		Total Number (000)			=D63
057		Income Cumulative %	=E58+E59		
058		Previous Income Cumulative %		=D46	
059		Income %		=(F60/F61)*100	
060		Total Range Income			=Input!E11
061		Total Year Income			=D65
062	Total Number (000)				
063		Total Number (000) *	=SUM(Input!D7:D11)		
064	Total Year Income				
065		Total Year Income *	=SUM(Input!E7:E11)		

continued ...

	B	C	D	E	F
066	1985				
067	Range				
068	1750 but under 2000				
069		Number Cumulative %	=E70+E71		
070		Previous Number Cumulative %		=0	
071		Number %		=(F72/F73)*100	
072		Number (000)			=Input!D14
073		Total Number (000)			=D124
074		Income Cumulative %	=E75+E76		
075		Previous Income Cumulative %		=0	
076		Income %		=(F77/F73)*100	
077		Total Range Income			=Input!E14
078		Total Year Income			=D126
079	2000 but under 2500				
080		Number Cumulative %	=E81+E82		
081		Previous Number Cumulative %		=D69	
082		Number %		=(F88/F84)*100	
083		Number (000)			=Input!D15
084		Total Number (000)			=D124
085		Income Cumulative %	=E86+E87		
086		Previous Income Cumulative %		=D74	
087		Income %		=(F88/F89)*100	
088		Total Range Income			=Input!E15
089		Total Year Income			=D126
090	2500 but under 3000				
091		Number Cumulative %	=E92+E82		
092		Previous Number Cumulative %		=D80	
093		Number %		=(F94/F95)*100	
094		Number (000)			=Input!D16
095		Total Number (000)			=D124
096		Income Cumulative %	=E86+E98		
097		Previous Income Cumulative %		=D85	
098		Income %		=(F99/F100)*100	
099		Total Range Income			=Input!E16
100		Total Year Income			=D126
101	3000 but under 3500				
102		Number Cumulative %	=E103+E104		
103		Previous Number Cumulative %		=D91	
104		Number %		=(F105/F106)*100	
105		Number (000)			=Input!D17
106		Total Number (000)			=D124
107		Income Cumulative %	=E108+E109		
108		Previous Income Cumulative %		=D96	
109		Income %		=(F110/F111)*100	
110		Total Range Income			=Input!E17
111		Total Year Income			=D126
112	3500 but under 4000				
113		Number Cumulative %	=E114+E115		
114		Previous Number Cumulative %		=D102	
115		Number %		=(F116/F121)*100	
116		Number (000)			=Input!D18
117		Total Number (000)			=D124
118		Income Cumulative %	=E119+E120		
119		Previous Income Cumulative %		=D107	
120		Income %		=(F121/F122)*100	
121		Total Range Income			=Input!E18
122		Total Year Income			=D126
123	Total Number (000)				
124		Total Number (000) *	=SUM(Input!D14:D18)		
125	Total Year Income				
126		Total Year Income *	=SUM(Input!E14:E18)		

Figure D5 (c): Model with Seeded Errors (Structured Layout – Formula View)

	B	C	D	E	F
004	Year				
005	1975				
006	Range				
007	675 but under 750				
008		Number Cumulative %	4.97		
009		<i>Previous Number Cumulative %</i>		0.00	
010		Number %		4.97	
011		Number (000)			357
012		<i>Total Number (000)</i>			7177
013		Income Cumulative %	2.84		
014		<i>Previous Income Cumulative %</i>		0.00	
015		Income %		2.84	
016		Total Range Income			255
017		<i>Total Year Income</i>			8975
018	750 but under 1000				
019		Number Cumulative %	23.78		
020		<i>Previous Number Cumulative %</i>		4.97	
021		Number %		18.81	
022		Number (000)			1350
023		<i>Total Number (000)</i>			7177
024		Income Cumulative %	16.10		
025		<i>Previous Income Cumulative %</i>		2.84	
026		Income %		13.26	
027		Total Range Income			1190
028		<i>Total Year Income</i>			8975
029	1000 but under 1250				
030		Number Cumulative %	48.59		
031		<i>Previous Number Cumulative %</i>		23.78	
032		Number %		24.80	
033		Number (000)			1780
034		<i>Total Number (000)</i>			7177
035		Income Cumulative %	38.38		
036		<i>Previous Income Cumulative %</i>		16.10	
037		Income %		22.28	
038		Total Range Income			2000
039		<i>Total Year Income</i>			8975
040	1250 but under 1500				
041		Number Cumulative %	74.22		
042		<i>Previous Number Cumulative %</i>		48.59	
043		Number %		25.64	
044		Number (000)			1840
045		<i>Total Number (000)</i>			7177
046		Income Cumulative %	66.57		
047		<i>Previous Income Cumulative %</i>		38.38	
048		Income %		28.19	
049		Total Range Income			2530
050		<i>Total Year Income</i>			8975
051	1500 but under 1750				
052		Number Cumulative %	100.00		
053		<i>Previous Number Cumulative %</i>		74.22	
054		Number %		25.78	
055		Number (000)			1850
056		<i>Total Number (000)</i>			7177
057		Income Cumulative %	100.00		
058		<i>Previous Income Cumulative %</i>		66.57	
059		Income %		33.43	
060		Total Range Income			3000
061		<i>Total Year Income</i>			8975
062	Total Number (000)				
063	Total Number (000) *		7177		
064	Total Year Income				
065	Total Year Income *		8975		

continued ...

	B	C	D	E	F
066	1985				
067	Range				
068	1750 but under 2000				
069		Number Cumulative %	9.26		
070		<i>Previous Number Cumulative %</i>		0.00	
071		Number %		9.26	
072		Number (000)			635
073		<i>Total Number (000)</i>			6855
074		Income Cumulative %	5.95		
075		<i>Previous Income Cumulative %</i>		0.00	
076		Income %		5.95	
077		Total Range Income			1190
078		<i>Total Year Income</i>			20000
079	2000 but under 2500				
080		Number Cumulative %	30.71		
081		<i>Previous Number Cumulative %</i>		9.26	
082		Number %		21.44	
083		Number (000)			1470
084		<i>Total Number (000)</i>			6855
085		Income Cumulative %	22.40		
086		<i>Previous Income Cumulative %</i>		5.95	
087		Income %		16.45	
088		Total Range Income			3290
089		<i>Total Year Income</i>			20000
090	2500 but under 3000				
091		Number Cumulative %	51.28		
092		<i>Previous Number Cumulative %</i>		30.71	
093		Number %		20.57	
094		Number (000)			1410
095		<i>Total Number (000)</i>			6855
096		Income Cumulative %	41.65		
097		<i>Previous Income Cumulative %</i>		22.40	
098		Income %		19.25	
099		Total Range Income			3850
100		<i>Total Year Income</i>			20000
101	3000 but under 3500				
102		Number Cumulative %	75.64		
103		<i>Previous Number Cumulative %</i>		51.28	
104		Number %		24.36	
105		Number (000)			1670
106		<i>Total Number (000)</i>			6855
107		Income Cumulative %	68.75		
108		<i>Previous Income Cumulative %</i>		41.65	
109		Income %		27.10	
110		Total Range Income			5420
111		<i>Total Year Income</i>			20000
112	3500 but under 4000				
113		Number Cumulative %	100.00		
114		<i>Previous Number Cumulative %</i>		75.64	
115		Number %		24.36	
116		Number (000)			1670
117		<i>Total Number (000)</i>			6855
118		Income Cumulative %	100.00		
119		<i>Previous Income Cumulative %</i>		68.75	
120		Income %		31.25	
121		Total Range Income			6250
122		<i>Total Year Income</i>			20000
123	Total Number (000)				
124	Total Number (000) *		6855		
125	Total Year Income				
126	Total Year Income *		20000		

Figure D5 (d): Correct Version of the Model (Structured Layout)

Experiment 4

Model file name: Xls

Warning: these are draft results, the model is still under development

Model version 1.3

Country: United Kingdom Subsidiary: North

Scenario description: Aggressive growth assumed post 2001, cost base as yet unchanged

Balance sheet

All units are £000s

Balance sheet

	1999/2000	2000/01	2001/02	2002/03	2003/04	2004/05	2005/06
Fixed assets	56,742	55,622	54,481	53,318	52,132	50,921	49,679
<i>Current assets</i>							
Cash	3,251	4,372	2,375	1,218	6,722	15,764	25,627
Debtors	3,088	3,328	3,602	3,762	4,142	4,566	5,151
Total assets	63,082	63,323	60,458	58,299	62,997	71,250	80,457
<i>Current liabilities</i>							
Debt	51,411	51,411	51,411	51,411	51,411	51,411	51,411
Creditors	3,122	2,619	2,541	2,648	2,857	3,053	3,505
Capital creditors	1,170	136	140	144	149	154	159
Tax creditor	132	352	1,358	1,729	2,388	3,227	3,715
Total liabilities	55,835	54,517	55,450	55,932	56,804	57,844	58,790
Net assets	7,247	8,805	5,008	2,367	6,192	13,406	21,667
Share capital	1,000	1,000	1,000	1,000	1,000	1,000	1,000
P&L reserve	6,247	7,805	4,008	1,367	5,192	12,406	20,667
Shareholders' funds	7,247	8,805	5,008	2,367	6,192	13,406	21,667

Results prepared by Nick Read

13/12/1998 12:17pm

Figure D6 (a): Original Model

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s	1999/2000	2000/01	2001/02	2002/03
04					
05					
06	Fixed assets	56,742	55,622	54,481	53,318
07					
08	<i>Current assets</i>				
09	Cash	3,251	4,372	2,375	1,218
10	Debtors	3,088	3,328	3,602	3,762
11					
12	Total assets	63,081	63,322	60,458	58,298
13					
14	<i>Current liabilities</i>				
15	Debt	51,411	51,411	51,411	51,411
16	Creditors	3,122	2,619	2,541	2,648
17	Capital creditors	1,170	136	140	144
18	Tax creditor	132	352	1,358	1,729
19					
20	Total liabilities	55,835	54,518	55,450	55,932
21					
22	Net assets	7,246	8,804	5,008	2,366
23					
24	Share capital	1,000	1,000	1,000	1,000
25	P&L reserve	6,246	7,804	4,008	1,366
26					
27	Shareholders' funds	7,246	8,804	5,008	2,366

