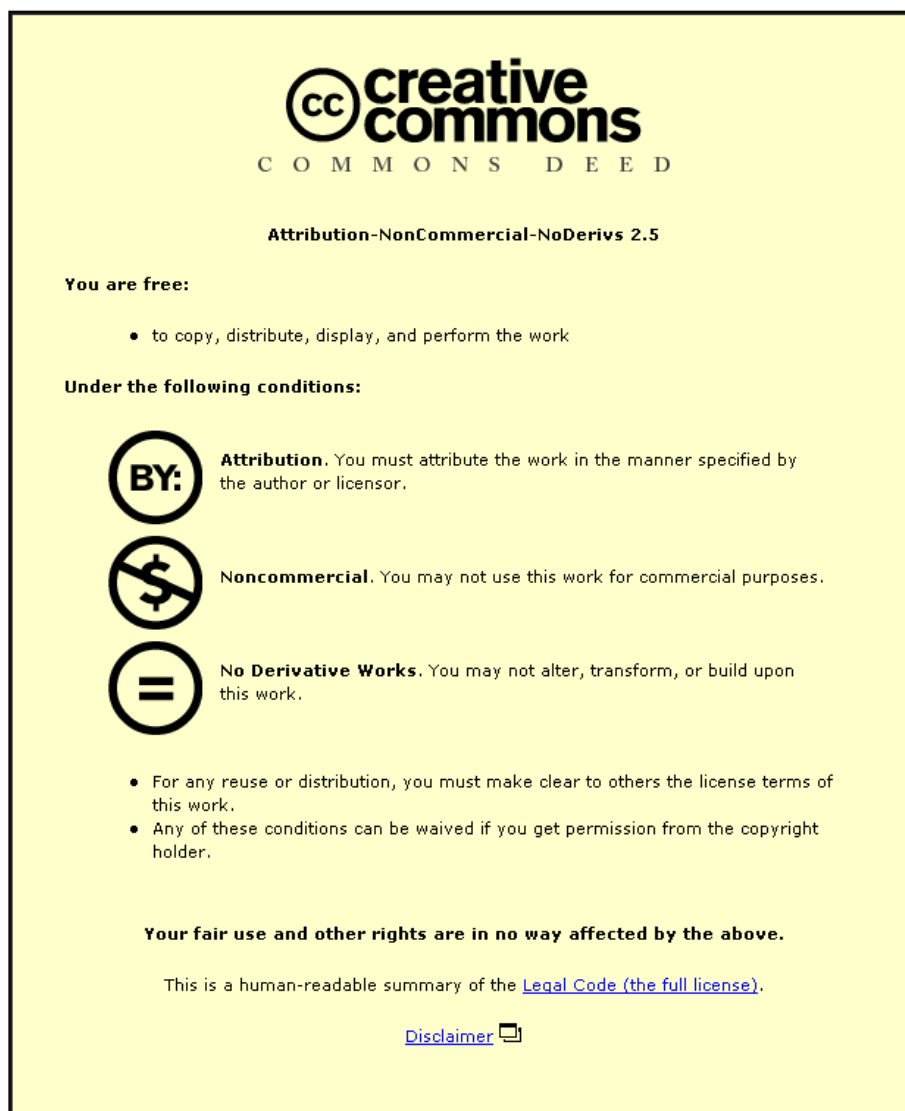


This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>



Pilkington Library

Author/Filing Title PAGONIS

Vol. No. Class Mark T

**Please note that fines are charged on ALL
overdue items.**

LOAN COPY

0402154592




**AN EFFICIENT VISUALISATION MECHANISM
FOR
COMMUNICATION NETWORK MONITORING
INFORMATION**

by
Antonis Pagonis

A doctoral thesis submitted in partial fulfilment of the requirements
for the award of Doctor of Philosophy of
Loughborough University

June 1999

© by Antonis Pagonis 1999

 Loughborough University Printed Library
Date Mar 00
Class
Acc No. 0402152489

M0001486CB

***Dedicated to my
family***

SYNOPSIS

Most, if not all, of the control and monitoring functions of modern communication network systems are implemented as software. Such software systems are of considerable size and complexity in order to cope with the pre-defined requirements. Effective presentation of results and control of network activities are part of the expected software specifications.

This thesis begins with an overview of the functions of Network Performance Management. Furthermore, it concentrates on Information Visualisation and its major role in the process of designing useful and usable software for communication network systems. Additional work follows on the author's novel idea of Figural Deformity Visualisation and its potential advantages with respect to network performance data pre-analysis. Figural Deformity (FD) can be a novel way of visualising multiple-variations by employing a single object and gradually deforming its original shape in order to represent relevant mapped parameters. Nevertheless, some research issues are considered before attempting to evaluate the effectiveness of the proposed visual mechanism by conducting user trials with qualified subjects. The thesis continues by presenting a prototype FD-Interface which has been used to collect the required experimental data. The relevant statistical analysis presents strong evidence in favour of the proposed type of user-interfaces. Finally, the work is concluded by presenting an initial implementation of the idea which has been evaluated in one of the largest companies of the British telecommunication industry.

ACKNOWLEDGEMENTS

Writing this page is normally a rather straightforward task by beginning the relevant paragraphs with banal phrases such as “I am indebted...”, “I would also like to thank...” etc. However, it is my belief that I should try to address my honest thanks to the various people with a different style not simply for the sake of being original but rather to show them that I took the time to work out a simple, sincere, warm and informal way of acknowledging them.

Therefore, first in the queue is my supervisor Dr. David Parish, the head of the High Speed Networks Research Group (HSN) in which I had the pleasure and honour to perform my research. His supervision, at least to my opinion can be characterised with a single word; impeccable.

Second in the queue to receive my virtual handshake is Dr. Ian Phillips. His help with some of the practical requirements of this research project has been greatly appreciated.

Multiple thanks should be addressed to each and every member of the HSN group at Loughborough for not only sitting my relevant trials but also for their interesting suggestions and discussions during various phases of this work.

Additional thanks must also be given to many academic and academic-related staff of the Department of Electronic & Electrical Engineering including the head of the Department Prof. Michael Kearney for sitting my lengthy user-trials.

Finally, I would like to say a big massive “Thanks” to my family for making possible my stay at Loughborough due to their encouragement and financial support.

PUBLICATION

Pagonis A. & Parish D. (1997),

“On the creation of a Highly Visualised Interface for Network Performance Data Pre-Analysis”, Proc. Fourth Communication Networks Symposium, The Manchester Metropolitan University, Manchester, 7 July, pp. 117-120.

Abbreviations & Acronyms

ANSI	American National Standards Institute
BT	British Telecom
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CASE	Computer-Aided Software Engineering
DFD	Data Flow Diagram
FD	Figural Deformity
FDV	Figural Deformity Visualisation
GIS	Geographic Information Systems
GPS	Global Positioning System
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HFI	Human-Factors Issues
IOG	Interaction Object Graph
IP	Internet Protocol
ISO	International Standards Organisation
LTM	Long Term Memory
MIB	Management Information Base
OODE	Object-Oriented Development Environments

OSI	Open Systems Interconnection
PC	Personal Computer
PDU	Protocol Data Unit
QOS	Quality of Service
SLA	Service Level Agreement
SMDS	Switched Multi-Megabit Data Service
STM	Short Term Memory
UDP	User Datagram Protocol
UI	User Interface
UIDS	User Interface Development Systems
UIMS	User Interface Management Systems
URL	Uniform Resource Locator
UTID	User-Tailored Interface Design

Contents

Synopsis	i
Acknowledgements	ii
Publication	iii
Abbreviations & Acronyms	iv
Contents	vi

Chapter 1

Introduction	1
1.1. Structure of the Thesis	5
1.2. Background Material	6
1.2.1. Network Performance Management	6
1.3. Objectives	9

Chapter 2

The Role of Visualisation in Communication Network Systems	11
2.1. Aims of Visualisation	11
2.1.1. The lack of Visualisation Methodologies	11
2.1.2. Relevant Questions for Successful Visualisation	12
2.2. Employing Visualisation for Effective Network Management	13
2.2.1. Characteristics of Human Perception	15
2.3. Innovations in Visualisation	17
2.3.1. Tree-Maps	17
2.3.2. Emotional Icons	19
2.3.3. Animated Icons	20
2.3.4. Resemblance, Exemplar, Symbolic and Arbitrary Icons	22
2.3.5. Flexible Filters	22
2.3.6. AlphaSliders	22
2.3.7. The Range-Selection Slider	25
2.3.8. Elastic Windows: A possible cure to 'Windowitis'	26

2.3.9. User Interface Management Systems (UIMSs)	28
2.4. Conclusion	30

Chapter 3

Creating a Highly Visualised Interface for Network Performance Data Pre-Analysis	32
3.1. Introduction	32
3.2. Network Monitoring	32
3.2.1. Visualisation and Current Network Monitoring tools	35
3.3. Designing Appropriate User Interfaces for Network Monitoring	39
3.3.1. Two Important Interface-Design Criteria	39
3.4. The Idea of Figural Deformity	41
3.4.1. Short-term and Long-term Memory	42
3.4.2. FD: The proposed alternative to conventional network monitoring screens	44
3.5. Designing an FD-Interface for Network Performance Data	45
3.6. Conclusion	54

Chapter 4

Research Issues in Evaluating the Effectiveness of FD-Interfaces	56
4.1. Introduction	56
4.2. The Scope of Empirical Research	56
4.3. Characteristics of Empirical Research	57
4.4. Components of Empirical Research	58
4.5. Statistical Analysis	59
4.6. Designing the FD-Interface Evaluation Coding Sheet	62
4.6.1. Scaling Human Responses	62
4.6.2. Coding Study Characteristics	64
4.6.3. General Questionnaire Construction Guidelines	64
4.6.4. FD-Interface Evaluation Coding Sheet	65
4.7. Minimising Observer-Caused Effects	68
4.8. Test Equipment and Environment Considerations	69
4.9. Some Sociological Considerations	69

4.10. Some Unanswered Questions	71
4.11. Conclusion	72
 Chapter 5	
Prototype FD-Interface For Pilot Testing	73
5.1. Introduction	73
5.2. A Brief Introduction to MATLAB	73
5.3. Program Presentation	75
5.3.1. Control-Group Assessment	75
5.3.2. Experimental-Group Assessment	82
5.4. Program Design Issues	88
5.5. Conclusion	94
 Chapter 6	
Experimental Data Analysis	95
6.1. Introduction	95
6.2. Variable –Z- analysis (Response to alarm situations)	96
6.2.1. Discussion	97
6.3. Variable –A- analysis (Object shape)	99
6.3.1. Discussion	100
6.4. Variable –B- analysis (Object colouring)	101
6.4.1. Discussion	104
6.4.1.1. Colour Perception	104
6.4.1.2. Subjective answers	106
6.5. Variable –C- analysis (Calibration method)	106
6.5.1. Discussion	107
6.6. Variable –D- analysis (Parameter-mapping)	108
6.6.1. Discussion	109
6.7. Variable –E- analysis (Number of objects per screen)	110
6.7.1. Discussion	111
6.8. Variable –F- analysis (D.O.F. per object)	111
6.8.1. Discussion	113
6.9. Conclusion	113

Chapter 7

Prototype FD-Interface For SMDS Alarm Station	114
7.1. Introduction	114
7.2. The SMDS Alarm Station	114
7.3. Alarms and Reports	115
7.4. Synoptic program presentation	116
7.5. From the user's viewpoint	120
7.6. Deciding on the course of action	121
7.6.1. Deliverables	121
7.7. Engineering the supplementary FD-Interface	122
7.7.1. Deciding on DOF, Object Shape and Parameter Mapping	122
7.7.2. FD-Object manipulation and related details	122
7.7.3. Relevant structure diagrams	124
7.7.4. Relevant code segments	125
7.7.5. The finished Interface	127
7.8. Prototype Evaluation	129
7.8.1. Users' reaction	130
7.8.2. Users' answers	130
7.8.3. Suggestions	132
7.9. Conclusion	132

Chapter 8

Conclusion	133
8.1. Discussions & Conclusions	133
8.2. Recommendations for further work	135
8.2.1. Detailed information Vs Speed of interaction	135
8.2.2. Combining FDV with existing visualisation innovations	135
8.2.3. Additional applications of FDV	135

References	137
-------------------	------------

Bibliography	146
---------------------	------------

Appendix –A	147
A.1. Introduction	147
A.2. Jackson Structure Charts for the Prototype FD-Interface	148
 Appendix -B	 163
B.1. Introduction	163
B.2. Source code for the prototype interface for pilot testing	164
B.2.1. fd_vs_bar.m	164
B.2.2. startnow.m	168
B.2.3. fd1.m	174
B.2.4. fdDraw.m	179
B.3. Source code for the telescopic antenna visualisation prototype	181
B.3.1. dokimi.m	181
B.3.2. Antenna.m	189
 Appendix –C	 192
C.1. t distribution critical values	192

CHAPTER

1

Introduction

Managing information from huge data warehouses in computerised environments has always been a significant issue in not only computer science, but in electronic engineering where the introduction of networked information systems and the frequent changes in the format and amount of the relevant data requires further scientific evolution to cope with the future challenges.

Improving computer performance has been the main objective of some developers but accelerating the relevant and necessary user interactions with the numerous user interfaces embedded as software components of today's computer networks, has been an overlooked dimension of the optimisation process.

Today's communication networks present complex multidimensional problems. Their composition consists of numerous heterogeneous elements examples of which are the different transmission media (satellites, fibre optic cables), the kind of data being transmitted (i.e. sound, video, data) and differing protocols. As a result of this situation, modern networks are extremely demanding in terms of software-requirements since there are numerous elements to be controlled each having many additional parameters to be specified. Practically many of the control and monitoring functions of communication network systems are implemented as software. Suffice it to say that such software systems are usually of enormous size and complexity in order to be able to satisfy the given requirements.

Improved transmission facilities and development of sophisticated protocols have dramatically reduced the traditional problems associated with early data networks [Ce97]. Network applications have managed to cope with the various data anomalies (such as data loss, ordering and duplication) which could in the past threaten their dependability and functionality and due to many technical innovations, real-time applications with multimedia streams are rapidly emerging.

However, with the vast increase in time-sensitive applications running across networks, new concerns begin to appear and network management has been given a new substantial role to play. Communication networks present serious challenges to the service providers whose aim is to provide an acceptable performance to the users by managing their resources in the most optimised way.

The development of massive internetworking has lead to the formalisation and standardisation of critical functional areas of network management such as network configuration, accounting for service utilisation and customer billing, security issues, fault detection and network performance measurement. However, some internetworking aspects involve human interaction in the User domain and not exclusively in the Expert domain which presumably consists strictly from software and electrical-electronic engineers in the current technological context.

Consequently, HCI issues could become relevant to the subject-matter of Communication Networks since their proper consideration can result to significant improvements and benefits. Some HCI issues [Pr94] are briefly analysed below:

- **Organisational Factors** such as training, job design, and roles can inevitably play a dramatic role on the overall operation of a computer network. Consider a case where a network operator with inadequate training is expected to monitor a heavily loaded network and take appropriate action. Even worse, due to inefficient job design, the same person might be charged with extra obligations such as maintenance of particular network components without possessing the required technical skills or heavily specialised knowledge.

- **Environmental Factors** such as noise, heating, lighting and ventilation which can impair human performance with possibly unpredictable results.
- **Comfort Factors** such as seating and equipment layout can possibly result in health and safety factors such as stress, possible headaches and musculo-skeletal disorders.
- **Constraints** such as costs, budgets, timescales, staff size, quality and quantity of equipment along with building structure can affect the overall network operation and the quality of its functionality.
- **User Issues**, which can be analysed based on cognitive processes and capacities relevant to motivation, enjoyment, satisfaction, personality and experience level.
- **User Interface**, which is the only network component which allows user control and user-input into the system.

From this brief analysis, it becomes obvious that many of the mentioned factors can correlate with each other positively or negatively. As a simple example, cost constraints may dictate [Sli99] a particular type of computer screen which can lead to user dissatisfaction, therefore to lower motivation and enjoyment and presumably to decreased productivity.

The large difference between the traditional software engineering “waterfall model” and HCI design is based on three points which consist of the premises of the HCI model [Pr94]:

- The design must be **user-centred** and users must be allowed to participate and influence the design process

- Contribution to HCI design through **interdisciplinary** knowledge and expertise is the most basic condition for a successful outcome
- The whole process must be **highly iterative**; continuous testing of the design to meet user's requirements is a crucial part of the design process.

On the other hand, the conventional “waterfall model” is (at least theoretically) linear and the requirements analysis and definition once concluded are immediately processed and converted into a design which is coded, tested and then maintained for the rest of its life. According to this traditional approach design stages are totally independent from requirements analysis and iteration is possibly performed only at the last step of integration and system testing.