Figure D6 (b): Abridged Model

Item	Formula
Fixed assets	{Input}
Cash	{Input}
Debtors	{Input}
Total assets	= Fixed assets + Cash + Debtors
Debt	{Input}
Creditors	{Input}
Capital creditors	{Input}
Tax creditor	{Input}
Total liabilities	= Debt + Creditors + Capital creditors + Tax creditor
Net assets	= Total assets - Total liabilities
Share capital	{Input}
P&L reserve	{Input}
Shareholders' funds	= Share capital + P&L reserve

Figure D6 (c): Formulae

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s	1999/2000	2000/01	2001/02	2002/03
04					
05					
06	Fixed assets	56,742	55,622	54,481	53,318
07					
08	<i>Current assets</i>				
09	Cash	3,251	4,372	2,375	1,218
10	Debtors	3,088	3,328	3,602	3,762
11					
12	Total assets	59,993	63,322	5,977	58,298
13					
14	<i>Current liabilities</i>				
15	Debt	51,411	51,411	51,411	51,411
16	Creditors	3,122	2,619	2,541	2,648
17	Capital creditors	1,170	136	140	144
18	Tax creditor	132	352	1,358	1,729
19					
20	Total liabilities	55,835	54,166	52,769	55,932
21					
22	Net assets	4,581	9,156	58,746	1,907
23					
24	Share capital	1,000	1,000	1,000	1,000
25	P&L reserve	6,246	7,804	4,008	1,366
26					
27	Shareholders' funds	7,246	8,804	5,008	1,366

Figure D6 (d): Errors Highlighted

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s	1999/2000	2000/01	2001/02	2002/03
04					
05					
06	Fixed assets	56742	55622	54481	53318
07					
08	Current assets				
09	Cash	3251	4372	2375	1218
10	Debtors	3088	3328	3602	3762
11					
12	Total assets	=C6+C9	=D6+D9+D10	=E9+E10+E11	58298
13					
14	Current liabilities				
15	Debt	51411	51411	51411	51411
16	Creditors	3122	2619	2541	2648
17	Capital creditors	1170	136	140	144
18	Tax creditor	132	352	1358	1729
19					
20	Total liabilities	=SUM(C15:C18)	=SUM(D14:D17)	=SUM(E15+E18)	=SUM(F15:F18)
21					
22	Net assets	4581	=D12-D20	=E12+E20	=F6-F15
23					
24	Share capital	1000	1000	1000	1000
25	P&L reserve	6246	7804	4008	1366
26					
27	Shareholders' funds	=SUM(C24:C26)	=SUM(1000+7804)	=SUM(E24:E26)	=SUM(F25:F26)

Figure D6 (e): Errors Highlighted (Formula View)

	B	C	D	E	F	G
02			1999/2000	2000/01	2001/02	2002/03
03			£000s	£000s	£000s	£000s
04		Fixed assets	56,742	55,622	54,481	53,318
05		Current assets				
06		Cash	3,251	4,372	2,375	1,218
07		Debtors	3,088	3,328	3,602	3,762
08		Current liabilities				
09		Debt	51,411	51,411	51,411	51,411
10		Creditors	3,122	2,619	2,541	2,648
11		Capital creditors	1,170	136	140	144
12		Tax creditor	132	352	1,358	1,729
13		Shareholders' funds				
14		Share capital	1,000	1,000	1,000	1,000
15		P&L reserve	6,246	7,804	4,008	1,366

Figure D7 (a): Input Component

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s				
04					
05	Year				
06	1999/2000				
07		Net assets	4,581		
08		Total assets		59,993	
09		Fixed assets			56,742
10		Cash			3,251
11		Debtors			3,088
12		Total liabilities		55,835	
13		Debt			51,411
14		Creditors			3,122
15		Capital creditors			1,170
16		Tax creditor			132
17		Shareholders' funds	7,246		
18		Share capital		1,000	
19		P&L reserve		6,246	
20	2000/01				
21		Net assets	9,156		
22		Total assets		63,322	
23		Fixed assets			55,622
24		Cash			4,372
25		Debtors			3,328
26		Total liabilities		54,166	
27		Debt			51,411
28		Creditors			2,619
29		Capital creditors			136
30		Tax creditor			352
31		Shareholders' funds	8,804		
32		Share capital		1,000	
33		P&L reserve		7,804	
34	2001/02				
35		Net assets	58,746		
36		Total assets		5,977	
37		Fixed assets			54,481
38		Cash			2,375
39		Debtors			3,602
40		Total liabilities		52,769	
41		Debt			51,411
42		Creditors			2,541
43		Capital creditors			140
44		Tax creditor			1,358
45		Shareholders' funds	5,008		
46		Share capital		1,000	
47		P&L reserve		4,008	
48	2002/03				
49		Net assets	1,907		
50		Total assets		58,298	
51		Fixed assets			53,318
52		Cash			1,218
53		Debtors			3,762
54		Total liabilities		55,932	
55		Debt			51,411
56		Creditors			2,648
57		Capital creditors			144
58		Tax creditor			1,729
59		Shareholders' funds	1,366		
60		Share capital		1,000	
61		P&L reserve		1,366	

Figure D7 (b): Errors Highlighted

	B	C	D	E	F
05	Year				
06	1999/2000				
07		Net assets	4581		
08		Total assets		=F9+F10	
09		Fixed assets			=Input!D4
10		Cash			=Input!D6
11		Debtors			=Input!D7
12		Total liabilities		=SUM(F13:F16)	
13		Debt			=Input!D9
14		Creditors			=Input!D10
15		Capital creditors			=Input!D11
16		Tax creditor			=Input!D12
17		Shareholders' funds	=E18+E19		
18		Share capital		=Input!D14	
19		P&L reserve		=Input!D15	
20	2000/01				
21		Net assets	=E22-E26		
22		Total assets		=F23+F24+F25	
23		Fixed assets			=Input!E4
24		Cash			=Input!E6
25		Debtors			=Input!E7
26		Total liabilities		=SUM(F26:F29)	
27		Debt			=Input!E9
28		Creditors			=Input!E10
29		Capital creditors			=Input!E11
30		Tax creditor			=Input!E12
31		Shareholders' funds	=1000+7804		
32		Share capital		=Input!E14	
33		P&L reserve		=Input!E15	
34	2001/02				
35		Net assets	=E36+E40		
36		Total assets		=F38+F39+F40	
37		Fixed assets			=Input!F4
38		Cash			=Input!F6
39		Debtors			=Input!F7
40		Total liabilities		=SUM(F41:F44)	
41		Debt			=Input!F9
42		Creditors			=Input!F10
43		Capital creditors			=Input!F11
44		Tax creditor			=Input!F12
45		Shareholders' funds	=E46+E47		
46		Share capital		=Input!F14	
47		P&L reserve		=Input!F15	
48	2002/03				
49		Net assets	=F51-F55		
50		Total assets		58298	
51		Fixed assets			=Input!G4
52		Cash			=Input!G6
53		Debtors			=Input!G7
54		Total liabilities		=SUM(F55:F58)	
55		Debt			=Input!G9
56		Creditors			=Input!G10
57		Capital creditors			=Input!G11
58		Tax creditor			=Input!G12
59		Shareholders' funds	=E61+E62		
60		Share capital		=Input!G14	
61		P&L reserve		=Input!G15	

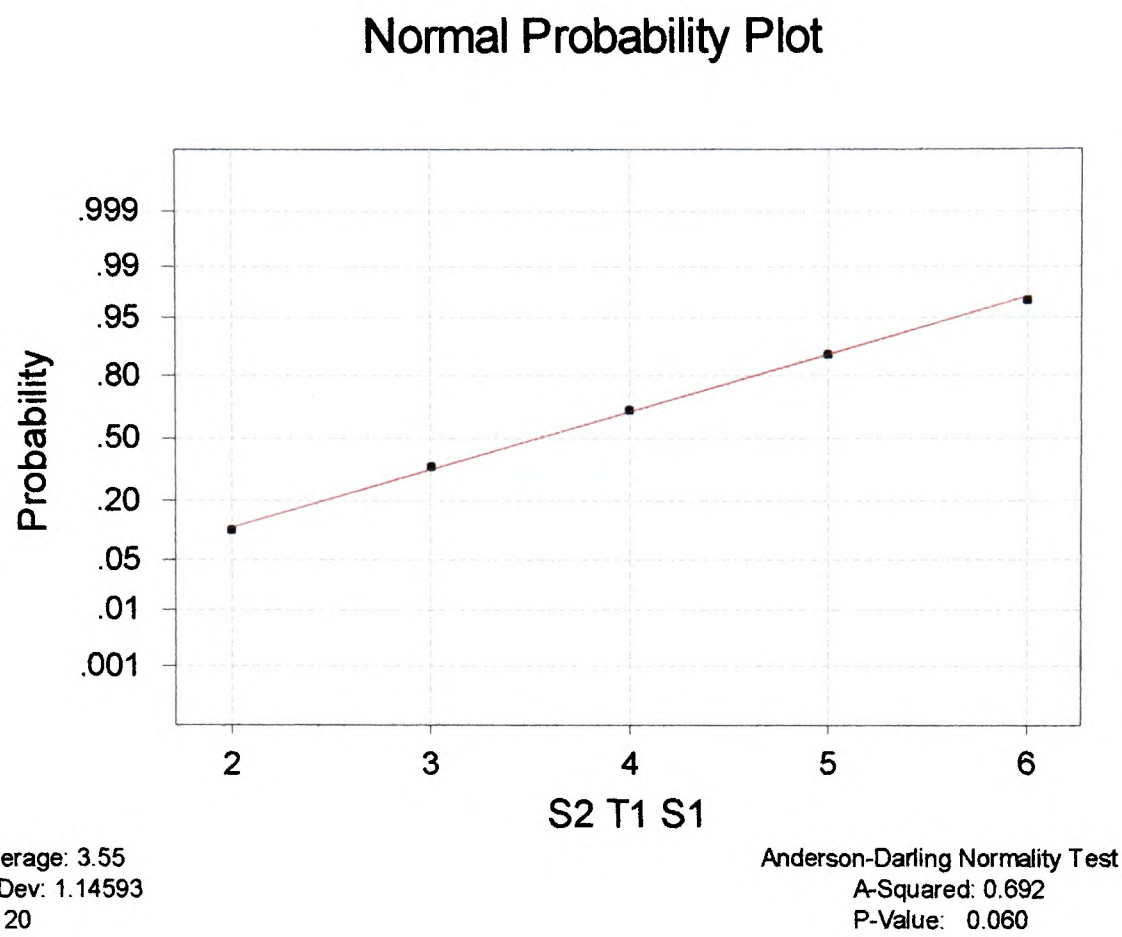
Figure D7 (c): Errors Highlighted (Formula View)