The advantages of the “Waterfall model” are mainly in the fast progress which is promised due to the lack of numerous time-consuming user-trials and continuous testing of the design. However, user-satisfaction remains an open issue and is not always of primary importance since users are not allowed to express an opinion, at least not before the stage of implementation.

The area of Network Monitoring has been one of the areas where HCI issues have not yet become of primary, if any, importance and the impact of this delay starts to become obvious after even a little market research on network monitoring tools. There are many instances where expensive network monitoring software with promising and “highly visualised” UIs (as the creators put it) are using text-based screens to present network data. In such instances users are taken back to the “golden” paper and pencil era and are presented with tables of plain numbers and IP-Addresses in order to create reports on the network statistics. In the best possible scenario, simple bar graphs, histograms and, rarely, conventional tree-structures are employed to deliver “high visualisations” for “high-end functionality” implying that such visual tools are the state-of-the art.

Therefore, a question arises: “Could other techniques be employed to deliver novel UIs which could in return offer truly novel and effective visualisation for network performance management in particular?” The answer could be “Yes”.

Section 1.1. Structure of the thesis

The thesis is divided into eight chapters. Each chapter has been produced, so as to be as self-contained and self-explanatory as possible whereby dependence on previous chapters is minimised.

Chapter Two attempts to expand up on the role of Visualisation in Communication Network Systems, whilst introducing some of the major advancements in highly visualised UIs.

Chapter Three presents the novel idea of Figural Deformity Visualisation and provides some theoretical analysis of HFI.

Setting up the research scenario for implementing FDV in network performance monitoring is attempted in Chapter Four. The contents of this chapter include a brief analysis of some research issues in evaluating the effectiveness of the method.

Chapter Five contains the description of a prototype FD-Interface designed for pilot testing.

Chapter Six discusses the results obtained after concluding the relevant user-study and processing the gathered data. Some simple conclusions are drawn in order to establish the basic theoretical foundations behind FDV.

Chapter Seven presents a practical implementation of the idea on the BT-SMDS Alarm Station Prototype at BT’s Walsall Operation Centre.

Finally, general conclusions maybe found in Chapter Eight which also discusses possible recommendations for future work.

The remainder of this introduction briefly reviews some background material.

Section 1.2. Background Material

This section briefly reviews some background material; parts of which are often referred to in the main body of the text.

1.2 .1. Network Performance Management

The ISO / ANSI standards committee has classified all relevant Network Management issues into six categories [Ku94] which are namely: 1) *Configuration Management*, 2) *Fault Management*, 3) *Performance Management*, 4) *Security management*, 5) *Accounting Management* and 6) *Directory management*.

The Performance Management Standard uses ISO 10064-11 to define the requirements and criteria for measuring network performance [Bla92]. The same standard, also, defines many parameters relevant to workload, throughput, resource waiting time, response time, propagation delay, availability and Quality of Service (QOS) changes. The performance activity is modelled as a *monitoring and tuning* protocol in the sense of continuously monitoring resources to assess system performance, adjusting measurement criteria and finally determining whether performance is indeed satisfactory. Furthermore one of the important features of any network management system is alarm reporting [Bla92].

Following a rather macroscopic view, performance management is organised around monitoring, analysis and tuning functions. Black [Bla92] suggests the following diagram to demonstrate the organisation of the performance management functions:

Performance Management

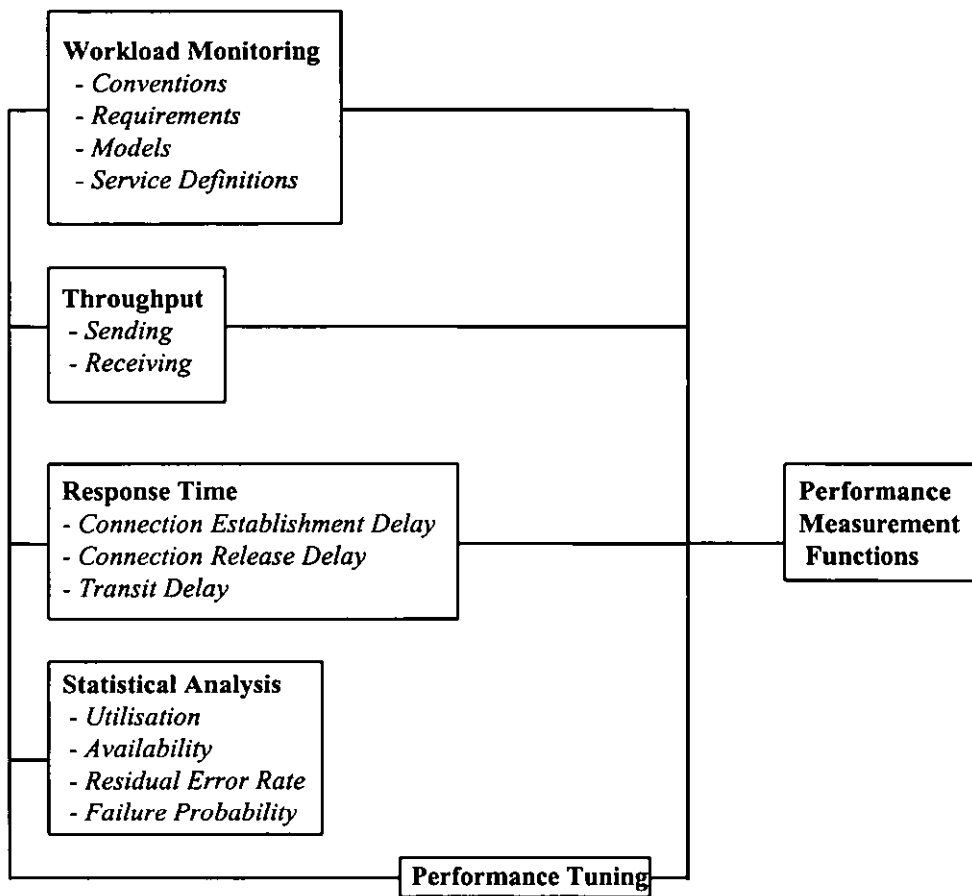


Figure 1: Performance Management Functions [Bla92]

The **Workload Monitoring function** is aimed at monitoring the performance and measuring the workload of a managed object. To put it in Black's words, the monitoring model is organised around *requirements*, *models* for monitoring and *service definitions*. Such requirements specifications are organised around:

- Early warning workload
- Early warning workload clear
- Overload
- Warning rejection

- Warning rejection clear

The monitoring function specifications are organised in relation to:

- Workload model
- Workload threshold
- Workload clear threshold
- Overload threshold
- Overload model
- Load threshold
- Loss threshold
- Loss clear threshold

Lastly, the service definition specifications are organised with respect to the following:

- Report workload alarm service
- Report loss alarm service

It is expected that early warnings are operating in relation to values potentially approaching each particular threshold value. Therefore a specific degree of tolerance is usually assumed when deciding the required threshold. Although it is not intended any further analysis on the workload monitoring function, the reader should note the importance of the *overload function* which is used to report an alarming situation for a managed object.

The **Throughput monitoring function** measures the throughput on a communication circuit or network node. There are two kind of throughputs: The **Sending** and the **Receiving** throughput. Both of these are defined as the ratio of successfully transferred PDUs within a sequence of data units provided at the maximum rate to the time between the first and last request for the unit transfers. A major assumption is needed though, according to which all measured units are transmitted without any errors.

The **Response time** monitoring function is designed to assess the response time of a communications node or a network. Response time is defined as the time between the issuance of a request and the receipt of an indication (single-way) or the receipt of a confirm (2-way) [Bla92]. The **connection establishment delay** is the time required to establish a connection starting at the time a connection request is sent to the time a corresponding confirmation is received. The **connection release delay** defines the time taken to release the connection. The **transit delay** measures the delay from the sending to the receiving end users.

The **Statistical analysis** function is a large group of activities having the primary purpose of monitoring records and determining performance of several entities within an OSI managed network. Within this function are other QOS operations, such as *Utilisation, Availability, residual error rate*, and failure probability.

Lastly, the **Performance tuning** function is used to measure the performance of the queueing mechanism. The function measures the waiting time (the time spent in waiting for a service), the service time (the time spent to process the relevant activity) and the inter-arrival time (the time between the arrival of events at a queue). The importance of this particular function is great due to the fact that its role is to reassure a successful synchronisation among all performance management modules. Especially in the case of real-time multimedia communication, the potential problematic operation of this function can have a negative impact on the whole model. On the other hand, if the function is operating satisfactorily an overall smoother operation can be expected, since the temporal relationships associated with real-time application traffic are preserved and are useful to the receiving host [Ce97].

Section 1.3. Objectives

The main aim of the research is threefold:

- To investigate the significant role of visualisation in the context of Network Performance Management.
- To present a novel way of visualising network monitoring information.
- To demonstrate the merits of the particular novelty with a practical verification of the theoretical claims.

CHAPTER

2

**The Role of Visualisation
in
Communication Network Systems**

Section 2.1. Aims of Visualisation

According to Robertson [Ro91], underlying the concept of visualisation is the idea that an observer can construct a *mental model*, which represents data attributes in a definable manner with the help of its visual attributes. Highly visualised data presentations and management tools are becoming more widely incorporated into information systems. Furthermore, Robertson observes that there is a great need for a range of data representation options which will cater for different types of data, interpretation goals and contextual requirements of the various systems.

2.1.1 The lack of Visualisation methodologies

Choosing the best way to represent data requires the correct matching of the type of information we wish to visualise with the potential data representation which, it is hoped will possess all these “ingredients” which will manage to convey the information

correctly. The verb “hoped” bears a very important meaning in the previous sentence. Although researchers such as Haber & Wilkinson have helped establish the aims of visualisation, the development of methodologies remains limited [Ro91]. It might be true that much work on data displays has been done by following existing guidelines, yet this set of guidelines do not constitute a proper methodology on which we can base automatic data display generation. Most guidelines lack a comprehensive and systematic treatment and are relevant only to specific display issues such as colour appearance or graphical rules applied to precisely defined graphical languages[Ro91].

2.1.2 Relevant questions for successful Visualisation

Bearing the above in mind, several questions can be raised:

- *What are the types of mental models that most effectively carry various kinds of information?*
- *Which definable and recognisable visual attributes of such effective models are most useful to pass specific information either independently or in combination with other attributes?*
- *How can the programmer effectively fortify chosen mental models in the mind of the observer / user?*
- *How can we generate instructions on selecting appropriate models and their attributes for a user-interface designer?*

Although such questions look quite new, artists and scientists have long considered them; if not explicitly, certainly implicitly. One potential approach to answer them successfully suggests: 1) to use simple and understandable models such as 3D scenes, 2) to represent data variables by the most obvious visual attributes of the selected models and 3) to induce them in the observer’s mind by using graphic scene simulation techniques [Ro91]. This approach is called the *Natural Scene Paradigm* and it is based

on the human ability to look at a particular scene and gain an immediate appreciation by isolating the recognisable properties.

In theory such approaches to the previous questions seem sufficiently promising, but the problem of the intangible nature of the visualisation issue still remains. [Eli96]. According to Eliot, the ultimate objective of Visualisation is to assist users through *exploration, navigation* and *browsing* in order to reach the appropriate data from a data world currently growing in size (but also declining in average information content) from which they will collect only the relevant information in order to realise timely, informed decisions while gaining increased, shared knowledge.

Visualisation has a fundamental role to play, providing newer and even more powerful tools in order to help users understand and solve complex, data-intensive problems more pleasantly and efficiently.

Section 2.2. Employing Visualisation for Effective Network Management

From the previous brief analysis on Network Management (see Chapter 1), the important role of software and its capability to satisfy all predefined requirements starts to become apparent. Modern network management systems rely heavily on their software and especially on forms in their user-interfaces. The interfaces reflect the complication of the network hardware components but unfortunately they provide little support for guiding users through tasks. According to Kumar [Ku94] there is a scarcity of useful graphical visualisation and decision-support tools. Areas of network management such as network configuration and network performance are in great need of highly visualised software due to the fact that the activities involved are falling in the “chaotic” area of **information exploration**.

Information exploration should be a joyous experience but many commentators talk of information overload and anxiety [Shne96]. Many network management systems take a database-centric approach (i.e. the network is represented in a database called the Management Information Base -MIB-). Writing software for a Network Configuration Management System or Network Monitoring Management System can be a very challenging problem because of the many parameters that need to be entered into the MIB via the relevant user-interfaces. Network operators work under pressure to keep the network running 24 hours a day under all circumstances and current user-interfaces consist typically of hundreds of forms that need to be filled or read for successful network operation. As a result, users experience problems with piles of forms in the absence of effective visual organisation and information visualisation tools. Besides, in order to perform management operations in a heterogeneous environment, conventional user interfaces are not robust enough to satisfy the new challenges [Lia91]. Depending on the particular layered structure, the information presentation might not be independent of other system components and therefore integration of other system tools might not be easy, if possible at all.

On the other hand, integrated network management becomes more difficult as time progresses, due to the rapid growth of computer networks, both in complexity and/or heterogeneity. Although there are a great variety of network management tools offered by various vendors, these are usually intended for their own network products [Lia91]. As an organisation possesses sub-networks from various vendors, it has a number of independent subsystems each of which requires a different management system. In this case, the lack of interoperability among the various management tools becomes the key problem of the particular system operator who clearly is in great need of more visualised interfaces that are able not only to filter the network mechanics but also focus on the *raison d'être* of each operation while enhancing the user's perceptual capabilities.

At this point, the following very simple, yet crucial question can put things in the right perspective:

“What are the main characteristics of human perception?”

Ignorance to the previous question might be the cause of the labyrinth that every network operator falls into, at least the first time he attempts to perform his duties.

2.2.1 Characteristics of Human Perception

Preece [Pr94] suggests that the following key points describe accurately the basic characteristics of human perception:

- *Attention can be focused or divided:* all the events that the human mind attends in the world are filtered in order to understand the world.
- *Cognitive processes can be controlled or automatic.* Controlled processes have limited capacity and require attention and consciousness, while automatic processes are not affected from limited brain capacity and mostly do not require attention (an example of controlled process is the poor performance of someone who tries to learn typing against a skilful secretary whose performance after prolonged practice is automatic and effortless).
- *Memory is limited:* the more meaningful names and icons are, the more likely they will be remembered. For example, in a study relevant to the number of commands UNIX experts can recall, Draper [Pr94] concluded that they were able to recall very few. His scientific explanation was that the experts are experts in discovering information as and when only it is needed. Their skill is to remember where to find the information rather than memorising the information itself. Moreover, people

tend to use their memory very selectively. Even though many objects can be extremely familiar to their owners, useless details are rarely remembered.

- *For successful accomplishment of everyday tasks, it is necessary to combine memory with knowledge gathered from everyday life.* Such knowledge consists of our semantic memory. Semantic memory holds a large body of general knowledge and it does not include specialised information. Instead episodic memory is responsible for maintaining non-general purpose knowledge in the form of facts, rules, experiences and images [Pr94].
- *The use of graphical interfaces substantially reduces the required amount of knowledge users have to remember.* Consider early command-driven systems, such as **DOS** and **UNIX**. Messages such as “>” or “ls -a” were particularly cryptic [Pr94]. Even though single characters or function keys require a single stroke for command execution, generally it is a lot more difficult to remember an arbitrary letter than recognising and choosing a command from a menu. Unlike command-driven systems, menus have the strong advantage that users do not necessarily have to remember the particular command but rather to recognise it. In other words, the use of UIs has resulted in minimal mental effort to interact with computer systems. Much of the information concerning the system’s structure and functionality is available at the interface. In many instances, the intuitive direct feel of the interface means that users do not have to think about what they are doing [Pr94]. The scientific explanation for this is that the phenomenon of recognition has clear superiority over recall. In 1988, Norman [Pr94] developed the notion of recognition and recall in terms of *Knowledge in the head* and *Knowledge in the world*. According to this theory, when a person carries out everyday tasks, he combines information that is stored in memory with information in the world. The term

“information in the world” comprise the usage of various types of cognitive aids, such as structuring our environment in a particular way to offer the crucial information that will remind us of what is to be done. Recognition can be anaesthetic and inelegant, especially if there is an urge to maintain a lot of information. In the case of recall nothing needs to be visible, therefore aesthetic considerations do not present an important issue.

Bearing in mind all these points, scientists are trying to create highly visualised user-interfaces in order to achieve greater user satisfaction. Such interfaces are aiming to utilise brain capacity as efficiently and economically as possible. Some modern innovations that are beginning to become gradually popular and seem to be quite promising are analysed in the following section.

Section 2.3 Innovations in Visualisation

2.3.1 Tree-Maps

A tree-map maps hierarchical information usually to a rectangular two-dimensional display space trying to utilise 100% of the designated space [Jo91]. Tree-maps use a slice-and-dice strategy to partition the relevant display area into a collection of rectangular boxes that represent the tree-structure. Their obvious advantage is that they maintain the global context while being able to provide access to detail. The whole idea of tree-maps is to summarise data in a fully understandable and pleasant manner. The screen utilisation is maximised while scrolling is not required. The number of nodes that a tree-map can display is considerably higher than that of a traditional node-link diagram.

In searching visual representations for information, software designers have to face the incorporated difficulties of hierarchical structures such as *table of contents*, *menu structures*, *organisation charts*, *stock portfolios* or *computer programs*. As a result, a

great variety of tree-maps have been developed for each particular case. Two characteristic examples of novel interface-development employing tree-maps are:

- The user-interface designed and implemented on an Apple Macintosh by Brian Johnson (fig.2) having accomplished the extremely challenging objective of visualising a file structure of a thousand files in a single display area [Jo91]. Additionally, the interface is capable of representing the file type by using special colour coding. The objective was accomplished by developing a novel two-dimensional space filling representation of tree structures in which the represented files use an area proportional to their size. Within a few moments of observation the user can instantly trace several large files which evidently cover the smaller ones. In general, colour can be used to represent file properties such as date of creation, security status, file type, and many other attributes relevant to each particular file. After a few minutes of a brief introduction, the users have unusual powers to browse rapidly among different subtrees, crossing many levels of the hierarchy.

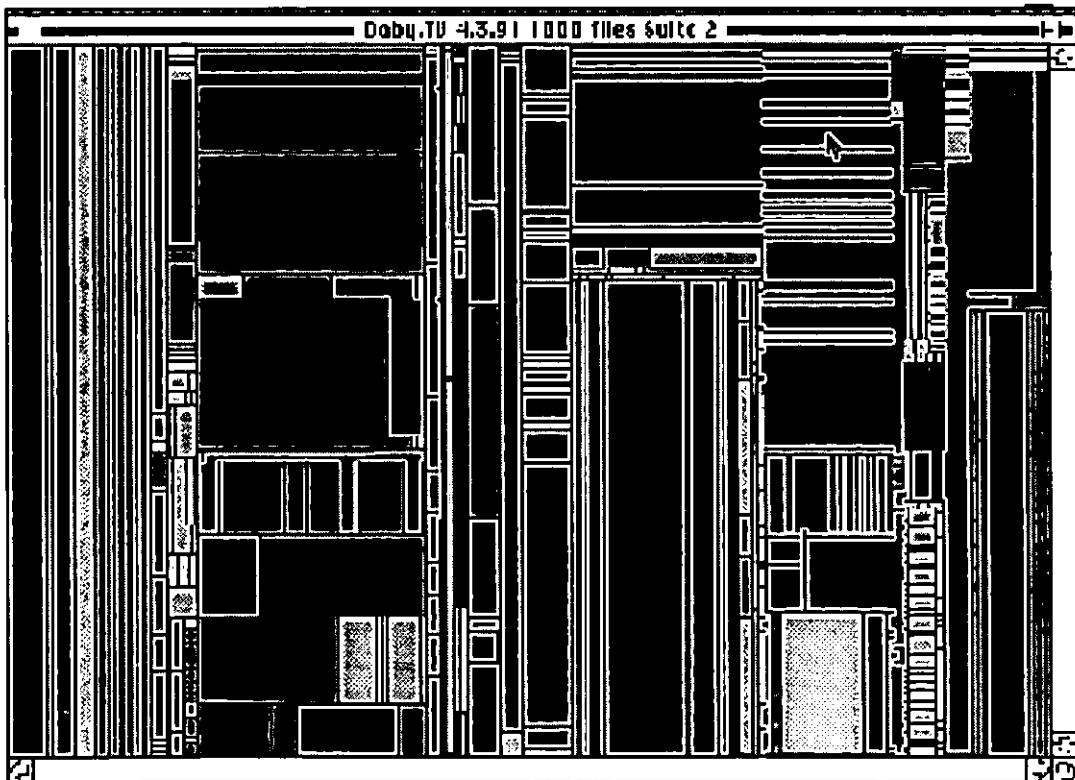


Figure 2: Tree-map screen snapshot with 1000 files

- The visual information management interface for network configuration, designed and implemented by the Human-Computer Interaction Laboratory at University of Maryland. The objective of the project was to provide a unified interface for exploration, querying, data entry and verification. Compact colour-coded tree-maps allow successful representation of the multiple hierarchies in networks, while a combination of tree-maps and conventional node-link visualisation of trees show hardware containment hierarchies. During the first stage of the project, the focus was on satellite networks, which in general are less complex than heterogeneous networks, due to their *star* topology, as opposed to *mesh* topology [Ku94]. Starting with satellite networks enabled the particular research group to design and develop initial prototypes quite rapidly in order to collect immediate feedback from relevant user groups.

2.3.2 Emotional Icons

Another modern innovation which still is not met often, is the invention of emotional icons. The use of emotional icons is intended to increase the effectiveness and impact of the user interface onto a complex data panorama [Eli96]. The icons respond to the presence of a particular user as an element of this virtual world, with characteristic behaviour, which might depend on the user profile, his current activities and interests. Icons are designed to advance or retreat, grow or shrink, act aggressively or passively, reflecting the relative importance and attributes of the data set that they represent. Additionally, the visual impact is enhanced by the use of sound to create a richer information environment. Examples of emotional icons are shown in fig.3c.

Besides their “emotional” role, icons are an excellent alternative to command names. One of the main advantages is that although it’s quite usual for users to forget the meaning of command names, especially those who use a system rarely, they find it a lot easier to remember the meaning of an icon. However, many software designers have

exhausted the iconic power due to replication of representation, which leads to confusion. In order to confront this problem, it's now common practice to combine pictorial images with names which dissolve any ambiguities with respect to the icon meaning.

2.3.3 Animated Icons

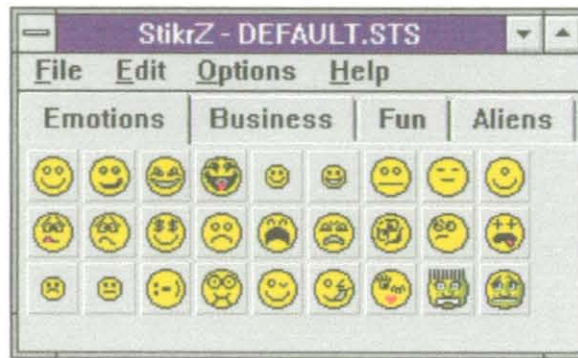
A recent development is that of *animated icons*. The idea behind them is “bringing icons to life” [Pr94]. By animating the underlying function of the icons, complex and abstract processes can be more effectively portrayed. Animated icons are expected to



(a)



(b)



(c)

Figure 3: Snapshots of various kinds of icons

(The above are available on <http://www.traxsoft/stikrz.html>)

be easier to recall and add more meaning to the visualisation process. Up to a specific extent, they can be considered as an extension of the use of emotional and static icons. However, the main difference is that the addition of animation can save the user the trouble of imagining what dynamic actions the conceptual symbols are interpreting. Instead the animation does this task on his behalf.

The critical point of the whole effort is whether or not the animation is properly focused on the key aspects of the function to be represented. If the animation is not relevant, animated icons can lead to complete confusion by reminding to the user the simple fact that what he sees is just a set of moving elements. There are a few technical problems as well which can degrade the performance of animated icons such as the relatively limited size and the cursor which can get in the way and obscure the animation (since in order to select an icon the task involves covering it with the cursor). Therefore, in particular instances a well designed static icon may be easier to comprehend than an animated one, since it is always less complex and it can capture more successfully the essence of some particular visual representations.

2.3.4 Resemblance, Exemplar, Symbolic and Arbitrary Icons

Although this set of icons is not novel, they present a certain theoretical interest relevant to the various iconic representation forms. *Resemblance icons* portray the underlying concept through an analogous image (i.e. a phone for a phone-box, see fig.3a). *An exemplar icon* serves as a typical example (such as a cross for a drug store, see fig. 3b) while a *symbolic icon* is used to convey an underlying referent that is at a higher level of abstraction [Pr94]. A characteristic example of a symbolic icon is that of the STOP word to convey the concept of caution (fig.3b). The usefulness of the above generic types of icons is directly proportional to the compatibility of the different semantic memories between the icon designer and the set of users. In general, different semantics may not allow the most to be made out of iconic representation.

2.3.5 Flexible Filters

The value of flexible filters is best illustrated by mentioning a *concept demonstrator* that has been developed at BT Labs in order to demonstrate the use of an interactive visual interface for exploration of network fault data [Eli96]. The particular visualisation allows effective browsing of a network database in order to look for trends or predictive patterns. The user is able to pinpoint specific problematic sections with a range of flexible filters. By resetting the thresholding and filtering, it is possible to identify standard alarm conditions. However, flexible filters can lead to excessive data summarisation which inevitably will sacrifice the maximum possible resolution for a more global context in return.

2.3.6 AlphaSliders

AlphaSlider is a query interface that uses a direct manipulation slider to select words, phrases or names from an existing list [Osa93]. A prototype of Alphaslider developed by the HCI Laboratory at the University of Maryland is shown in fig.4. AlphaSliders are a closer step to the ideal easy-to-use interface with non-keyboard input devices. As

portable, palm-top and pocket computers become more popular, AlphaSlider might be the only serious keyboard rival until voice (too many sources of variability associated with the signal) and hand-writing recognition become faster and more dependable [Osa93, Da96]. AlphaSliders are very practical for small portable devices of the size of a credit card; however this does not imply that can't be used for larger and more complex applications. By the inclusion of the appropriate widgets in graphic user interface management software, further refinements and extensions to the AlphaSlider concept seem possible.

Pause

Find: CAMBRIA SHORES MOTEL

Figure 4: 800-Number Yellow Pages

Osada [Osa93] described an experimental example of the AlphaSlider in which he chose a potential consumer electronics application that involved a small credit card size pocket computer with travel information such as airlines, hotels, toll-free telephone numbers etc. As can be seen from fig. 4, a screen with the potential to retrieve eight hundred number yellow pages was presented to users whose task was to find entries appearing in the question field and then push the 'Dial' button. If the selected entry was

correct, the system would emit a bell-like sound and the next question would appear. If not, a short beep would indicate an erroneous answer. 'Pause' and 'Resume' buttons gave the subjects the option to stop working if they needed a break. A mouse was used to manipulate the interface. The point of the study was to calculate the response time of the twenty four subjects used in the particular experiment. Fig.5 shows how the AlphaSlider Interface can be used.

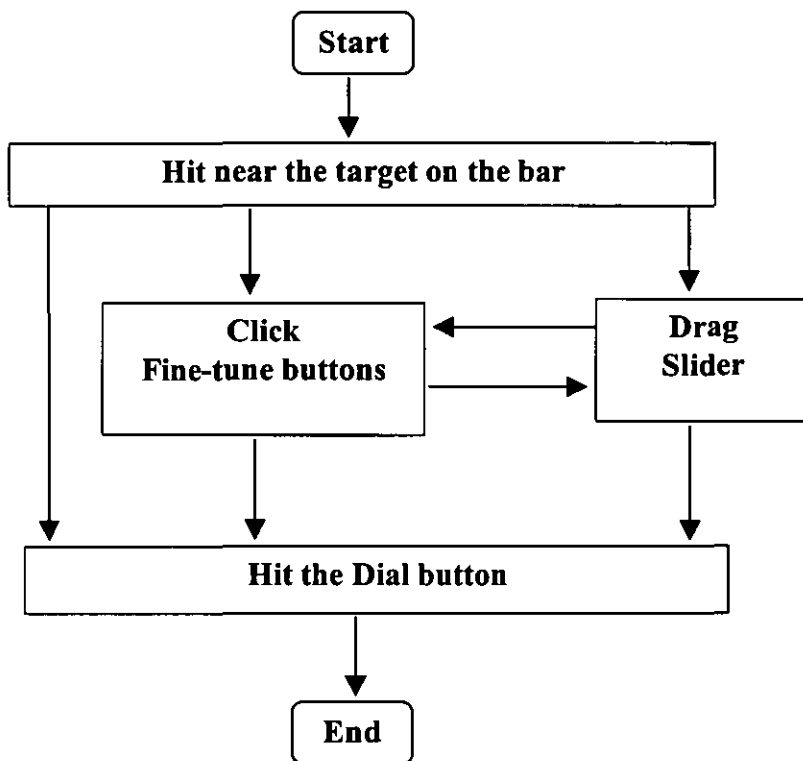


Figure 5: Transition diagram of 800-Number Yellow page usage

Users start searching by looking at the question field to the target entry. Performing a rough estimate, they have to click the big slider near the letter of the target name. If the result box does not contain what they were asked for, they have to click the fine-tune buttons or re-drag the big slider until they find the targeted name. To complete their task they must click the 'Dial' button.

2.3.7 The Range-Selection Slider

Most existing sliders provide a visual way of selecting a single value from a set of values [Car95]. However, specifying a range of values is impossible unless a pair of sliders is used. This approach however allows for a simple, yet important problem: the lower boundary can be erroneously set to a higher value than the upper boundary. The range-selection slider prevents this happening (fig.6). As it can be seen, it uses two separate indicators which can be moved separately by the user in order to select a range of values while still occupying the same space as a conventional slider.

Each indicator operates like an autonomous horizontal slider whose boundaries are set by the respective sides of the rectangle and the counterpart indicator. Dragging the indicators as required redefines the available range of values.

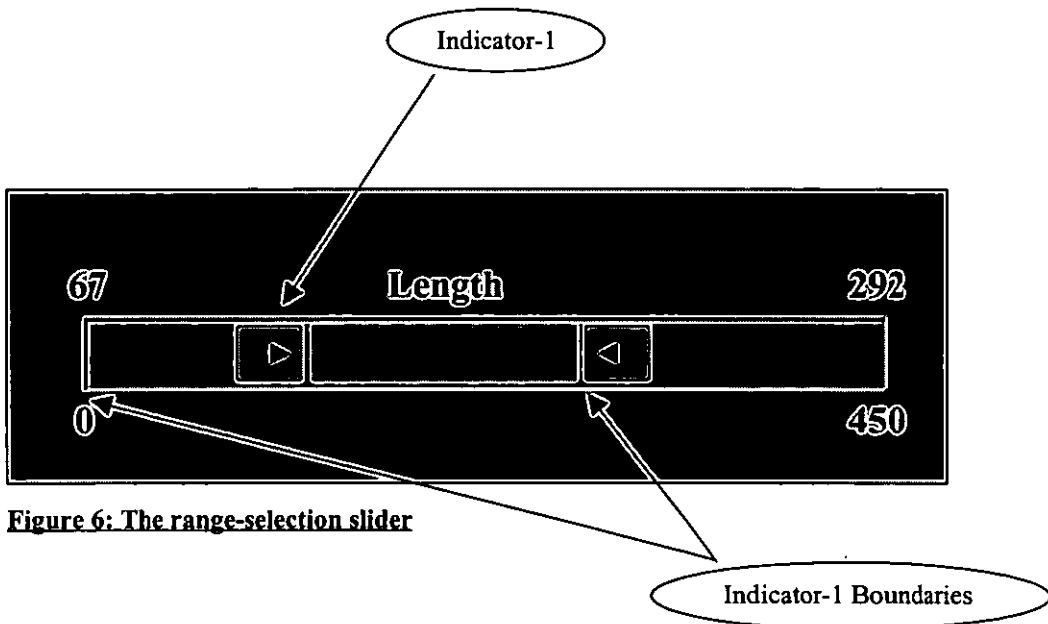


Figure 6: The range-selection slider

2.3.8 Elastic Windows: A possible cure to 'Windowitis'

As stated by Card et al. [Ka96], the computer monitor is not only used as a visual output but also as an external memory for users. Based on the previous remark, a user could possibly demand a windowing system that maintains only the necessary information on the screen whilst filtering out the unnecessary windows at a specific instance. Windowing systems should allow users to integrate, compare, organise, distil, summarise and apply information. According to Kandorgan [Ka96], modern windowing systems still operate on the basic principles of window management. Almost all systems adopt the *independent overlapping windows approach* where windows are allowed to overlap each other, only a single window operation is possible at a time and, finally, the size and location of any window is autonomous.