	B	C	D	E	F
02	BALANCE SHEET				
03	All units are £000s				
04					
05	Year				
06	1999/2000				
07		Net assets	7,246		
08		Total assets		63,081	
09		Fixed assets			56,742
10		Cash			3,251
11		Debtors			3,088
12		Total liabilities		55,835	
13		Debt			51,411
14		Creditors			3,122
15		Capital creditors			1,170
16		Tax creditor			132
17		Shareholders' funds	7,246		
18		Share capital		1,000	
19		P&L reserve		6,246	
20	2000/01				
21		Net assets	8,804		
22		Total assets		63,322	
23		Fixed assets			55,622
24		Cash			4,372
25		Debtors			3,328
26		Total liabilities		54,518	
27		Debt			51,411
28		Creditors			2,619
29		Capital creditors			136
30		Tax creditor			352
31		Shareholders' funds	8,804		
32		Share capital		1,000	
33		P&L reserve		7,804	
34	2001/02				
35		Net assets	5,008		
36		Total assets		60,458	
37		Fixed assets			54,481
38		Cash			2,375
39		Debtors			3,602
40		Total liabilities		55,450	
41		Debt			51,411
42		Creditors			2,541
43		Capital creditors			140
44		Tax creditor			1,358
45		Shareholders' funds	5,008		
46		Share capital		1,000	
47		P&L reserve		4,008	
48	2002/03				
49		Net assets	2,366		
50		Total assets		58,298	
51		Fixed assets			53,318
52		Cash			1,218
53		Debtors			3,762
54		Total liabilities		55,932	
55		Debt			51,411
56		Creditors			2,648
57		Capital creditors			144
58		Tax creditor			1,729
59		Shareholders' funds	2,366		
60		Share capital		1,000	
61		P&L reserve		1,366	

Figure D7 (d): Correct Version of the Model (Structured Layout)

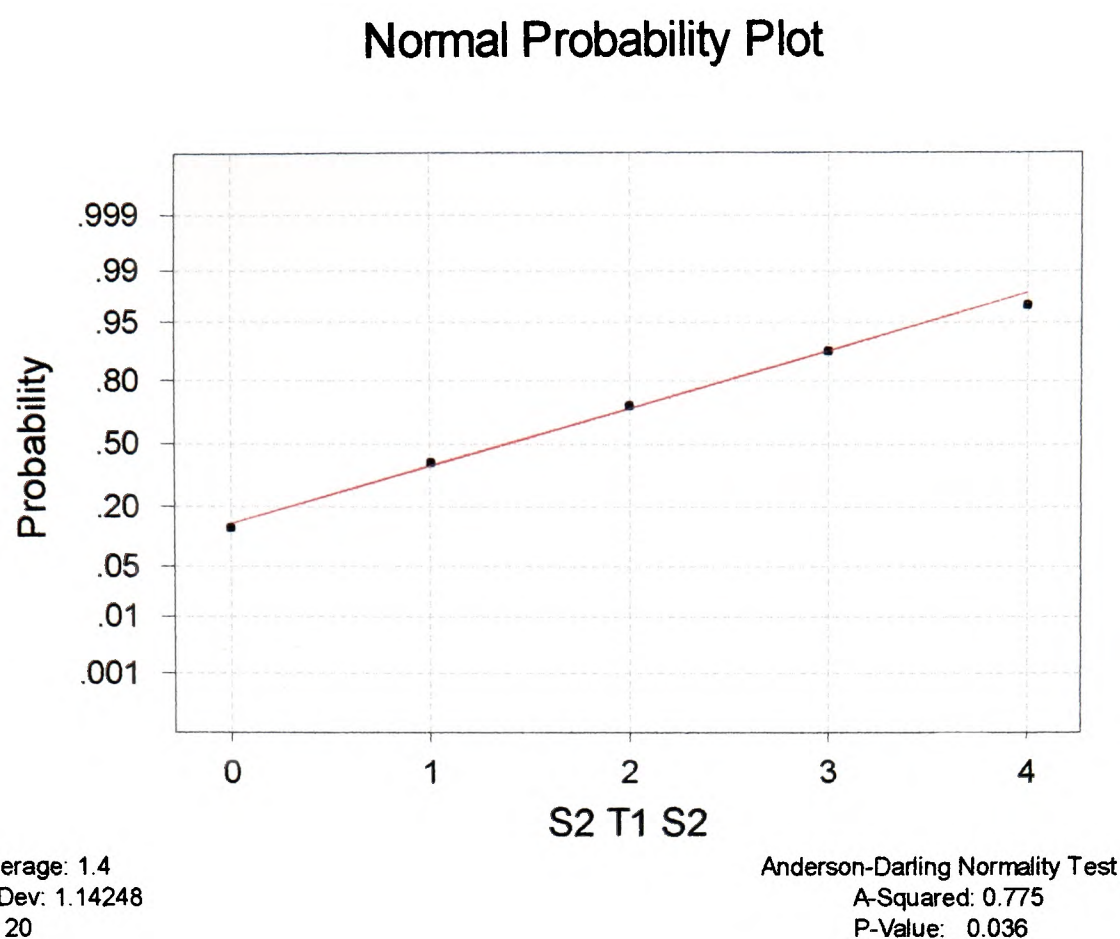
APPENDIX E
NORMALITY TEST RESULTS

Experiment 1



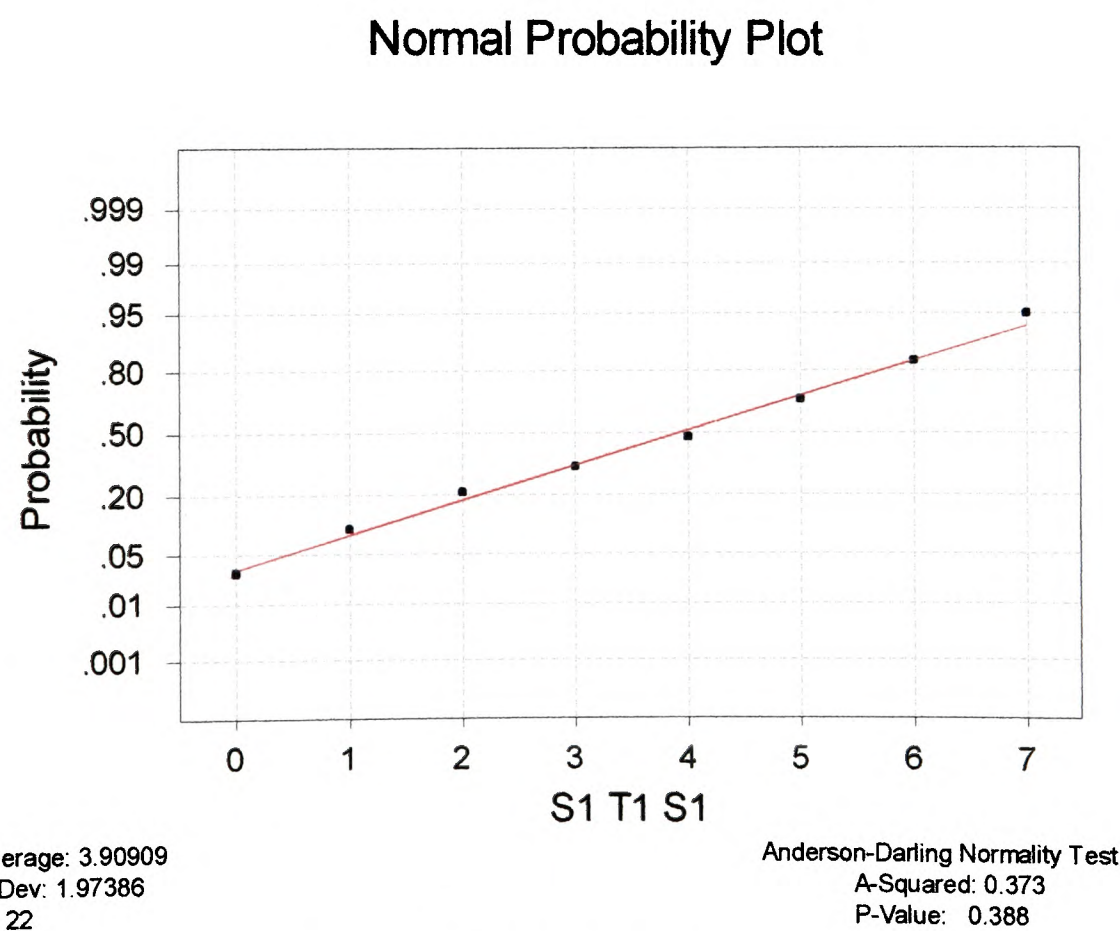
The P-Value is 0.060. It is greater than 0.05. Therefore, there is NO significant difference from Normal. The sample is from a normal distribution.

Figure E1 (a): Normality Test Result (in Minitab)



The P-Value is 0.036. It is less than 0.05. Therefore, there is significant difference from *Normal*.
The sample is NOT from a normal distribution.