However with advances in computer technology, much more demanding software applications have come into existence. The amount of information that users have to face is constantly increasing whilst this is usually unorganised and dynamically changing [Ka96]. Therefore users themselves are forced to do the organisation. Applications such as CAD, CAE, OODEs (Object-Oriented Development Environments), GIS (Geographic Information Systems) and network monitoring tools are typical multi-window applications. In such software it is often necessary to open many windows displaying simultaneously different parts or representations (i.e. toolboxes, commands, nodes etc). With such a vast increase in open windows on a single screen, concurrent visualisation of all relevant information becomes an important and complex issue. As the number of windows per task has dramatically increased, task-switching has become time-consuming since more windows need to be opened/closed or moved/resized under the independent overlapping windows approach. According to Gaylin [Ka96], the number of window operations that are used to change the current active window set approaches 63% of the entire operations in an independent overlapped window manager system. This means that people switch among tasks rather frequently and each time, in order to do so, are forced to interact

with each and every window separately due to lack of any graphical depiction of the window relationship.

Kahn et al. [Ka96] observed the phenomenon of '*windowitis*' that describes situations in which too many open windows are present. In windowitis situations users become quickly disoriented and turn out to be unproductive in completing their tasks. On the basis of all these previously discussed problems, Kandogan and Shneiderman updated the requirements of multi-window systems as follows:

- Promotion of organisation and co-ordination of windows according to tasks
- Provide rapid task-switching and resumption
- Users' cognitive resources economy on window management operations, so focus can still remain on task related operations
- Efficient and productive screen space utilisation
- Provision of spatial layout that indicates window relationships

The proposed idea that seems to fulfil the above visually demanding requirements is called 'Elastic Windows'. The principles behind this invention are:

- Hierarchical window organisation
- Space-filling tiled layout
- Multi-window operations

According to the creators, hierarchical window organisation allows users to structure their work environment according to their tasks. On the other hand, multi-windowing operations can reduce cognitive load on users by decreasing the number of window operations. Last but not least, space-filling tiled layout guarantees screen space efficiency without the wasted background of the overlapped windows approach.

Whilst being aware of possible oversimplification, one could say that the most obvious characteristic of elastic windows is their elastic resizing. Each window operation affects the entire screen and echoes a consecutive resizing to every other window. Their obvious advantage is a much more global view enabling zooming in/out of an application without losing orientation and concentration.

2.3.9 User Interface Management Systems (UIMSs)

UIMSs are a fairly recent tool to ease the design of user-computer dialogues and related software. These systems significantly reduce the work required to specify and design a user-computer dialogue [Car95]. They also allow non-programmers to prototype and design complex UIs due to their highly visualised and non-technical interfaces.

One serious criticism about UIMSs can be that they comprise a finite set of widgets that can be fully manipulated but not easily enriched with newer interaction objects. Therefore if the designer concludes that the existing widget set is not sufficient to design a new front-end, he would have either to compromise and redesign the computer dialogue with the existing visual resources provided or to manually code the desired dialogue and then integrate it to the UIMS. By admitting such a possibility, the entire user-friendly and non-programming essence of UIMSs is jeopardised.

The Interaction Object Graphs (IOGs) present an effort to solve this widget-building problem [Car95]. IOGs are a new method of adding widget specification at a higher level than programming languages and have been used to specify the University of Maryland widgets. Although there are a number of methods to specify user interfaces, none concentrates on interface layout and spatial relations between interface objects. This can lead to important omissions (i.e. when an interface has been resized, the spatial relationships must effectively change). IOGs are created to fill this particular gap. They combine attributes from DFDs and state transition charts, whilst at the same time they show data relationships as well as control flow.

In order to describe the various state-transitions, IOGs abstract the interface into several objects such as *booleans*, *numbers*, *strings*, *points*, *regions*, *icons*, *view ports*, *windows* and *user inputs*.

Booleans, numbers and strings are the typical asbtractions with all known operations. Points are an ordered pair of numbers usually representing screen co-ordinates. A region is a set of display points relevant to an origin called the *location*. They also have a size operator which returns the height and width of the smallest rectangle which covers the region. Icons are regions with pictures while a view-port is a region with an associated mapping function. User inputs are mapped to IOG events and can be input either via the keyboard or mouse. Finally windows group the above objects together. Fig.7 demonstrates the IOG of the range-selection slider.

As can be seen, the above IOG presents the range-selection slider as four different entities: the Background, the ButtonViewPort, the low SlideIndicator and the high SlideIndicator. Although IOGs might look complex and generate antipathy at first glance, the user might agree that in visual terms, they have nothing in common with any archetypal documentation procedure. Their strong advantages are:

- *More compact*: IOGs are a much shorter way to specify user interfaces. In addition, the specification for handling unexpected user behaviour is much simpler.
- *Direct representation of data*: The direct representation and modification of data increases the specification power of IOGs with regard to interface objects.
- *Offer visualisation of Widget behaviour*: IOGs are the only specification method that gives their readers an idea of how the widget will appear on the screen.

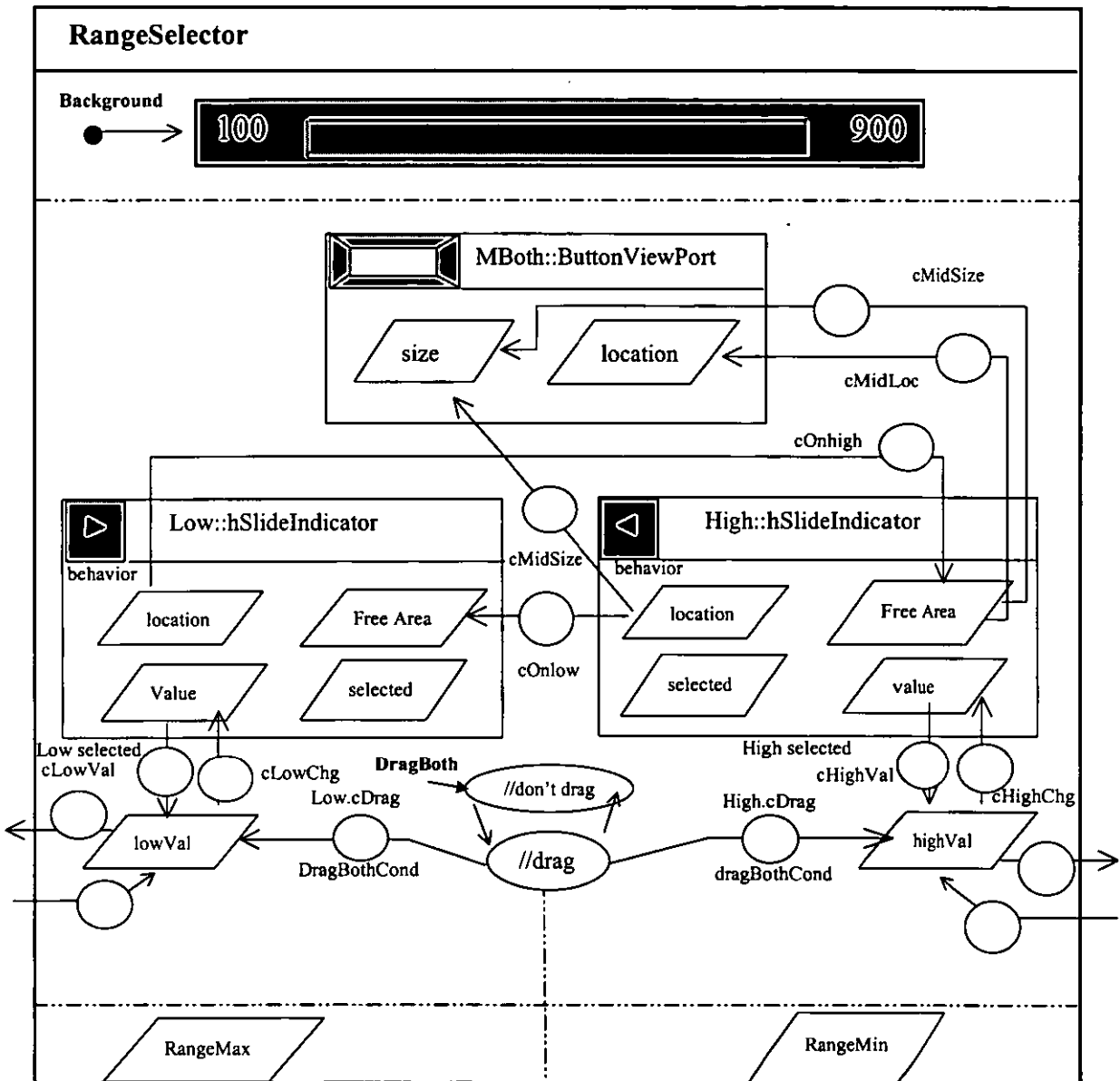


Figure 7: Specification of the range-selection slider [Car95]

Section 2.4. Conclusion

To conclude, this chapter was intended to provide rather a spherical point of view and to enable understanding and successful linking between the notion of visualisation and its relevance and importance to communication network systems. The implications of

this increasingly data-driven world can not leave network systems unaffected. The importance of keeping human participation in the process of decision making is great. Although computers communicate with each other at tremendous speed, effective human involvement is very much required at this point. Visualisation seems to start playing a major role in the formation of modern user-interfaces which by default are going to change considerably human-computer interaction in the near future. The saying "*a picture is worth a thousand words*" is very true and underlines the simple fact that raw data of itself has no value just as most raw materials have no value if not properly processed. It is only in the final conversion through to improved presentation that one can increase his knowledge and achieve a real benefit. Even in particular circumstances under which human participation makes a minimal contribution to the overall analysis, simple visualisation has the potential to improve control and understanding dramatically. In past years, computer programmers were preoccupied with successfully answering the question "What to present?". Nowadays although the same problem still exists, a second question "How to present?" has evolved, getting slowly but steadily more difficult than the first one.

CHAPTER

3

Creating a Highly Visualised Interface for Network Performance Data Pre-Analysis

Section 3.1. Introduction

This chapter summarises some fundamental network monitoring issues and then presents work on designing a highly visualised interface to be used as a pre-analytical tool for network performance measurements. The cornerstone of the whole effort is directly linked to the notion of "**Figural Deformity**" (FD) and its potential implementation.

Section 3.2. Network Monitoring

The primary purpose of any Network Monitoring System is to generate accurate and up to date information with respect to the current network load and status. Network information can be collected by a system of distributed monitor stations that are able to generate test traffic to each other across the network. This is referred to as intrusive monitoring. Alternatively, network monitoring components themselves can deduce information about the network from the traffic that passes their interfaces. This is non-intrusive monitoring.

The task of monitoring distributed systems can be generally more difficult than monitoring a centralised station while is not limited to simple observation [Ja91, Bas98]. This statement is based on the fact that distributed monitoring systems incorporate numerous components that function both separately and concurrently (i.e. for initiating and terminating tests or processing the retrieved data [Phi95]). All monitoring systems must perform their function as dependably and accurately as possible whilst generating only minimal interference to the system being monitored. Some monitoring systems (i.e. the monitor station developed by Loughborough's High Speed Networks Research Group for monitoring and measuring SuperJanet's QOS) manage to be synchronised by reference to an external global time agent, such as the Global Positioning System [Phi96, Bas98b].

Network performance monitoring systems generate significantly bulky data with respect to the characteristics of the systems they monitor. This data, being in a basic form presents little direct information to the users involved in the monitoring system, but its importance is great when processed as further conclusions can be drawn. Performance monitoring of networks is a complex task involving many different levels (layers) the first of which is called *the observation layer*. The observation layer, as the reader may suspect, is concerned with the raw-data gathering process which is generally realised by *implicit spying*, *explicit instrumenting* and *probing*.

Implicit spying is the least intrusive monitoring technique. It basically involves observing activity on the system bus or network link and its strong advantage is that there is almost no impact on the performance of the monitored system [Ja91]. Filtering can be used quite extensively in order to decide which activities are of interest and, thus, to be recorded. The filters generally consist of conditional expressions.

Explicit instrumenting requires incorporating trace points, probe points, hooks or counters in the system and although the technique enables a standard data-naming and reporting format which other monitor components can make use of it, the process can cause some overhead on the system [Ja91].

Probing involves making feeler requests [Ja91] on the system being monitored by generating sample-workload to sense its current performance.

The *presentation layer* is the intermediate level in the entire performance monitoring process and deals with user-interface design [Ja91]. As fig.8 illustrates, its existence is intended to enable the user to pass his requests to the monitoring system and, moreover, to translate on his behalf the system's responses to his inquiries. This layer is closely related to the whole set of applications for which the monitoring system is used. For example, it maybe used for performance monitoring, configuration monitoring or fault location.

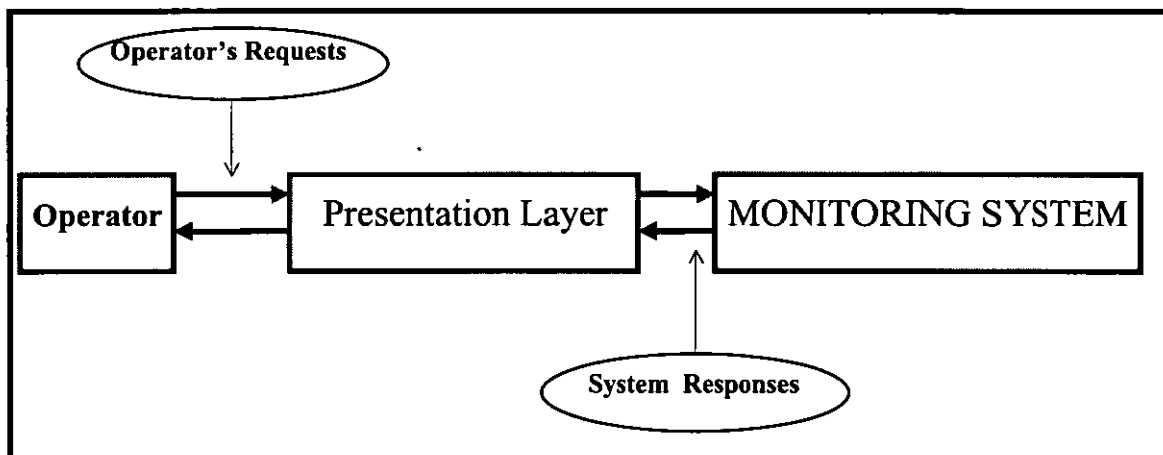


Figure 8: The Role of the Presentation Layer in Network Monitoring Systems

The first issue relevant to presentation is the generation of various summaries which can be needed especially when scanning historical data. Summaries can be based on an hourly, daily, weekly, monthly or even yearly basis. These summaries should be organised in a manner similar to the structure of the data being gathered (i.e. a daily summary could be obtained from the hourly summaries, a weekly summary could be obtained from daily summaries, a monthly from weekly ones, etc.).

Another important task of this particular layer is the alarm mode which eases the job of network managers by notifying them only if a specific set of conditions occurs (for

example, if error rates occur above a pre-set threshold or security threats such as unidentified new network nodes are detected).

3.2.1 Visualisation and Current Network Monitoring Tools

Whilst there has been significant progress towards more visualised User Interfaces under the broader area of Information Exploration (see chapter 2) current Network Monitoring Applications still persist in employing either text-based techniques or at the most **2-D** and **3-D** graphs which are a very basic method of visualisation of raw data [Pa97]. Despite the fact that such presentation methods are of great importance when mathematical and statistical conclusions are to be drawn, one arguably could claim that network operators are in great need of more qualitative rather than quantitative tools which can provide a more global context in exchange of numerical overdetailing.