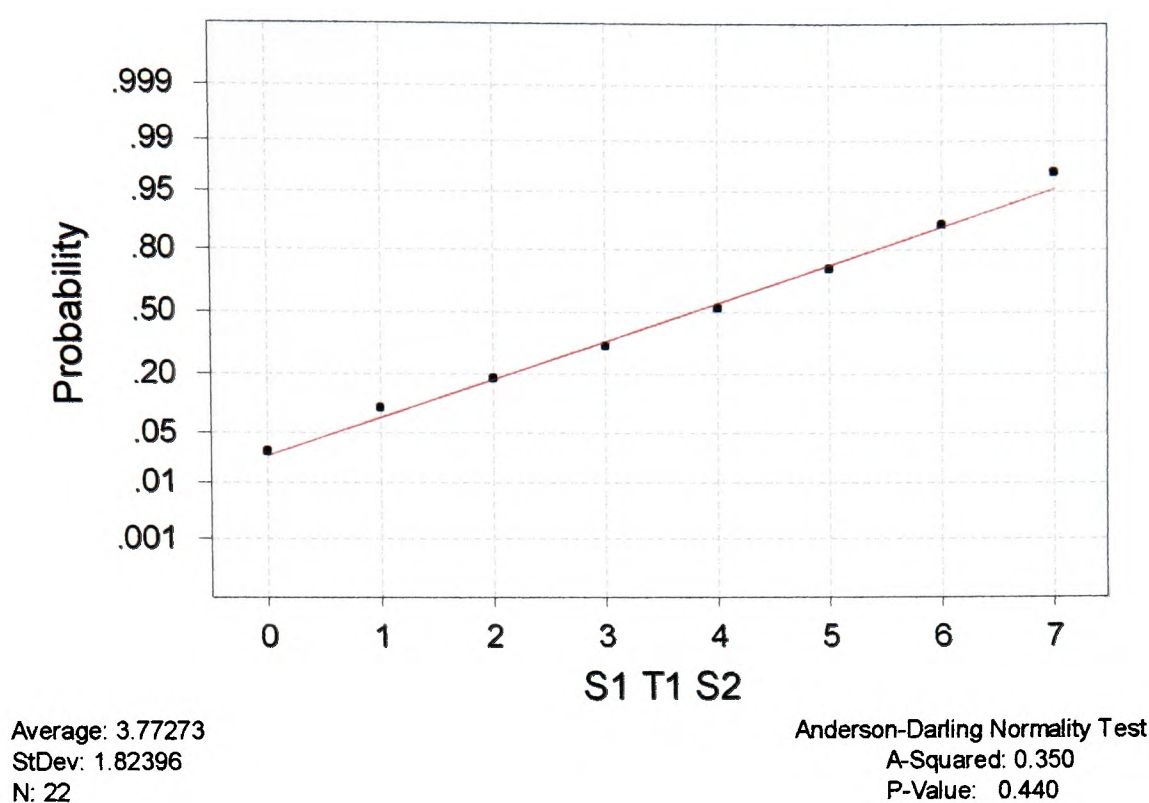
Figure E1 (b): Normality Test Result (in *Minitab*)



The P-Value is 0.388. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E1 (c): Normality Test Result (in *Minitab*)

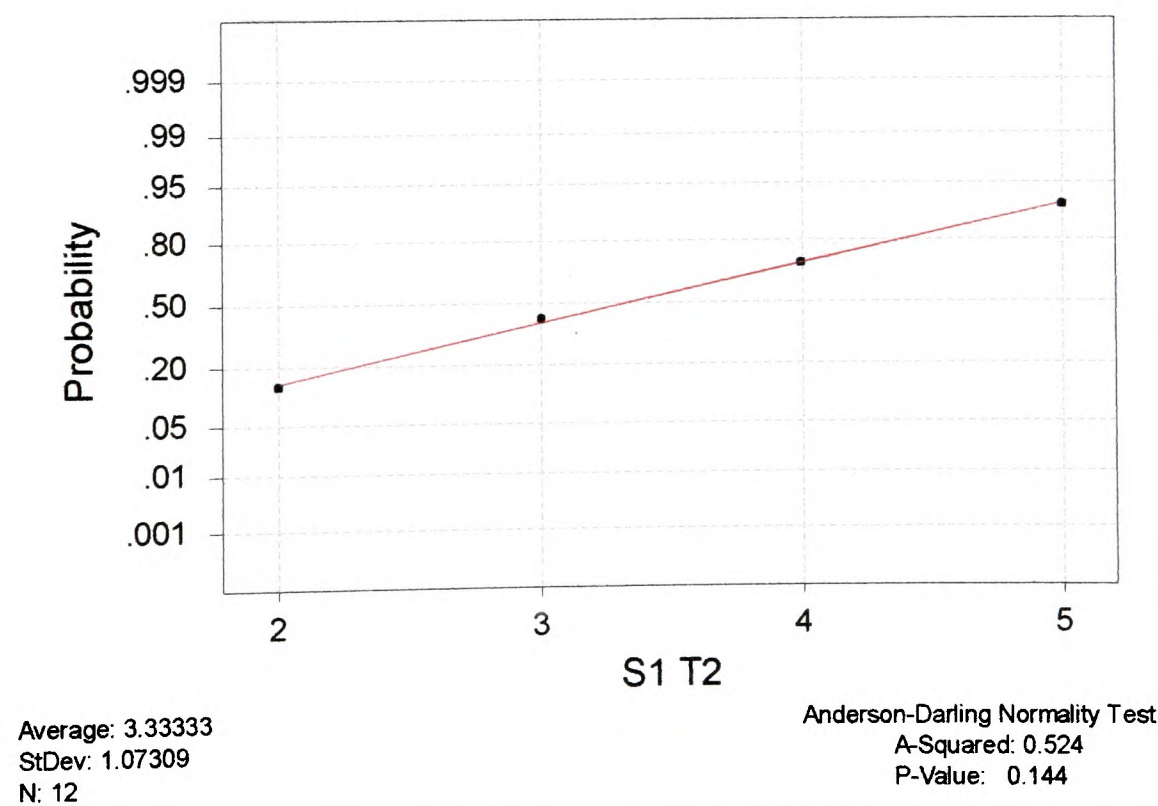
Normal Probability Plot



The P-Value is 0.440. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E1 (d): Normality Test Result (in *Minitab*)

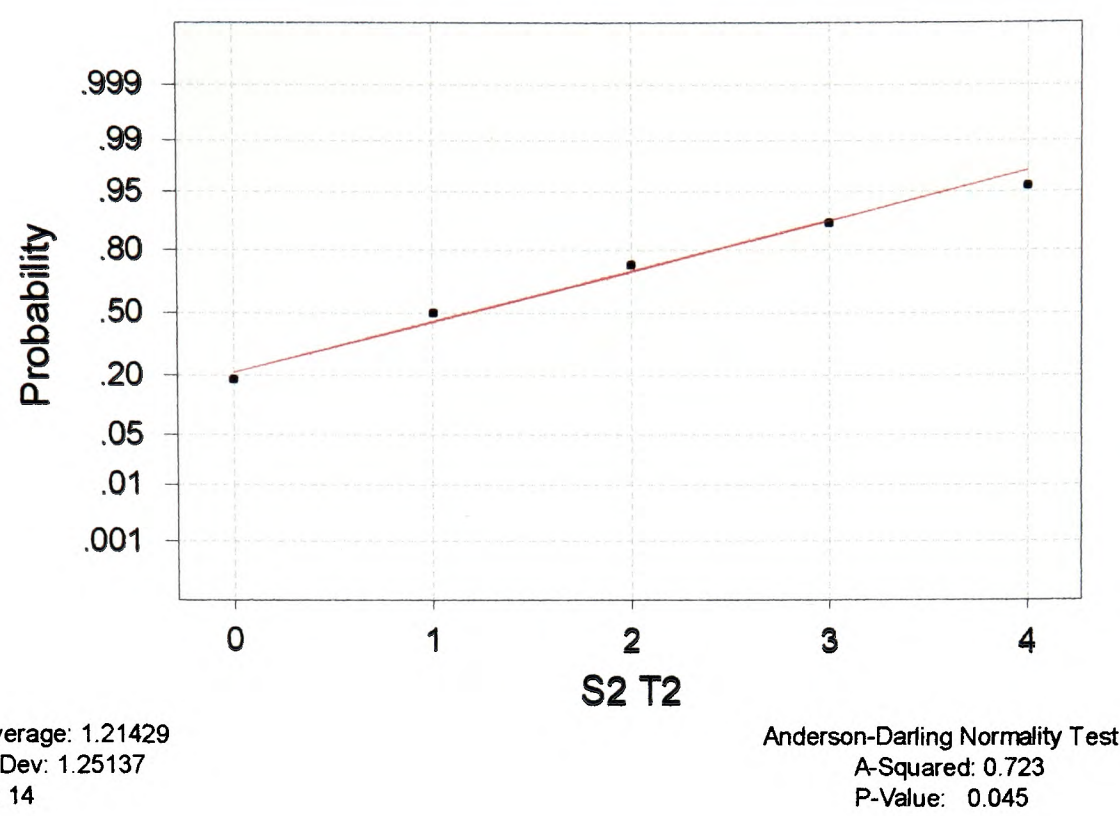
Normal Probability Plot



The P-Value is 0.144. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E1 (e): Normality Test Result (in *Minitab*)