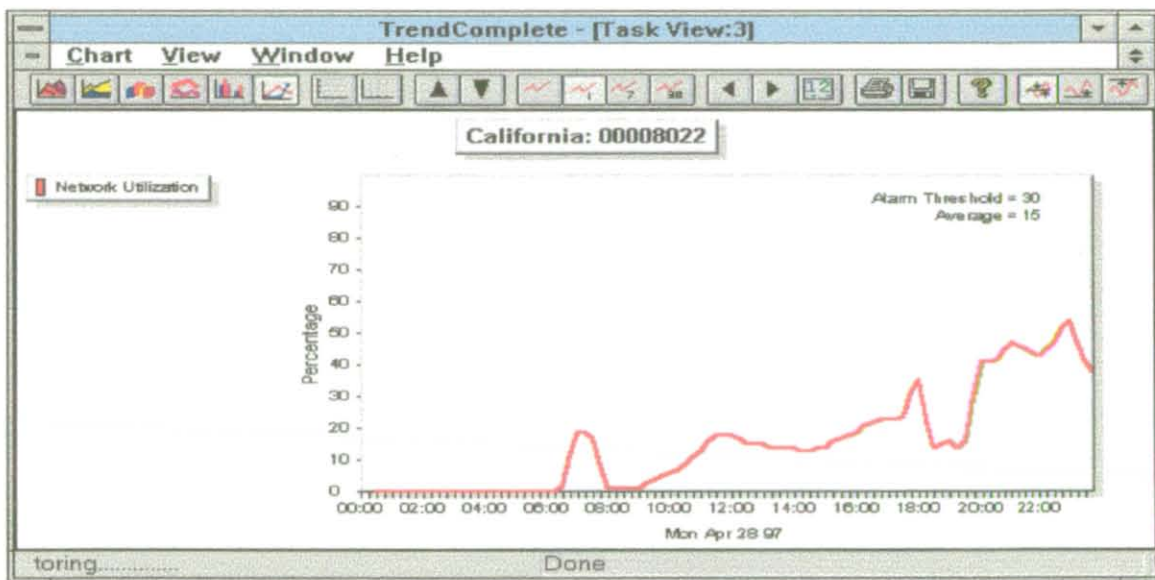


Figure 9: Network Utilisation Graph, Kansmen Products (Product Name: TrendComplete)

Fig.9-14 are various snapshots from two modern network monitoring tools whose creators are proud of the high degree of visualisation that is incorporated in these products. Indeed whilst they might look very competitive and very impressive at first glance, however a deeper analysis can expose their visual weaknesses. For instance, consider fig.9 which demonstrates an interesting [Su91] statistic: the network utilisation. Some of the most important information that one could expect to dominate this display

would be the Alarm threshold value and the Average value. However, these values seem to be considered of minor importance and are represented only textually, being squeezed at the right-upper corner of the display. To make things worst, the use of red colouring is not appropriate. Colour should be used for styling and to provide redundant information [HUSAT89, US86]. In fact most printers and photocopiers are black and white. Colour appeals but adds very little to overall performance and should be used when the desired outcome cannot be achieved by other methods. However colour is a very powerful medium and once used, will be the dominant element in a display. People perceive its use as logical and meaningful, even if it is not, and will attempt to fit rules to its use [HUSAT89]. Colours are associated with strong stereotypes e.g. red for danger, green for safe, etc. Therefore one should never go against established stereotypes and clearly in this graph the rule has been bent. The potential result is that initially the user that will examine this graph is going to assume that there is a serious problem whilst after a few instances he will simply ignore the red colouring.

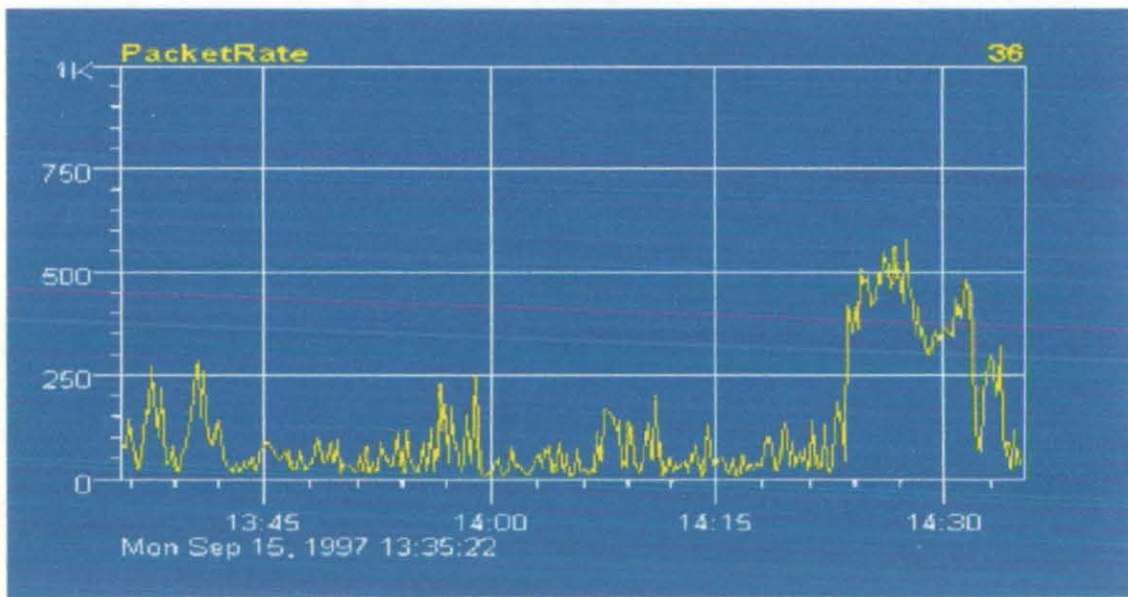


Figure 10: Network Trend Graph, Proteon Products

Furthermore, Fig.10 presents another instance of network monitoring products where colouring has not been given appropriate consideration. In this case, yellow is a stereotype for caution and still may alarm users unnecessarily.

Colour confusion (pollution [Pr94]) is another important issue. The choice of colour should be meaningful and clear to the user. When colour is irrelevant, it can be very distracting. Poor use of colour can reduce productivity while in extreme cases it can lead to a system which is objectionable to use [HUSAT89]. An example of excessive colouring can be found in fig.11 in which eight different coloured bars have been used. According to the HUSAT research institute, the number of colours used in a single screen should be enough to satisfy the requirements of the particular task and in most cases a maximum of four colours should be sufficient.

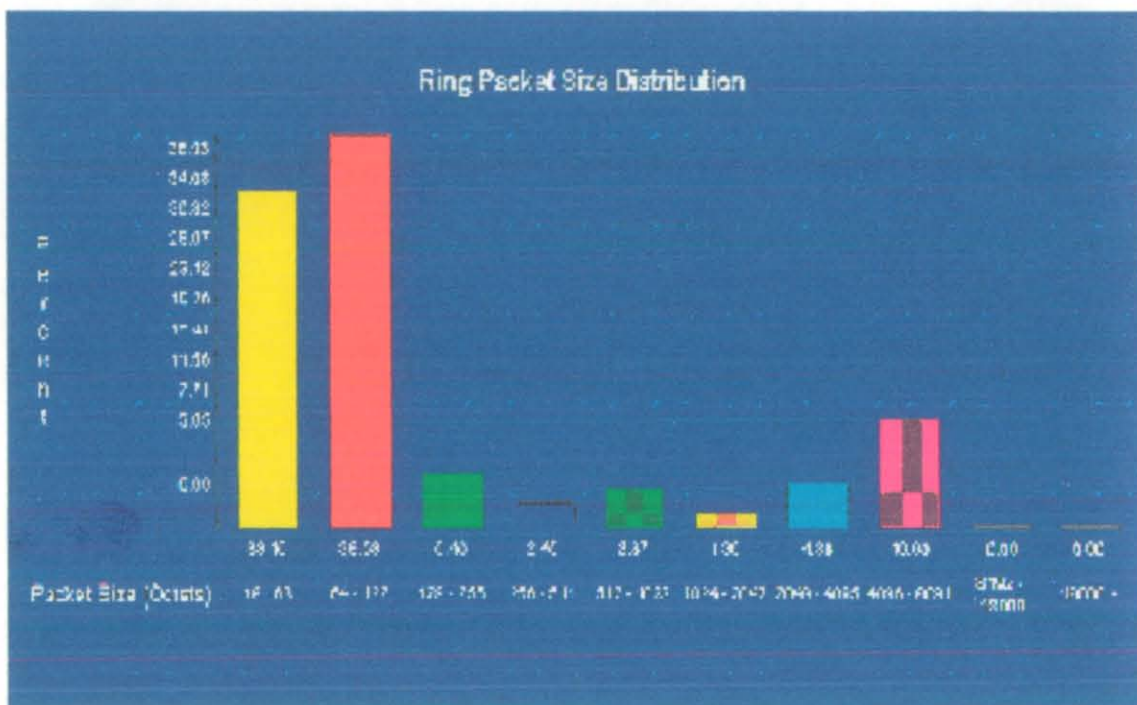


Figure 11: Packet Sizes, Proteon Products

Despite the fact that the above display screens might have some ergonomic mistakes, they are still much more visualised than the examples presented in figures 12, 13 and 14. Text-based screens are a classical example of bad User Interface Design (UID). UID is a very difficult assignment mainly because it is complex [Maha96] and combines two awkward disciplines: psychology and computer science [Thim91]. Apparently, these interfaces have been designed in absolute absence of any psychological consideration. A few years ago, the expert programmer was obliged to know only programming

mechanics. Today's programmer's repertoire has to be highly enriched with many additional issues such as ergonomics. Nowadays it is not the users who have to adapt to programmers idiosyncrasies but rather the opposite is true. Therefore such archaic practices are still present in network monitoring software either due to lack of competition, or because the people who design them do not allow users to participate in the design process.

Viewers/MIB/WebRMON/Log Tables/Log Table [All](#) [Top](#) [Bottom](#) [Forward](#) [Back](#)

#	Time	Description	Value Object
1	Wed Sep 17 23:15:01 1997 .	Parana Beta: Upper Binding Alarm	3021:fileDcastPkts.2
2	Wed Sep 17 23:15:23 1997 .	Parana Beta: Lower Binding Alarm	3021:fileDcastPkts.1
3	Wed Sep 17 23:15:23 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
4	Wed Sep 17 23:15:41 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
5	Wed Sep 17 23:16:01 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
6	Wed Sep 17 23:16:21 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
7	Wed Sep 17 23:16:41 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
8	Wed Sep 17 23:17:01 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
9	Wed Sep 17 23:17:21 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1
10	Wed Sep 17 23:17:41 1997 .	Parana Beta: Lower Binding Alarm	0 webP900LogControlRunEntries.1

Figure 12: Alarms Report, Proteon Products

TrendComplete - [Task View]

Task Configure Report View Window Help

Num	Name	Network Address	Type	Status	Region
1	Kansmen-K2	00000010:000000000001	NetWare Server	Monitored	California
2	Kansmen-K1	1970102:000000000001	NetWare Server	Monitored	California
3	00008022	00000010:000000000001	Ethernet Network	Monitored	California
4	00008025	00000010:000000000001	TokenRing Network	Monitored	California
5	00018022	205.158.201.197	Ethernet Network	Monitored	California
6	00018025	205.158.201.197	TokenRing Network	Monitored	California

toring... Done Uptime:

Figure 13: Main Screen, TrendComplete, Kansmen Products

Netware Server K2

4/22/97

Variable	Average	Minimum	Maximum	Trend	Next week projection	Next month projection	Threshold
Health	31	31	31	increasing	40	58	N/A
Server Availability (%)	100.0%	N/A	N/A	N/A	100.0%	100.0%	99.0%
Disk Free Redirection Area	100%	100%	100%	N/A	100%	100%	45%
Cache Hit Rate	74%	74%	75%	decreasing	70%	44%	20%
Volume Free Space	9%	9%	9%	decreasing	8%	7%	10%

Figure 14: Server Data, TrendComplete, Kansmen Products

Section 3.3. Designing *Appropriate* User Interfaces for Network Monitoring

The implied conclusion of the previous section is that monitoring tools must provide the potential of not only collecting, but also examining and presenting network information since data mining becomes increasingly difficult as the volume grows [Shne91]. Suffice it to say, collecting and investigating network performance data are two separate tasks. Although effective data collection is crucial for any further continuation, it is not the only part of the whole network monitoring operation. The investigation of the collected data will reveal pertinent conclusions with respect to the network stability and its operational efficiency. The required inspection can be best accomplished by using *appropriate* software tools. At this point, to put it almost in Berkeley's words [Ba86], we don't want to raise dust and then complain we cannot see! The word **appropriate** by itself incorporates some basic ergonomic issues that need to be considered before any further elaboration.

3.3.1 Two Important Interface-Design Criteria

Many can argue that making things easy to use is evidently a very good idea. However the idea of making things straightforward is by nature simple-minded and can incorporate trade-offs. One most interesting outlook by Thimbleby [Thim91] is that if a computer is sufficiently powerful to be 'easy to use' this implies that it is complex enough to confuse its users. Or, if it is made easy to use, it becomes increasingly trivial, and may

ultimately be easy to use only to do nothing worth doing. Based on the same pessimistic viewpoint, a useful system might never be really easy to use.

Designing interactive systems is much more than creating 'easy to use' screens. Often, the programmer might have to balance ease with security, computational power and other crucial factors. Accordingly, we can't always decide easily if an interface does or does not deserve to be labelled as appropriate, useful or usable. From an ergonomic perspective, the last two words do not possess the same meaning, although they do sound similar. Even though various definitions exist, according to the author's opinion the most successful are those of Shackel, chairman IFIP TC. 13 on H.C.I. (Human Computer Interaction) [Pa97]. Professor Shackel defines *useful* as something which is advantageous, profitable, fit for some desirable end, or has power to satisfy human wants. On the other hand, he defines *usable* as something which is able to be used, applicable to a purpose. Throughout the years, usability and utility have become two very important design criteria and their consideration almost guarantees three very important advantages: 1) *Improved Throughput*, 2) *Reduced Errors* and 3) *Easier Learning of the particular software application*. To illustrate what has just been claimed in the last sentence, let us compare the early command-driven applications with modern user-friendly programs employing all sorts of impressive features such as user-tailored interfaces with modifiable intelligent behaviour according to individual needs [Kar98, Ros94]. Both types of software are equally *useful*; yet not equally *usable*. The early command-driven applications were mostly used by expert-users or at least knowledgeable people who had already established the required technical background, and presumably they were expected to overcome any obstacles quite easily. On the other hand, user-friendly programs operate on the assumption that even persons with very little, if any, computer experience should be able to perform the same tasks equally effectively.

The rapid evolution of user-friendly systems was triggered by the fact that more and more non-specialists became computer-users. As a result, software designers had to cater for a much wider range of clients and thus they were forced to create more usable products. Direct Manipulation Systems are the best way to demonstrate the implications of creating

usable, in addition to useful, applications. The idea behind such systems is based on the metaphor of a desktop in which respective icons represent commonly used objects. Direct manipulation systems offer increased functionality, speed, greater comprehension, improved user confidence and also receive wide acceptance [Li92] in the computer community. Nevertheless, a very important remark should be made: the question should not always be whether to stamp systems as 'usable' or 'not-usable' but to decide if these systems are **easy enough** to use, and whether possibly one could find new ways for further improvement. Usability leads to a constant battle between technical and mental constraints.

Section 3.4. The Idea of Figural Deformity

As one may suspect, a hidden term behind user-friendly software innovations is the word *visualisation*. By definition, visualisation ensures improved user awareness and understanding. Fig.15 demonstrates the notion of visualisation and its relevance to interface design.

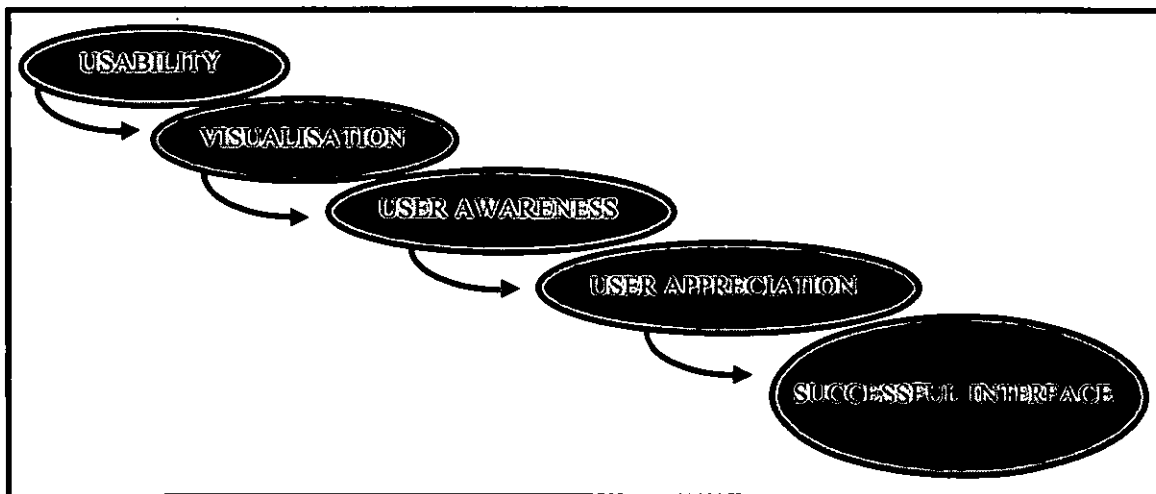


Figure 15: The impact of Visualisation on User Interface-Design

Although 2-D or 3-D traditional presentation methods (i.e. line or bar-charts) are useful for comparing and visualising small amounts of raw data, their advantages diminish

under the pressure of large amount of information [Ju92]. This is because they are not packable in terms of human perception; at least not without suffering major alteration [Pa97]. The human mind cannot process more than a few graphs simultaneously. This innate incapacity is directly linked to *short-term memory* (also *working memory*) and its fixed available storage of at most nine "chunks" of information [Pr94]. Even a single graph can involve nearly nine distinct concepts (i.e. scaling of axes, background colour, graph shape, axes-labels, grid, graph origin, etc.) and as a result, when graphs are embedded in user-interfaces, users are susceptible to short-term memory overflow, a condition which certainly does not maximise their efficiency nor their productivity. The above provides a scientific justification of why a single interface must not involve many graphs (not to mention the textual formats shown in fig.12-14). Yet many times, when browsing network performance data it is necessary to watch numerous graphical outputs concurrently and monitor the variation of several parameters in order to avoid abnormal deviations from a fixed equilibrium. In such cases, new ways of packing and delivering information are needed. At this point it might be useful to include a short analysis of the human memory mechanisms and how these are used to manipulate incoming information.

3.4.1 Short-term and Long-term memory

John Kennedy said that man is still the most extraordinary computer of all [Thim91]. Although many commentators can either disagree with or applaud the above declaration with respect to their religious, non-religious or even philosophical origins, most if not all of them would agree that any non-trivial human activity (i.e. studying and learning new concepts or tasks) would involve information processing limitations some of which are found in computers. One such limitation is relevant to human memory. Incoming information to the mind is buffered for about 250 milliseconds in any of our five sensory memories, that is one memory for each of the five senses (touch, smell, sight, hearing and taste). Even though sensory memory is quite large, there is no way to refresh the information besides repeating the external stimulus. In the case of a network operator, he would have to look at his visual display about four times a second in order to continuously refresh the information stored in his visual memory buffer, provided that he is still open to other sensory interruptions. If one needs to remember something for much

longer than a quarter of a second then the human mind must attend to a particular sense. However, shifting attention will cost fifty milliseconds, and that is when no other delays are incorporated. Therefore a user who wishes to shift his visual attention to a particular screen area would have to allow nearly 200 milliseconds for this to happen since eye-movement is involved in the process. As in the case of a micro-controller, the human brain follows a sensory interrupt prioritisation whenever attention has been shifted. A common example of focused attention is the Cocktail Party phenomenon, which is well known to psychologists; despite the noise and interference from various conversations, everyone is able to carry on his own discussion with a selected group of people. What is most interesting though is that the sensory interrupt prioritisation is dynamic and can change depending on the person's expectancy.

Once attention has been shifted to a particular sense, the information is shifted and copied to a small Short-term memory (STM) or working memory as previously mentioned. Short-term memory appears to be where consciousness resides so thinking can be performed. Therefore the role of STM is analogous to the cache memory of a modern microprocessor. Even though information can be retained for about 20 seconds in STM before decay takes place, it can be lost much quicker if overwritten by newer information generated when further thinking takes place.

Thinking involves various mental processes which need to be distinguished as *resource-limited* and *data-limited*. With respect to eye-sight, vision can be either data-limited or resource-limited. As an example, consider a computer user who works with a very old screen with poor contrast and brightness. In this case his vision is data-limited. However, if the same person works with an overloaded visual field then his vision is resource-limited (i.e. his working memory overflows). Moreover, the brain tends to devote extra resources to compensate for data-limited mental processes. Therefore it might be possible that someone who wear his spectacles, improves his remaining senses by releasing resources [Thim91].

Long-term memory (LTM) is the place where information can be retained indefinitely without any risk of decay or being overwritten. However transferring basic blocks of data from STM to LTM requires rehearsal. LTM has a large capacity and decays very slowly if at all. According to many scientists, LTM in principle is perfect but memory access without rehearsal becomes gradually more difficult because of interference from other memories. Maximising the efficiency of recall can be done by the so-called *elaboration techniques* examples of which are the various mnemonics and mental imageries. The latter can be used to strengthen the association of particular information to be recalled with a well-known picture. Although elaboration techniques are used to eliminate interference and maximise recall, they themselves can be the cause of interference as well. Another source of interference is *priming* and can occur either in STM or LTM as well.

3.4.2 FD: The proposed alternative to conventional network monitoring screens

By now two points should be clear:

- memory caching happens in STM which is very limited
- LTM is not accessible without rehearsal

Therefore if one wishes to display a vast amount of network information in a single screen and wants safely to assume that the user will be continuously alerted and will respond promptly to any unexpected changes, he has to invent visual ways that will not unwisely consume the already limited human memory caching resources.

The current visualisation novelties discussed in Chapter Two are inappropriate for substitution of the currently used practices in Network Monitoring Tools since they are not designed to display multiple parameters concurrently. Either they visualise in a hierarchical manner by obscuring the visibility of lower levels of the hierarchy due to the tiling approach being followed [Tur92] (i.e. tree-maps that are appropriate for identifying large files within a directory structure) or offer the possibility of quick data isolation from various large databases (i.e. the range-selection slider and the flexible filter).

On the other hand iconic representation can easily generate confusion as already mentioned (see chapter 2) and therefore must be escorted with pictorial images and/or names. By doing so, the data panorama would incorporate working memory overheads.

The notion of *Figural Deformity* (FD) can be a novel way of effectively visualising **multiple-variations** by employing a single object and gradually deforming its original shape [Pa97]. A very simplistic example of *Figural Deformity Visualisation* (FDV) is the use of bars in a bar-chart; or the single horizontal bar which represents downloading progress with the World Wide Web browser, Netscape. Clearly, Diagrams (1) and (2) in fig.16 represent a perceptually enhanced version of (3) and (4).

These two illustrations allow only for a single *degree of freedom*. In other words, the current object (i.e. the bar) can be deformed only with respect to its length. More sophisticated FDVs utilise objects that allow several degrees of freedom. Fig.17 illustrates an FDV with three degrees of freedom. As a result, such a graphical pattern can be used to demonstrate the variation of three distinct parameters while at the same time it economises graphical space as well as working memory.

Section 3.5. Designing an FD-Interface for Network Performance Data

Before starting an FDV implementation, one needs to decide how many parameters are going to be incorporated in the current FDV, and what will be the shape of the manipulated object. For an implementation relevant to network performance, most of the time there is interest in four parameters which are essential for the Service Level Agreement (SLA) of the network: the average packet delay, the longest/shortest packet delay, and the worst 5% of the delay distribution. The shape of the manipulated object has been chosen as the letter *E* since it is a well known symbol. The initial model of this unusual interface is presented in fig.18. Notice the additional information that accompanies the presentation of the objects. By announcing the expected figure shape and the mapping of the various parameters, we enable the user to become instantly aware of any unexpected change in the network and, thus, take appropriate action. Although it is

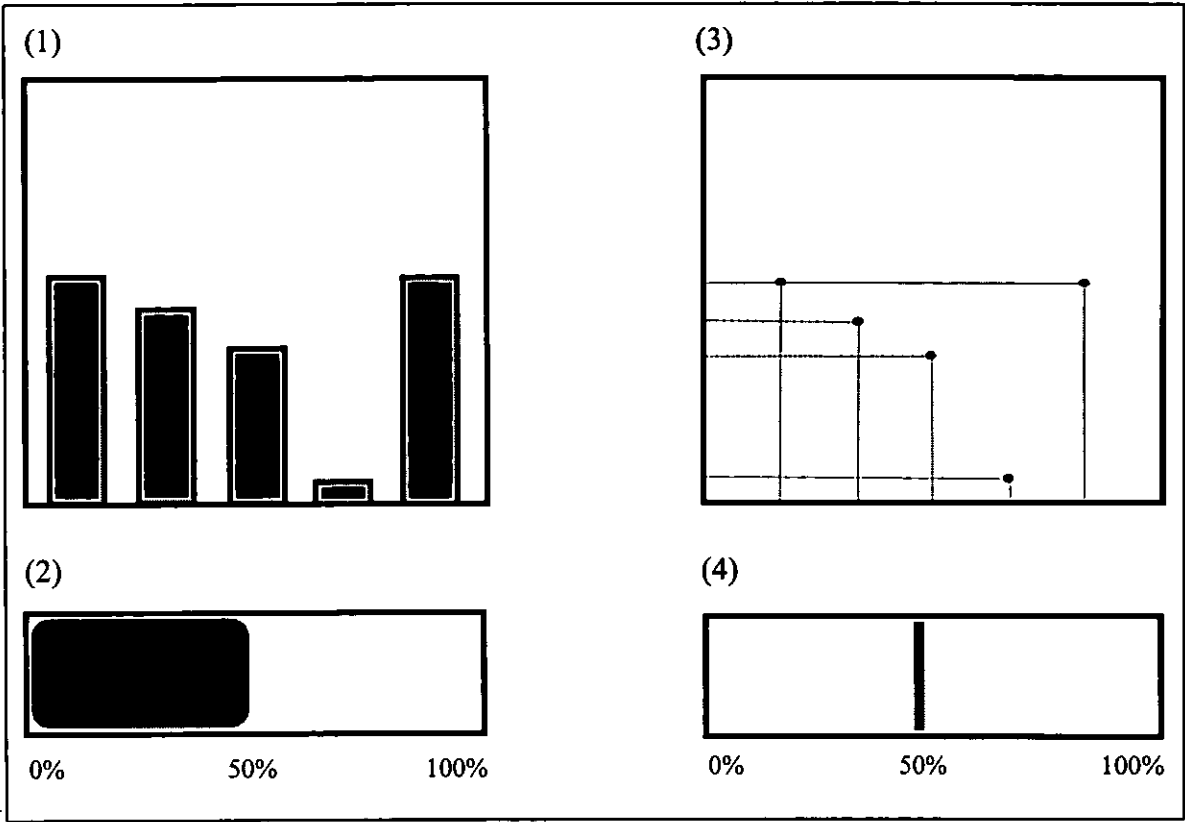


Figure 16: An instant example of perceptual enhancement: bars vs plain numbers

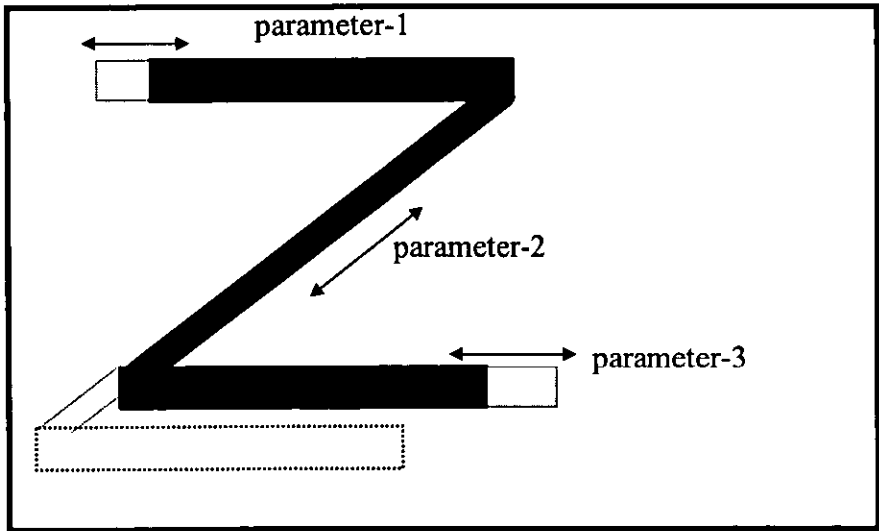


Figure 17: A visual Object with 3 degrees of freedom

not so apparent, this simple collection of ten objects summarises more data than ten conventional bar-charts. The density of the delivered information is significantly higher. Nevertheless, the role of FD-interfaces is only indicative. Their strong advantage is to rapidly indicate which particular parameters differ from the entire collection by utilising some, if not all, of the important laws of our perceptual organisation based on the constructivist theorists.

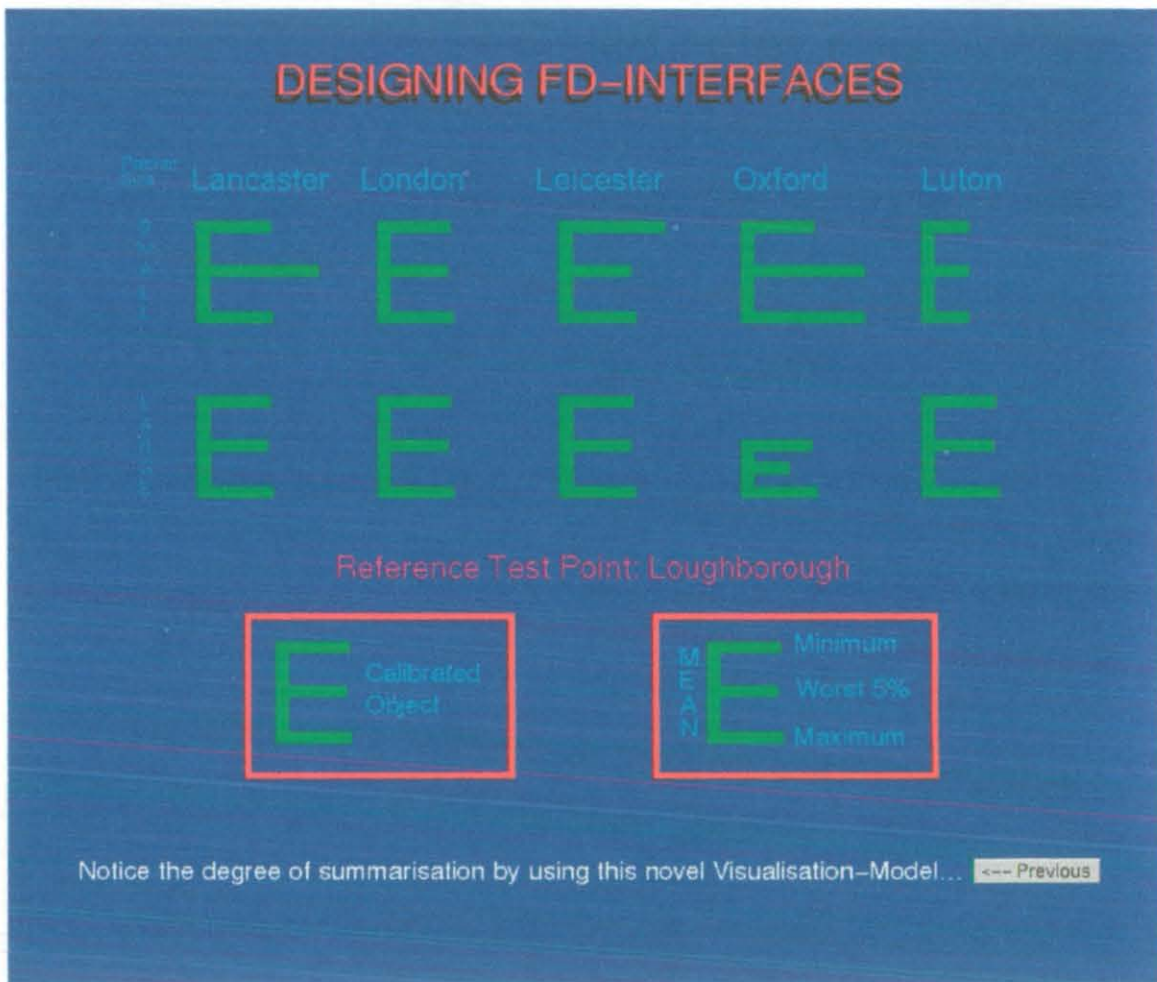


Figure 18: An initial conception of how to construct an FD-Interface

According to the constructivist approach, visual perception involves the intervention of representations and memories. In other words vision is not based only on a replica of the world such as a simple frame that a camera grabs but rather on a complicated model which involves transforming, enhancing, distorting and discarding information [Pr94].

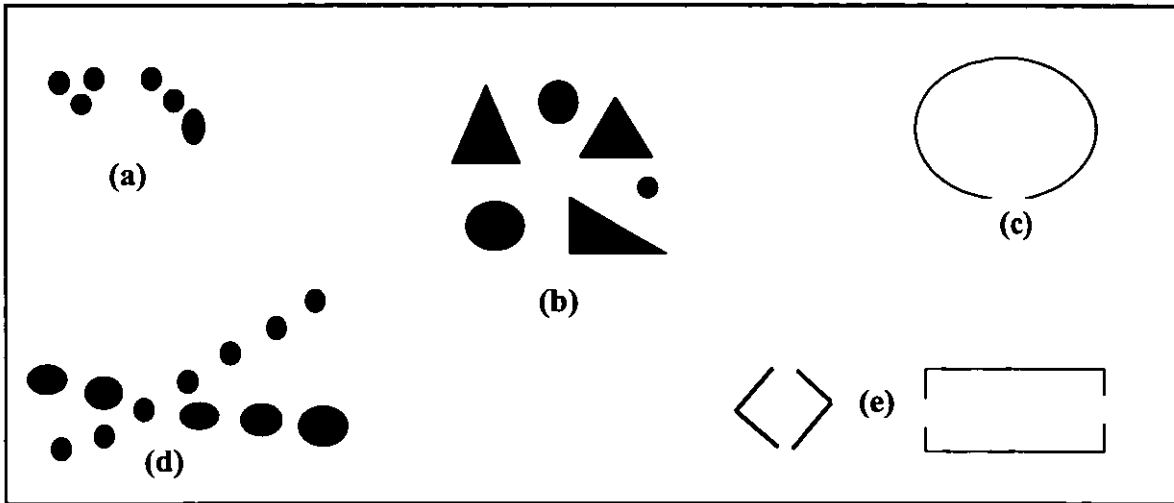


Figure 19: Various visual stimuli demonstrating Gestalt laws of perceptual organisation [Pr94]

Bearing the above in mind, let us elaborate on the five organising principles which enable the human brain to perceive the various patterns of our visual stimuli as meaningful wholes. These are *proximity*, *similarity*, *closure*, *continuity* and *symmetry* (also known as **Gestalt laws of perceptual organisation**).