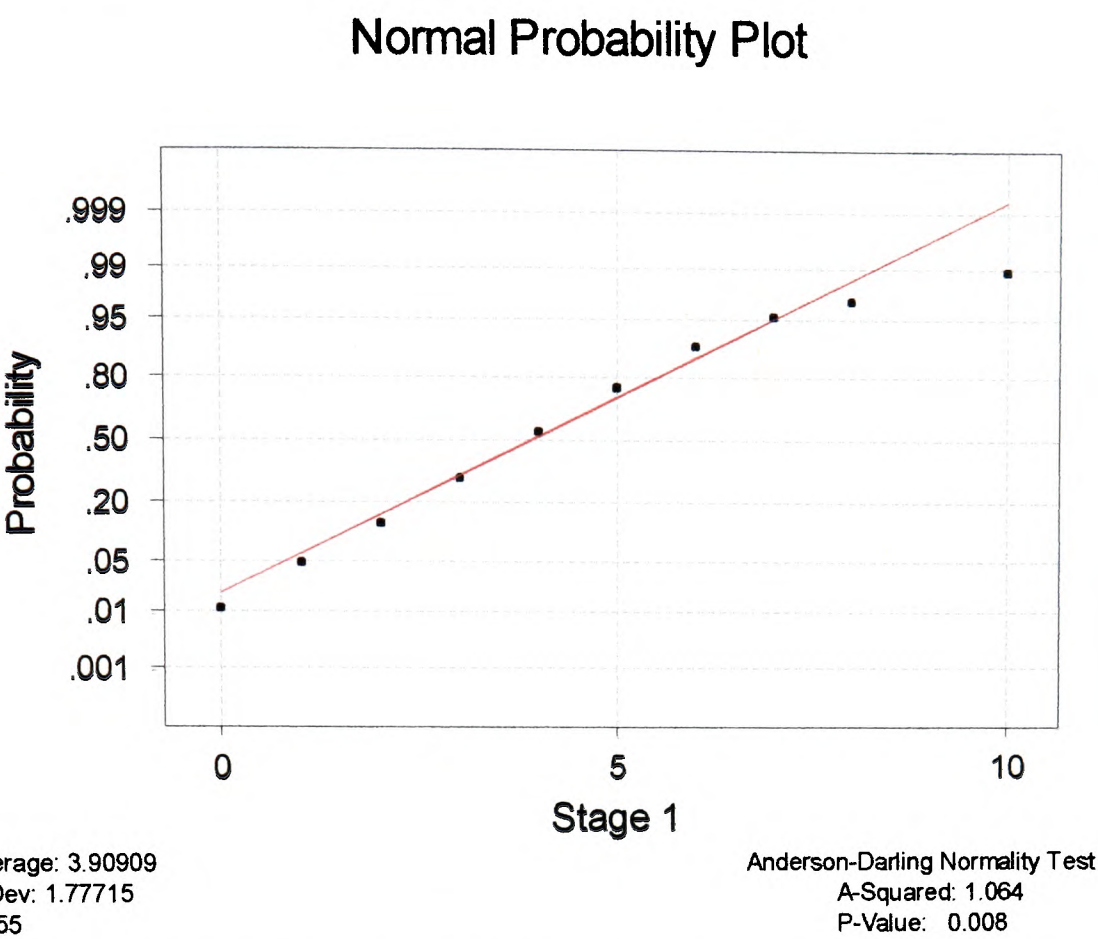
Normal Probability Plot



The P-Value is 0.045. It is less than 0.05. Therefore, there is significant difference from *Normal*.
The sample is NOT from a normal distribution.

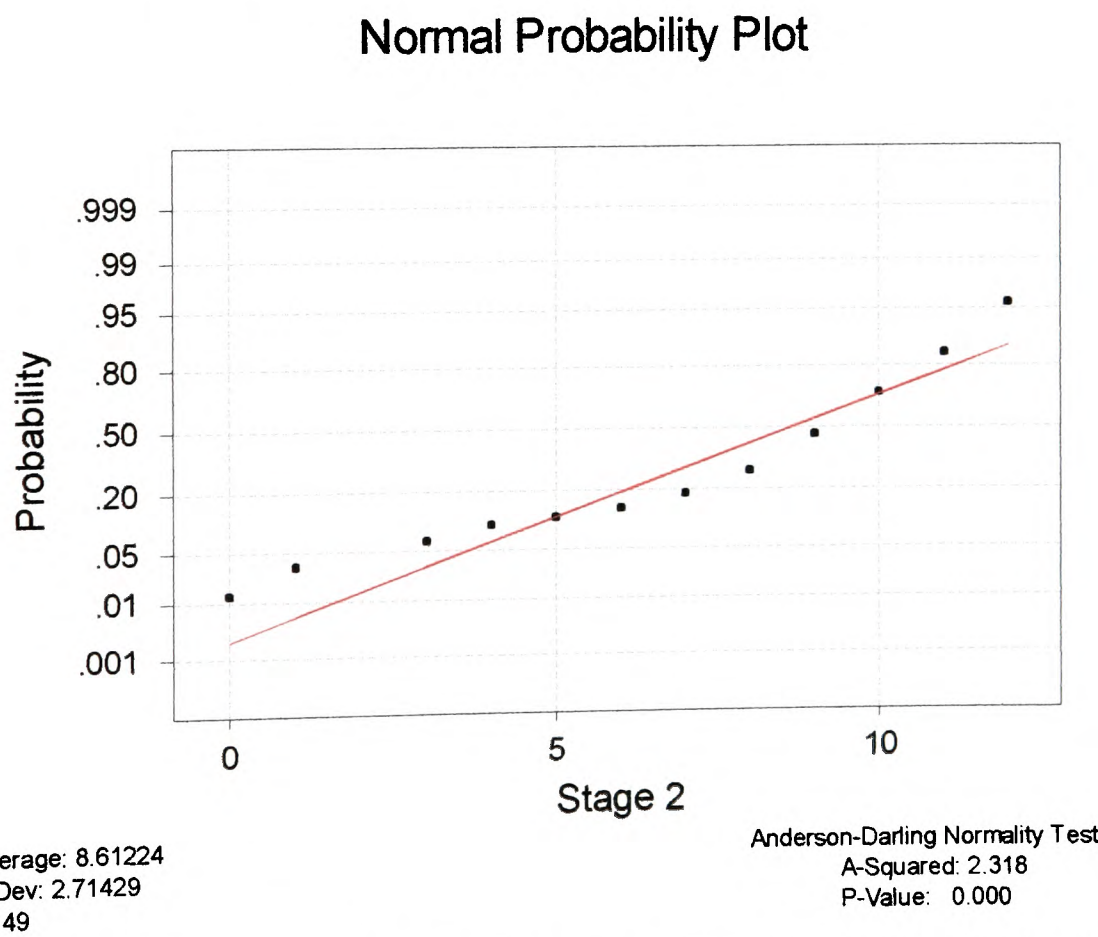
Figure E1 (f): Normality Test Result (in *Minitab*)

Experiment 2



**The P-Value is 0.008. It is less than 0.05. Therefore, there is significant difference from *Normal*.
The sample is NOT from a normal distribution.**

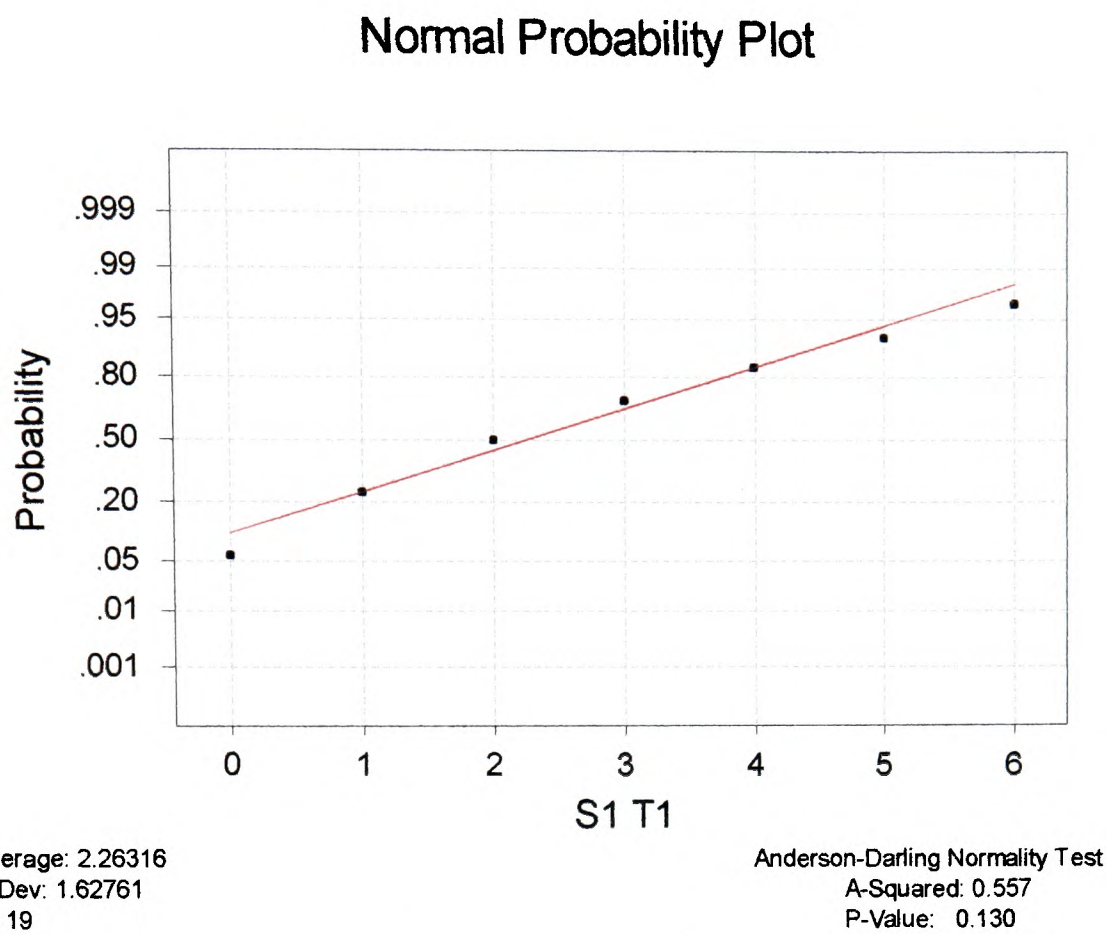
Figure E2 (a): Normality Test Result (in *Minitab*)



The P-Value is 0.000. It is obviously less than 0.05. Therefore, there is significant difference from *Normal*. The sample is NOT from a normal distribution.

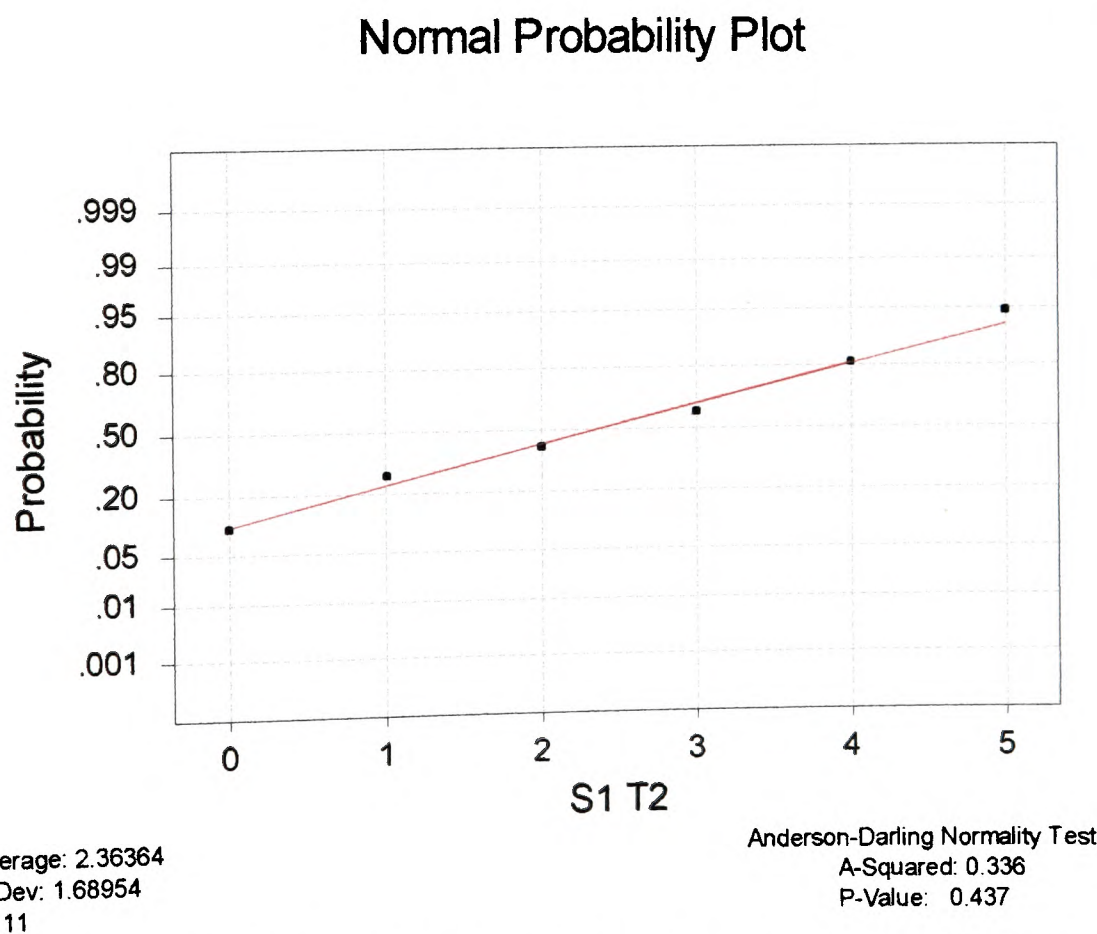
Figure E2 (b): Normality Test Result (in *Minitab*)

Experiment 3



The P-Value is 0.130. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

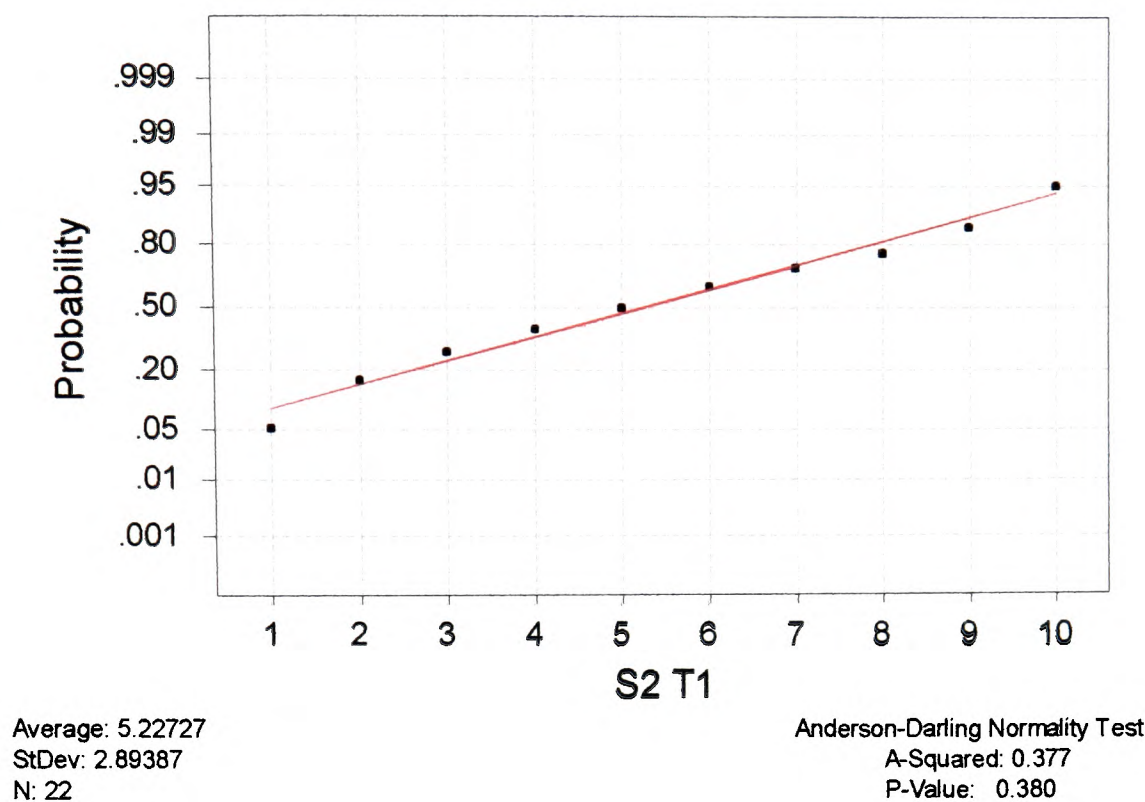
Figure E3 (a): Normality Test Result (in *Minitab*)



The P-Value is 0.437. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E3 (b): Normality Test Result (in *Minitab*)

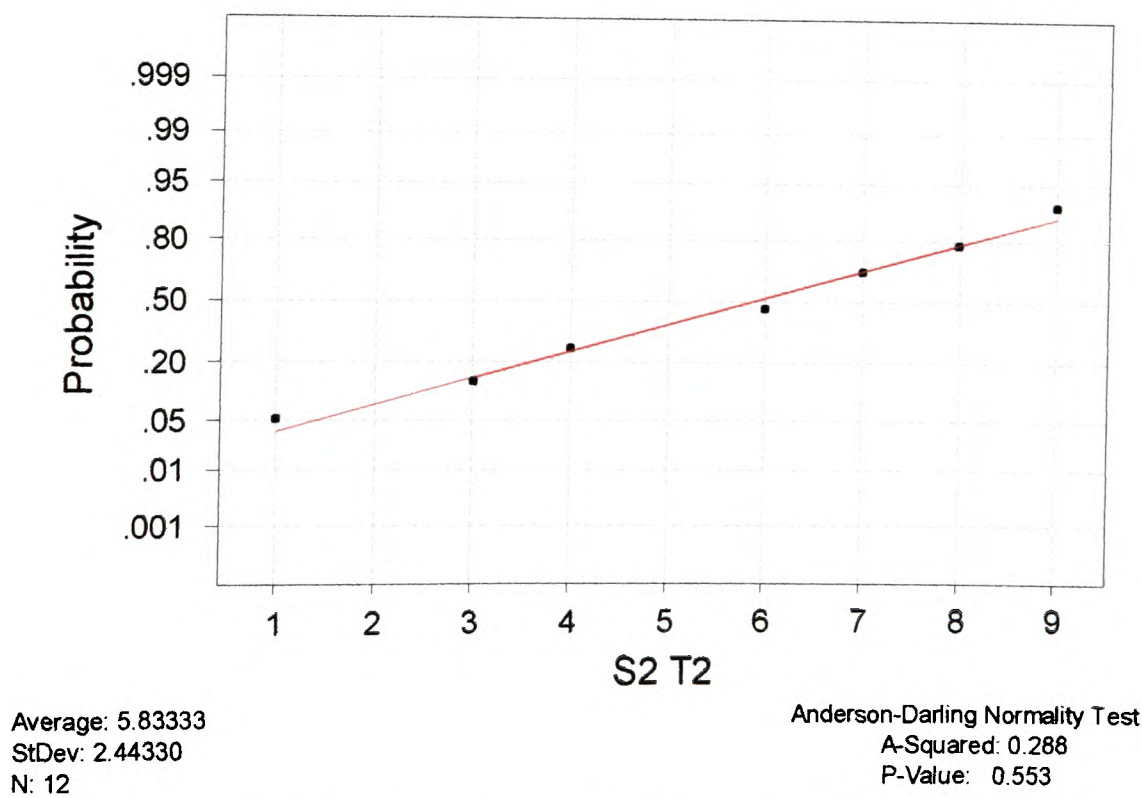
Normal Probability Plot



The P-Value is 0.380. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E3 (c): Normality Test Result (in *Minitab*)