Proximity describes the brain tendency to group (or categorising [Elg94]), where possible, elements that could have been perceived as separate entities. Consider fig.19a where the various dots are perceived as two groups of triplets despite their difference in size. This law enables FD-implementations to economise on working memory by allowing the user to perceive the entire collection of Es as a single group.

Similarity defines the tendency for elements of the same shape or colour to be seen as belonging together. Fig.19b demonstrates the notion. Notice how fast one can trace the similarities among various shapes. In the case of FDV, the law of similarity has been implemented by the existence of a calibrated object.

Closure portrays the brain ability to fill in missing parts of a figure in order to complete it (see fig.19c). With respect to FDV, because of this law, deformed objects are still perceived properly and not as a confusing set of moving elements.

Continuity exposes the ability to perceive discrete points as two continuous lines traversing each other (see fig.19d).

Symmetry describes the tendency for regions that are bounded by symmetrical borders to be perceived as coherent figures (see fig.19e). FD-interfaces combine the laws of continuity and symmetry by incorporating calibrated shadows that can be perceived as separate entities even if these might cover a large part of the FD-objects (calibrated shadows constitute a technique to visualise relevant thresholds; for more detail see chapter 5).

Figure 20 demonstrates a different FD implementation employing telescopic antennas to cater for seven parameters per network route (average packet delay of the fastest 95% or slowest 5% of the delays in the entire delay distribution, Std. Deviation, Average, Median, Min. and Max. Packet Delay). What might be appealing is the pseudo-mono-dimensional appearance of each visual object that allows great screen-space economy since deformations are only vertical with respect to the object's height. Therefore the object's width can be minimal. Representing the same amount of data by using traditional bar-charts (also called column-charts [Wa96]) would require the accommodation of one hundred and forty (twenty bar-charts each one having seven parameters) separate bars in a single screen. Furthermore, these bar-charts should be complete and not, as one often finds, cryptic with ill-defined scales, mysterious symbolism and decreased letter readability [Enr72]. They should also clearly illustrate their discrete variables by using appropriate bar-width to gap ratios (1/4 approximately) [Wa96].

Suffice it to say, the above bar chart scenario cannot yield acceptable results; the justification lies below:

A dot matrix of 7x9 pixels is the recommended minimum for legible characters in coloured CRTs [Lon84]. Therefore assuming a pixel for a normal 15'' display screen is approximately a 0.28 mm square [En96], a legible character must occupy an area of 1.96x2.52 mm (see fig.21). Given a common high resolution of 768 rows X 1024 pixels

[En96], we can conclude that no more than 14.6 characters should appear horizontally in any particular X-line of each bar-chart (assuming 10 bar-charts/line). Taking out the seven required gaps in the form of space characters in order to separate the labels, we are left with 7.6 available characters to define seven meaningful labels per bar-graph. Clearly, in this case accompanying text is not an option, therefore the bar-graph readability will be decreased. In a horizontal bar-graph scenario, the number of available characters would be even smaller (only 768 pixels are available) since most screen dimensions normally form a 3x4x5 right-angled triangle where 5 is the diagonal and 4 the corresponding screen width [En96].

Already having sacrificed textual help (by now, many usability experts would see no reason to continue this pointless scenario), we can safely assume the following:

To accommodate seventy bars in a single row in an acceptable manner would mean that one should allow **28.67 mm per graph**; that is without any gaps in between. From these, 7.16 mm should be sacrificed for spaces (bar-width to gap ratio 1/4 approximately) among the bars and only 21.50 mm would be the actual useful width implying that each bar could not be wider than 3.07 mm. Having this in mind, in order to achieve a 90% success in discriminating at least six colours, (red, green, blue, yellow, cyan and purple), screen items should subtend a visual angle of at least forty five minutes of an arc (45') [Lon84]. Therefore at a viewing distance of 500mm, the minimum width (or height) of an area to be properly colour discriminated should be $6.54\text{mm} \{ 500\tan(45') \}$. Clearly, the 3.07mm width previously achieved is by far smaller and as a result unacceptable. Therefore FDV can offer an increased screen-space economy compared to bar-charts.

Figures 22 and 23 show an effort to 'visualise reality' or 'realising virtuosity', that is supporting the experts (virtuosos) in doing best their duties [Pr94] by offering WYSIWYG tools. The WYSIWYG (What You See Is What You Get) concept was initially used to characterise products which were designed to offer consistence between the visual and printed output [Cla94] but now the concept is more broadly used for



Figure 20: An FD-Interface using a telescopic-antenna to accommodate 7 parameters.

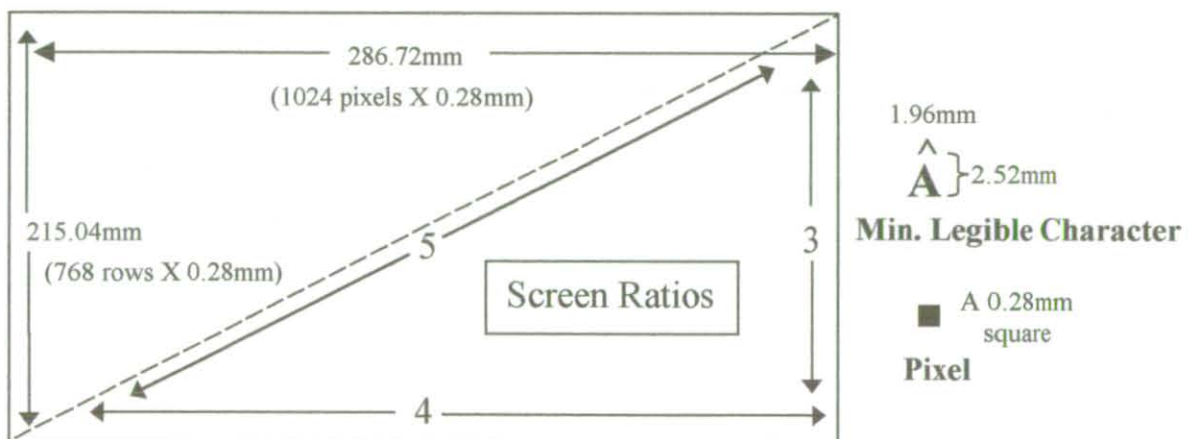


Figure 21: A standard 15'' computer screen

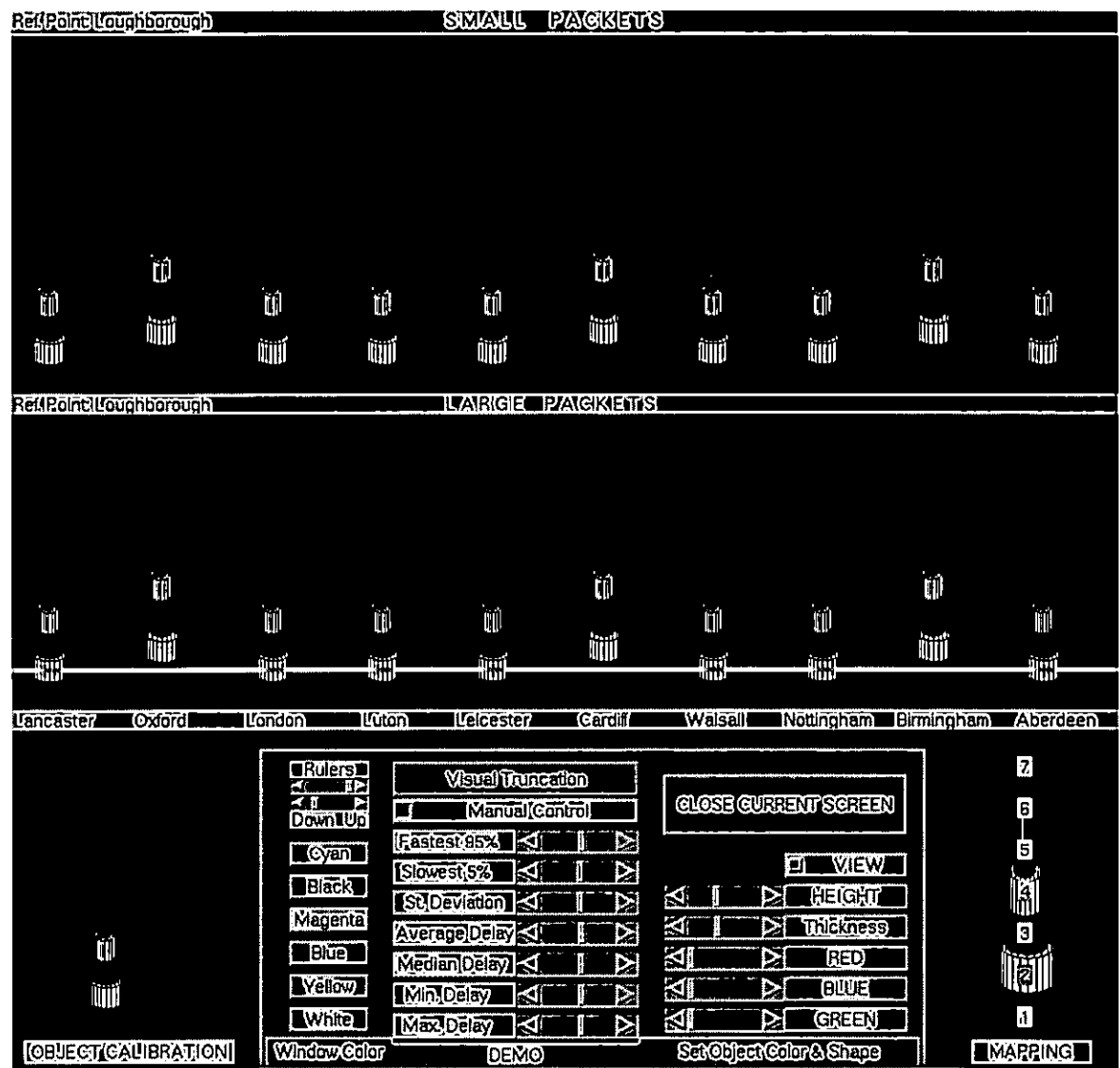


Figure 22: Towards a step closer to the ideal WYSIWYG Network Monitoring Tools

By appropriate adjustments to the background colour, it is possible to eliminate parts of the visual objects and therefore focus on particular changes. The current snapshot presents only Min. & Average Delays across the various Network Routes, whilst a set of rulers is also available to pinpoint sudden network changes with respect to the invisible parameters (i.e. Lboro-Oxford, Lboro-Cardiff and Lboro-Birmingham traces are not aligned with the remaining route-traces).

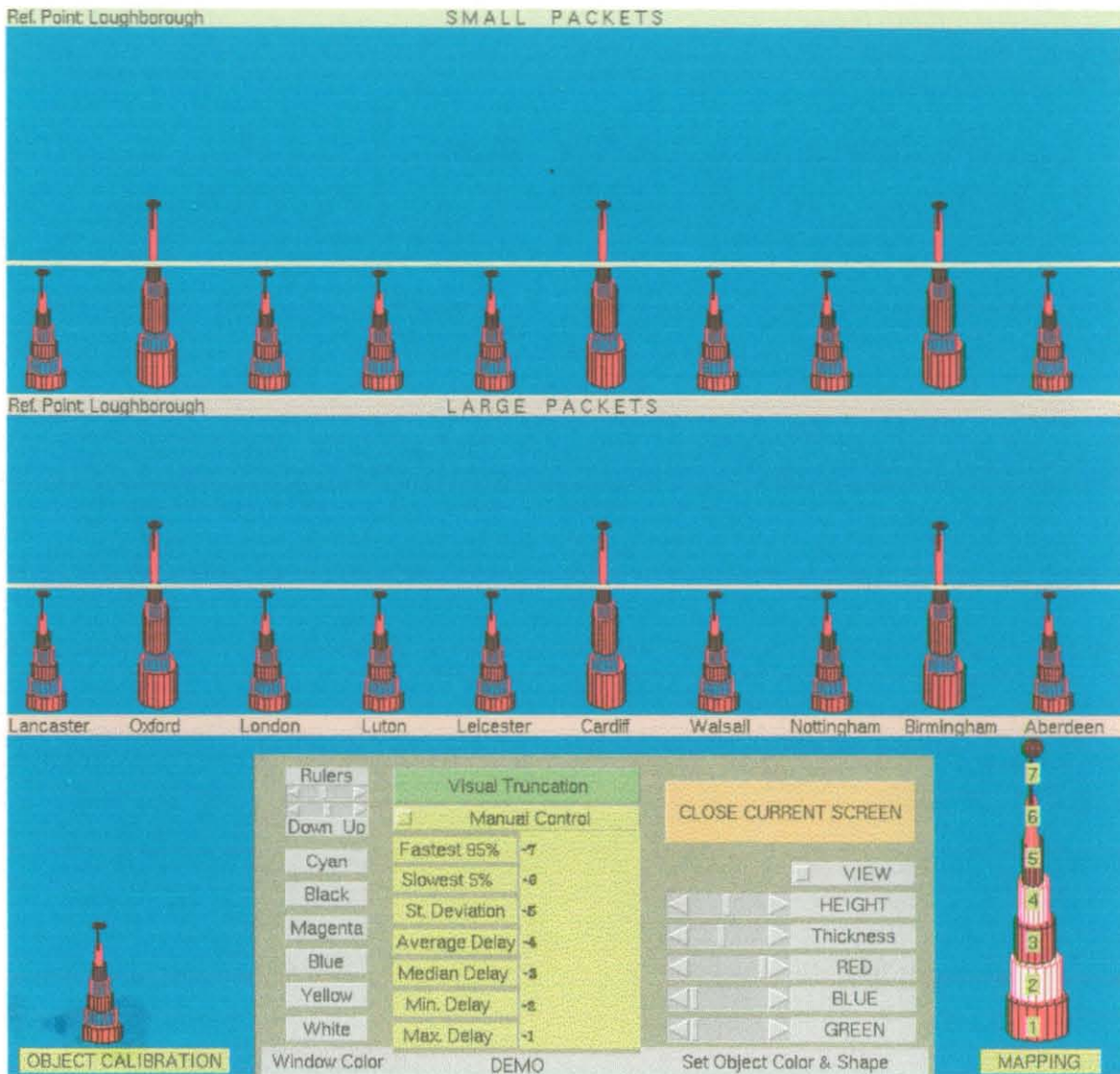


Figure 23: The current snapshot demonstrates the notion of *Visual Truncation*

As the term implies, after clicking on the relevant pushbutton, the associated algorithm truncates all input values which are below their corresponding thresholds, to their particular thresholds. Consecutively, the operator can know at a glance if something is wrong by simply looking for any stretched elements of the objects (i.e. Maximum, Median and Slowest 5% of Packet Delays are above thresholds in Lboro-Oxford, Lboro-Cardiff and Lboro-Birmingham routes)

describing smoother human-computer interaction using interfaces with immediate visual feedback [URL1].

A number of different constituents can affect the efficiency of FD-interfaces. One must decide on 1) the object-shape, 2) the number of included objects per screen, 3) the calibration-method, 4) the number of degrees of freedom, 5) the mapping of the various parameters, and finally 6) the colouring. All six can severely alter user-awareness and perception therefore further investigation follows.

Section 3.6. Conclusion

The main aim of this chapter is to demonstrate a novel way of designing effective pre-analytical interfaces for monitoring multiple variations. This is referred to as Figural Deformity Visualisation; the main idea behind FDV is to visualise multiple network parameters by employing a single object and gradually deforming its original shape. There is a great need to understand exactly how the notion of FD can be best used and this can be done only by conducting actual research with relevant user-groups. The important objective behind this effort is to invent and develop not only useful but usable network monitoring software tools which guarantee effective accomplishment of the required task by utilising the powerful cognitive abilities of the human mind [Rob99] to their maximum potential. The problem of showing too much detail relevant to the underlying execution mechanism or presenting information at an inappropriate level of abstraction has been already identified in other computer relevant areas (such as Parallel Programming [Pre96] or WWW Search Engines [Tan98]). The best separation among user-interface components and computation achieves the right amount of power in the interface, however achieving the ideal separation is neither always easy nor in fact always possible [Ha89]. What it is hoped this chapter has accomplished is to identify and tackle the same problem in the area of Network Monitoring.