Normal Probability Plot

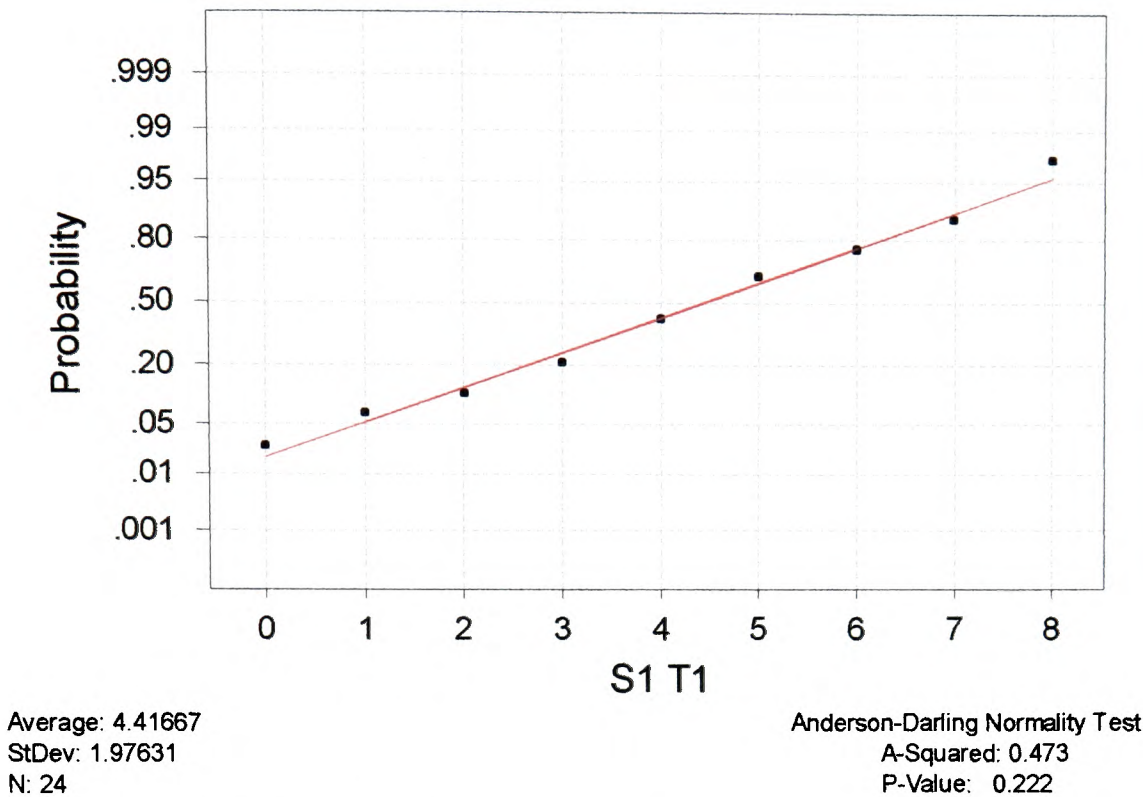


The P-Value is 0.553. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E3 (d): Normality Test Result (in *Minitab*)

Experiment 4

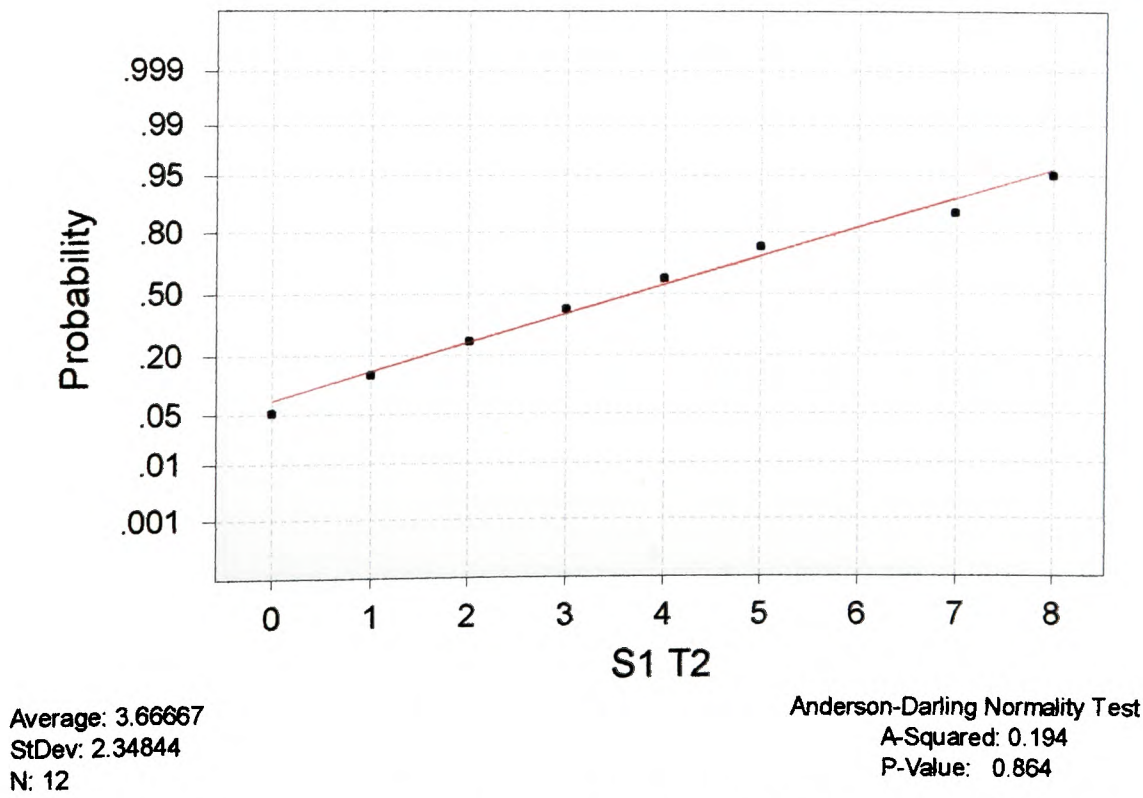
Normal Probability Plot



The P-Value is 0.222. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E4 (a): Normality Test Result (in Minitab)

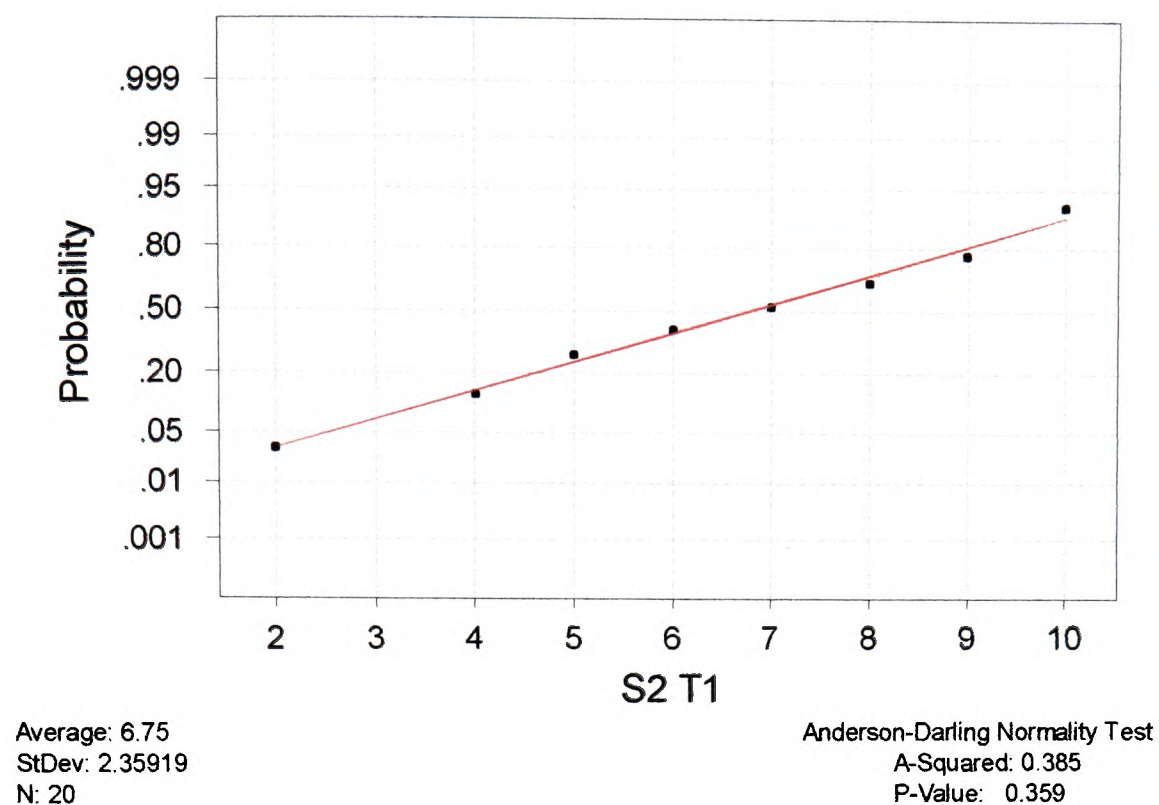
Normal Probability Plot



The P-Value is 0.864. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E4 (b): Normality Test Result (in Minitab)

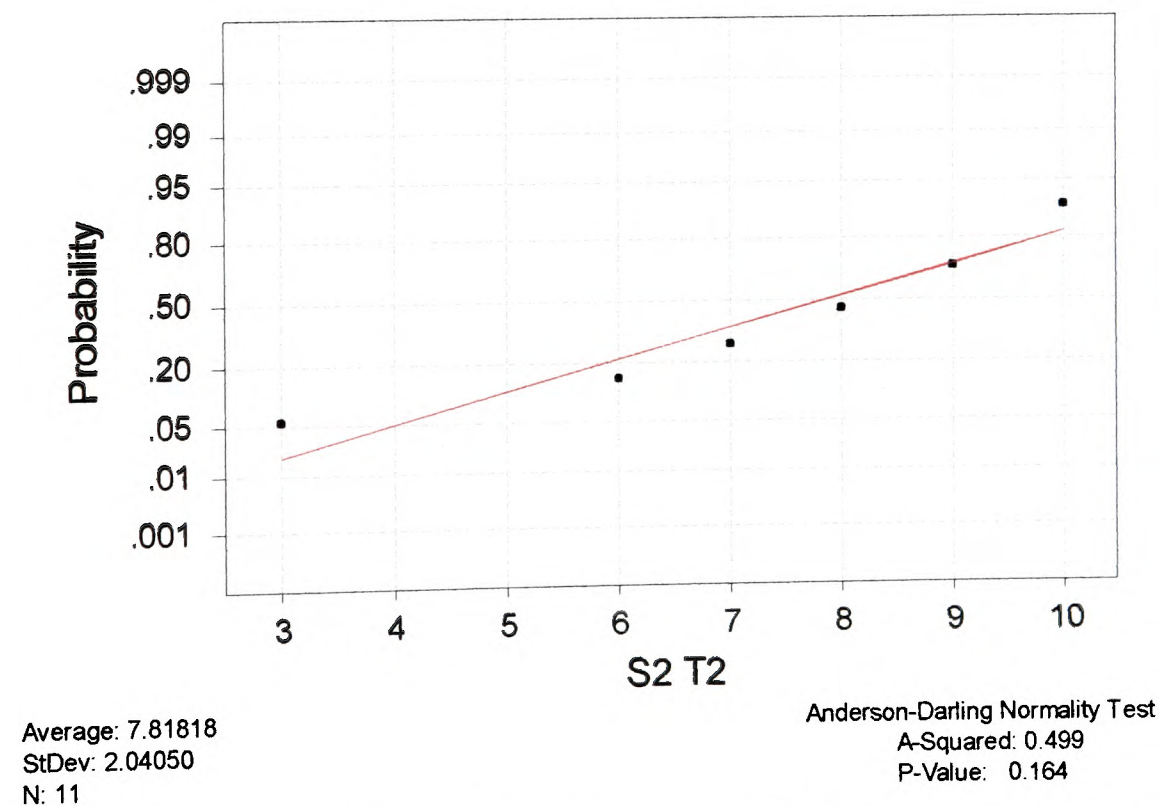
Normal Probability Plot



The P-Value is 0.359. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E4 (c): Normality Test Result (in *Minitab*)

Normal Probability Plot



The P-Value is 0.164. It is greater than 0.05. Therefore, there is NO significant difference from *Normal*. The sample is from a normal distribution.

Figure E4 (d): Normality Test Result (in *Minitab*)

APPENDIX F RAW DATA FROM THE EXPERIMENTS

Experiment 1

STAGE 1	Test 1	
	Session 1	Session 2
	<i>Without lesson</i>	<i>without lesson</i>
Subject Type	post-graduate students	post-graduate students
	Number of Errors Committed	
Subject 1	1	1
Subject 2	4	3
Subject 3	5	4
Subject 4	2	1
Subject 5	0	0
Subject 6	5	5
Subject 7	6	6
Subject 8	4	5
Subject 9	5	5
Subject 10	3	3
Subject 11	4	4
Subject 12	4	4
Subject 13	3	3
Subject 14	2	3
Subject 15	7	7
Subject 16	5	5
Subject 17	2	2
Subject 18	6	4
Subject 19	4	4
Subject 20	6	6
Subject 21	1	2
Subject 22	7	6
Mean	3.91	3.77
Standard Deviation	1.97	1.82

STAGE 1	Test 2
	<i>without lesson</i>
Subject Type	short course students
	Number of Errors Committed
Subject 1	3
Subject 2	2
Subject 3	3
Subject 4	4
Subject 5	5
Subject 6	2
Subject 7	3
Subject 8	4
Subject 9	4
Subject 10	3
Subject 11	5
Subject 12	2
Mean	3.33
Standard Deviation	1.07

STAGE 2	Test 1	
	Session 1	Session 2
	<i>without lesson</i>	<i>with lesson</i>
Subject Type	post-graduate students	post-graduate students
	Number of Errors Committed	
Subject 1	3	2
Subject 2	4	1
Subject 3	4	1
Subject 4	3	1
Subject 5	5	2
Subject 6	4	1
Subject 7	4	0
Subject 8	5	1
Subject 9	3	0
Subject 10	2	2
Subject 11	3	3
Subject 12	2	2
Subject 13	3	0
Subject 14	2	2
Subject 15	3	0
Subject 16	4	4
Subject 17	6	3
Subject 18	5	1
Subject 19	4	0
Subject 20	2	2
Mean	3.55	1.40
Standard Deviation	1.15	1.14

STAGE 2	Test 2
	<i>with lesson</i>
Subject Type	short course students
	Number of Errors Committed
Subject 1	0
Subject 2	1
Subject 3	1
Subject 4	2
Subject 5	0
Subject 6	3
Subject 7	2
Subject 8	1
Subject 9	4
Subject 10	0
Subject 11	2
Subject 12	0
Subject 13	0
Subject 14	1
Mean	1.21
Standard Deviation	1.25

Experiment 2

STAGE 1	<i>without lesson</i>
Subject Type	under-graduate students
	Number of Errors Detected
Subject 1	4
Subject 2	3
Subject 3	5
Subject 4	2
Subject 5	4
Subject 6	3
Subject 7	4
Subject 8	1
Subject 9	3
Subject 10	2
Subject 11	3
Subject 12	4
Subject 13	5
Subject 14	4
Subject 15	6
Subject 16	10
Subject 17	2
Subject 18	4
Subject 19	3
Subject 20	3
Subject 21	5
Subject 22	1
Subject 23	4
Subject 24	8
Subject 25	4
Subject 26	3
Subject 27	3
Subject 28	5
Subject 29	4
Subject 30	1
Subject 31	4
Subject 32	3
Subject 33	2
Subject 34	5
Subject 35	5
Subject 36	2
Subject 37	4
Subject 38	3
Subject 39	4
Subject 40	5
Subject 41	6
Subject 42	5
Subject 43	3
Subject 44	6

Subject 45	5
Subject 46	3
Subject 47	6
Subject 48	4
Subject 49	0
Subject 50	5
Subject 51	6
Subject 52	3
Subject 53	4
Subject 54	7
Subject 55	2
Mean	3.91
Standard Deviation	1.78

STAGE 2	<i>Without lesson</i>
Subject Type	under-graduate students
	Number of Errors Detected
Subject 1	10
Subject 2	9
Subject 3	6
Subject 4	12
Subject 5	11
Subject 6	8
Subject 7	1
Subject 8	8
Subject 9	10
Subject 10	0
Subject 11	4
Subject 12	10
Subject 13	9
Subject 14	12
Subject 15	7
Subject 16	11
Subject 17	9
Subject 18	10
Subject 19	9
Subject 20	7
Subject 21	10
Subject 22	9
Subject 23	8
Subject 24	12
Subject 25	3
Subject 26	11
Subject 27	11
Subject 28	9
Subject 29	7
Subject 30	9
Subject 31	10

Subject 32	8
Subject 33	9
Subject 34	11
Subject 35	9
Subject 36	10
Subject 37	3
Subject 38	10
Subject 39	9
Subject 40	12
Subject 41	9
Subject 42	7
Subject 43	10
Subject 44	5
Subject 45	11
Subject 46	9
Subject 47	9
Subject 48	8
Subject 49	11
Mean	8.61
Standard Deviation	2.71

Experiment 3

STAGE 1	Test 1
	<i>without lesson</i>
Subject Type	post-graduate students
	Number of Errors Detected
Subject 1	2
Subject 2	4
Subject 3	2
Subject 4	1
Subject 5	3
Subject 6	0
Subject 7	3
Subject 8	4
Subject 9	2
Subject 10	6
Subject 11	1
Subject 12	3
Subject 13	5
Subject 14	0
Subject 15	1
Subject 16	1
Subject 17	2
Subject 18	1
Subject 19	2
Mean	2.26
Standard Deviation	1.63

STAGE 1	Test 2
	<i>without lesson</i>
Subject Type	short course students
	Number of Errors Detected
Subject 1	3
Subject 2	3
Subject 3	0
Subject 4	1
Subject 5	2
Subject 6	5
Subject 7	1
Subject 8	0
Subject 9	4
Subject 10	3
Subject 11	4
Mean	2.36
Standard Deviation	1.69

STAGE 2	Test 1
	<i>with lesson</i>
Subject Type	post-graduate students
	Number of Errors Detected
Subject 1	8
Subject 2	2
Subject 3	2
Subject 4	9
Subject 5	3
Subject 6	1
Subject 7	8
Subject 8	10
Subject 9	6
Subject 10	5
Subject 11	1
Subject 12	6
Subject 13	4
Subject 14	5
Subject 15	6
Subject 16	9
Subject 17	4
Subject 18	7
Subject 19	2
Subject 20	4
Subject 21	3
Subject 22	10
Mean	5.23
Standard Deviation	2.89

STAGE 2	Test 2
	<i>with lesson</i>
Subject Type	short course students
	Number of Errors Detected
Subject 1	7
Subject 2	9
Subject 3	4
Subject 4	4
Subject 5	8
Subject 6	9
Subject 7	6
Subject 8	1
Subject 9	6
Subject 10	7
Subject 11	3
Subject 12	6
Mean	5.83
Standard Deviation	2.44

Experiment 4

STAGE 1	Test 1
	<i>without lesson</i>
Subject Type	post-graduate students
	Number of Errors Detected
Subject 1	7
Subject 2	4
Subject 3	1
Subject 4	3
Subject 5	4
Subject 6	8
Subject 7	5
Subject 8	4
Subject 9	7
Subject 10	4
Subject 11	3
Subject 12	6
Subject 13	5
Subject 14	7
Subject 15	5
Subject 16	3
Subject 17	5
Subject 18	4
Subject 19	4
Subject 20	7
Subject 21	0
Subject 22	2
Subject 23	5
Subject 24	3
Mean	4.42
Standard Deviation	1.98

STAGE 1	Test 2
	<i>without lesson</i>
Subject Type	short course students
	Number of Errors Detected
Subject 1	2
Subject 2	4
Subject 3	0
Subject 4	5
Subject 5	5
Subject 6	1
Subject 7	8
Subject 8	3
Subject 9	2
Subject 10	4
Subject 11	3
Subject 12	7
Mean	3.67
Standard Deviation	2.35

STAGE 2	Test 1
	<i>with lesson</i>
Subject Type	post-graduate students
	Number of Errors Detected
Subject 1	8
Subject 2	5
Subject 3	9
Subject 4	4
Subject 5	2
Subject 6	6
Subject 7	7
Subject 8	5
Subject 9	10
Subject 10	6
Subject 11	9
Subject 12	7
Subject 13	4
Subject 14	8
Subject 15	5
Subject 16	10
Subject 17	7
Subject 18	9
Subject 19	10
Subject 20	4
Mean	6.75
Standard Deviation	2.36

STAGE 2	Test 2
	<i>with lesson</i>
Subject Type	short course students
	Number of Errors Detected
Subject 1	8
Subject 2	7
Subject 3	9
Subject 4	10
Subject 5	3
Subject 6	6
Subject 7	8
Subject 8	10
Subject 9	7
Subject 10	9
Subject 11	9
Mean	7.82
Standard Deviation	2.04