Visualisation must involve much more than enabling users to simply “see” information [Rot97]. Instead it should allow them to browse at a glance and focus quickly on items of interest [Je96]. FDV is intended to provide this ability since it offers a well defined representation of network performance information.

CHAPTER

4

**Research Issues in Evaluating
the Effectiveness of
FD-Interfaces**

Section 4.1. Introduction

This chapter discusses some empirical research issues that need to be considered before attempting to evaluate the effectiveness of FD-interfaces in relation to network performance data pre-analysis. An indicative coding sheet is proposed as the main measuring instrument to be used for statistical data collection.

Section 4.2. The Scope of Empirical Research

According to Bausell [Ba86], *empirical research* is a term which is used to signify a method by which numbers are assigned to phenomena in order to actualise some sort of comparisons. Empirical research is both a **process** and a **product**. Being a process, it can be best described in terms of a discrete and relatively small set of behavioural attributes. On the other hand, being a product, it can be considered as a well-defined answer to a specific question. The term "*specific*" sustains a very important burden in the notion of empirical research. Exact questions can be expected to produce clear answers, however, general types of queries cannot be answered satisfactorily due to the fact that their genesis is quite complicated, involving many subsets of related questions that will eventually create "air-pockets" in the provided explanations. Although it is possible to categorise empirical research, the process and its product are completely generic to all possible categorisations. Apart from the specialised

knowledge required to successfully assign numerical data to various scientific matters, similar statistical procedures are still required to process the data itself (i.e. calculating the average packet-delay concerning a particular network performance datafile of delay measurements, would not require any different formulae than calculating the average life-span of a particular make of tyres, given a datafile of mileage measurements. In both cases however, collecting the required sets of data would involve totally different methods).

Section 4.3. Characteristics of Empirical Research

With respect to empirical research, there are three very important characteristics which need to be discussed:

- Its *verifiable* nature.
- Its *cumulative* nature.
- Its *finite nature* concerning the resulting product.

Every *valid* research result must be an outcome of a *verifiable* research process. Thus, every research process must involve a discrete series of steps that can be repeated by any other researcher who wishes to confirm the particular research findings. Mystical results can be very precarious when independent replication takes place.

Nevertheless, another very important research characteristic is the cumulative nature of empirical knowledge. The genre of research questions (also called *hypotheses*) that are addressed by empirical researchers have to be always as specific as possible. They can usually take the form of simple problems such as "*is X greater than Y?*" or "*is X related to Y?*". More complicated questions which cannot allow verifiable solutions are not to be answered by empirical research (i.e. philosophical problems).

Due to the fact that hypotheses are very straightforward, the provided answers are related only to the particular questions and, thus, the nature of the research outcome is finite. Utilising currently available results to answer modified hypotheses is not always scientifically acceptable. Researchers have to distinguish very carefully when finiteness, concerning a particular research product, allows or not for possible

tolerance when generalisations are attempted. Bausell, however, suggests that any kind of generalisation is not welcome and has to be avoided.

Section 4.4. Components of Empirical Research

The important steps that have to be followed, as part of the whole empirical research process, are analysed below [Ba86, Si78]:

Formulating the research question.

This step requires a general appreciation of the potential problem before continuing. Even the best chosen methods and most rigorous scientific techniques are useless without explicit justification of why have they been used in the first place.

Converting the research question into an hypothesis.

Once the important research question has been formulated, it must be converted into a declarative sentence which will clearly define what the study is all about. Such declarative sentences are called hypotheses and their advantages are: a) immediate reflectance of all the relevant variables and b) indication of the required type of research. Usually hypotheses are stated in the form either of *differences* or of *relationships*. Differences are usually assessed between groups; however, relationships are assessed between measures themselves.

Selecting an appropriate User Group to conduct the study.

There are a number of issues which need to be considered at this particular point. Firstly, it is very important to ensure that all prospective users reserve the right to refuse co-operation during the study. Secondly, users' safety and dignity must always be issues of major importance. Thirdly, the selected users must effectively represent the set of people for whom the study is most relevant. Finally, the number of users who will form the current sample must be statistically adequate.

Deciding the Research Strategy.

The questions which have to be answered at this point are:

- which user-attributes will the study address.
- how are these attributes going to be quantified.
- what possible type of error can the particular metrics incorporate.

- how can the potential error be calibrated.

Analyzing the gathered data.

Once the required set of data has been collected during the previous step, it must be subjected to a statistical analysis which finally should indicate if the hypothesis must be either rejected or accepted.

Section 4.5. Statistical Analysis

With respect to all previous sections, it is now appropriate to proceed into the actual statistical analysis that will reveal important conclusions concerning the effectiveness and usefulness of FD-interfaces. As the initial concern is to demonstrate the existence of potential advantages, the research scenario has been set in the form of the following *null hypotheses*:

- *"FD-interfaces do not enable network operators, on average, to respond faster to network performance data than typical visualisation-interfaces."*
- *"FD-Interfaces do not enable network operators, on average, to respond more effectively to network performance data than typical visualisation-interfaces."*

In order to allow the null hypotheses to be rejected, initially two groups of network operators must be formed. The first group (also called "control group") will be allowed to interact with conventional types of visualisation-tools while the second one ("experimental group") will only interact with the proposed one. However, both interfaces must interpret identical datafiles and furthermore must be equipped with similar timing procedures concerning human-responses when simulating alarming situations. The overall sample size should be at least 50 persons [25 operators per group] [Ba86].

Once the required data has been collected, it remains to be discovered if there is any difference between the two groups concerning their responses, and if such a difference can be declared as *statistically significant*. According to Bausell, the appropriate statistic to represent a difference between two groups on a single variable (assuming Mean and Standard Deviation are appropriate descriptors) is the **t-test** for

independent samples. The formula for the t-test used for groups containing different, unrelated subjects is given below:

$$t = \frac{\bar{X}_E - \bar{X}_C}{\sqrt{\frac{Var_E}{n_E} + \frac{Var_C}{n_C}}}$$

where:

\bar{X}_E = mean of the experimental group

\bar{X}_C = mean of the control group

Var_E = variance of the experimental group

Var_C = variance of the control group

n_E = number of subjects in E

n_C = number of subjects in C

Comparing the resulting values to a distribution of t-statistics will conclude the process by either rejecting or accepting the null hypotheses. The basis of the comparison will be established at the **0.05** level of significance which, is commonly used by most researchers [Tan98, Ba86].

Nevertheless, the previous formula cannot be used to indicate any relationship between two variables and since part of the analysis will require a more in-depth look at the potential constituents which could affect the efficiency of FD-interfaces, the need for another statistical tool is obvious at this point. What is proposed by many researchers, who have systematically manipulated scalar properties of measures [Ba86], is to employ the *Pearson Product Moment Correlation Coefficient* which is the single most important statistical application of the standard deviation (or variance) and arithmetic mean in empirical research. In its generic form, it is expressed as:

$$r = \frac{\sum Z_x Z_y}{n-1}$$

where

x = the first variable

y = the second variable

n = sample size

$$Zx = \frac{X_i - \bar{X}}{S_x} \text{ [z-score with respect to first variable] } *$$

$$Zy = \frac{Y_i - \bar{Y}}{S_y} \text{ [z-score with respect to second variable]} *$$

S_x = standard deviation of *x-variable*

S_y = standard deviation of *y-variable*

* [$i, i=1, \dots, n$]

* [\bar{X}, \bar{Y} = the arithmetic means of x and y respectively]

Suffice it to say, three minimal assumptions [Ba86, Bar79] are needed regarding Pearson's use:

- the arithmetic mean must be an appropriate descriptive statistic for the two measures involved
- the relationship to be investigated must not be curvilinear, otherwise there will be no indication of any relationship.
- if no relationship has been indicated, a curvilinear one may be present.

A more complicated form of the Pearson coefficient to detect partial correlation among three variables, is presented below:

$$r_{xy,z} = \frac{r_{xy} - r_{xz}r_{yz}}{\sqrt{1-r_{xz}^2}\sqrt{1-r_{yz}^2}}$$

where:

x = the first variable

y = the second variable

z = the third variable

$r_{xy,z}$ = the correlation between x and y with z partialled out

r_{xy} = the simple correlation between x and y

r_{xz} = the simple correlation between x and z

r_{yz} = the simple correlation between y and z

The variables to be exposed to Pearson correlation analysis will be the following:

- 1) *Object-shape*
- 2) *Number of included objects per screen*

3) *Number of degrees of freedom per object*

4) *colouring*

5) *mapping of various parameters*

6) *Calibration-method*

(At most three variables can be selected; whilst others remain constant)

Section 4.6. Designing the FD-Interface Evaluation Coding Sheet

4.6.1 Scaling Human responses

When it comes to sociological issues, the term 'scale' is often used as a synonym for 'indicator' [Si78]. An *indicator* is an empirical tool used to encapsulate a theoretical dimension. To demonstrate the above, in a series of questions a particular question is put in the list to identify information concerning the happiness of the particular respondent. This question can be considered as a happiness indicator. An indicator can have several elements in the form of related questions that are called *items*. Each item can be considered as a separate variable.

A scale may consist of one or more variables [Si78]. As a simple example, a driver's ability can be scaled based on a system of points allocated for each particular dimension of driving. One point can be allocated for parking, several points for his theoretical background, etc. The variety of scales that are designed to measure human responses is chaotic [Si78]. Simon has attempted to order them according to some characteristics:

- Measuring various types of mental activity
- Stimulus-Response Relationship
- Simple vs Composite Scales

Scales that measure mental activity focus mainly on perception, knowledge, intention and mental capacity. Perceptual scales concentrate on the human brain mechanics for differentiating various stimuli. Knowledge scales measure cognition in the form of exams usually met in educational institutions. Intention scales are used mostly in the area of advertising to measure consumers' behaviour with regard to their intention of purchasing a product. Usually they are simple questions seeking a 'yes'/'no' answer.

Finally, scales of mental capacity are the various IQ tests, involving challenging problems.

Stimulus-oriented scales focus on how various stimuli affect the same respondent. In such cases, the experimenter presents different graded stimuli in order to record the possible variation of responses by the same person. Nevertheless, it is possible to form the opposite scenario. In this case the experimenter presents the same stimulus to different respondents. This kind of research studies the similarities or differences among individuals by using response-oriented scales.

Although it is possible that some scales might be consisted of a single simple item, they might not reveal all relevant information. Consider an example in which someone is asked if he simply likes FD-Interfaces, being provided with a 'yes' / 'no' answer. No basic conclusions will be ever made regarding the accuracy or efficiency of the visual mechanism. Such a problem can be solved by employing composite scales which can enlighten the various dimensions of the response. Numerous composite scales are available, the most important of which are those of Guttman, Thurstone, Likert and Osgood [Si78].

After great consideration, it has been decided that the type of instrumentation to be used for measuring human responses relevant to variables 1,4,5 and 6 is the *5-points Likert-type scales* since they tend to give more stable results [Ba86]. Note that Likert-type scales are a special subset of *bipolar scales* consisting of statements such as that in fig.24.

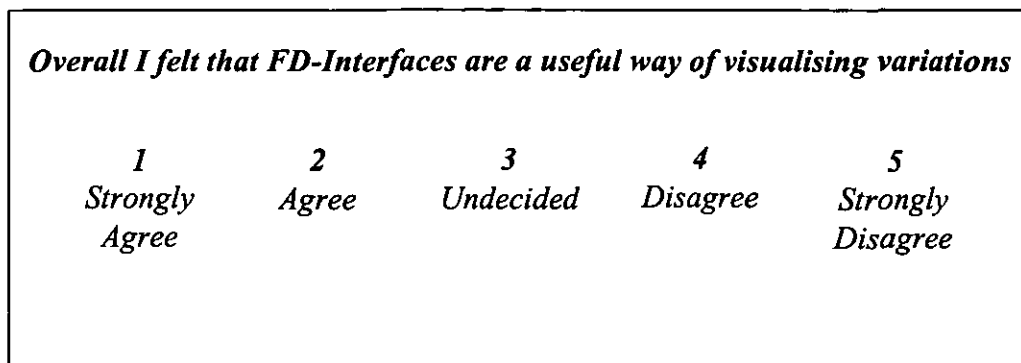


Figure 24: Likert-type scales

4.6.2 Coding Study Characteristics

The experiment was to be carried out on an individual basis. Although in theory subjects could be asked to participate concurrently, such a decision would demand numerous observers and computers, not to mention an increase in the observer-caused effects on the study (more on the issue is presented in *section 4.7*). Various characteristics of the research have to be coded in a proper way, allowing for quantitative descriptions [Smi79] which will produce refined answers to the particular research questions. With respect to the previous points, these are:

- ***The identification part.*** Some coding sheets include such a part mainly for allowing inferences with respect to complete data analysis or even meta-analysis in the light of newly gathered data. Since there is no need to place unnecessary restrictions in the gathered data of a study, unless these are related to ethical issues, the following items can be identified:
 1. Date in which the experiment has been conducted
 2. Sex of subject
 3. Age of subject
 4. Subject's computing experience
- ***Study Conditions.*** The coded items to be included in this part should identify the type of environment into which the study has been conducted.
- ***Outcome variables.*** All the relevant variables of this study are to be identified in this section of the relevant coding sheet. Two types of metric have been used and these are either simple gain scores which will be provided by the relevant developed software or Likert-type scores, to be generated from the included Likert-type scales.

4.6.3 General Questionnaire Construction Guidelines

Some key-elements to assist in the valid construction of the relevant FD-evaluation sheet are the following [Si78]:

1. ***Questions that do not need answers for the purpose of this study should not be included.*** Although there is great interest in gathering as much information as possible, one should define the limits.
2. ***The interviewee (subject) should find the organisation of the coding sheet logical and smooth.*** Submitting sloppy questionnaires with complicated questions will guarantee subjects' frustration and boredom.
3. ***Questions that could possibly affect the answers to others should be put afterwards.*** Nevertheless, in this particular study, much effort was expended in producing questions, which are not directly related.
4. ***The least important questions should be always at the end in case they do not get answered.*** This guideline is relevant to the second of this list. Suffice it to say, unless the respondent begins to regret for his decision to participate in the relevant experiment, this directive is not important.
5. ***Each question should be as short as possible and encompass simple non-sophisticated language.*** In a particular study, the observer should not forget that he is co-operating with subjects who might not possess the same scientific background as him. Therefore heavy scientific jargon might not be welcomed.
6. ***Open-ended questions should be converted to closed-ended form for increased reliability.*** Questions should be specific to a single particular dimension of the study.
7. ***The length of the questionnaire must be limited.*** Lengthy questionnaires might abuse the respondents' tolerance and kindness.

4.6.4 FD-Interface Evaluation Coding Sheet

Completing the current section, an FD-interface evaluation coding sheet is included in the following page. No assumptions have been made regarding respondents' previous experience with data-collection instruments. This particular survey instrument consists of two separate sections, the first of which is to be completed by every subject who will participate in the experimental group whilst the second part is intended for merely office use.

FD-INTERFACE EVALUATION CODING SHEET

PART I

Identification:

DATE ____/____/____

Sex of Subject: ____

Age of Subject: ____

Computing Experience: ____ Years.

Study Conditions:

____ Experimental setting ____ Regular working Environment (tick as appropriate)

Outcome Variables:

(circle as appropriate)

A. Overall, I felt that the object-shape does not alter the effectiveness of FD-interfaces

1	2	3	4	5
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

B. Altogether, the different colouring of the included objects did not alter my visual perception

1	2	3	4	5
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

C. I believe that the existence of a calibrated object eased visual detection of deformities

1	2	3	4	5
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

D. Vertical deformation of objects as opposed to horizontal one is perceived more easily

1	2	3	4	5
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

PART II (to be completed by the researcher)

Outcome Variables:

Number 1. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 2. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 3. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 4. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 5. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 6. Domain: _____	Metric 1: ____/____	Metric 2: ____/____
Number 7. Domain: _____	Metric 1: ____/____	Metric 2: ____/____

Domain: A = object-shape

B = coloring

C = calibration-method

D = mapping of various parameters

E = Number of included objects per screen

F = Number of degrees of freedom per object

Z = Response to alarming situations

Metric: G = Simple gain scores*

L = Likert-type scores

[* scores recorded by the associated software application relevant to each particular domain]

Section 4.7. Minimising Observer-Caused Effects

According to Simon [Si78], the researcher's efforts to examine and analyse a phenomenon always affect the particular study in which he is involved! That is because the researcher, taking the role of the observer, inevitably becomes a part of the environment in which the phenomenon occurs. The reader's concern will eventually be whether the particular effect is important, therefore altering significantly the outcome of the study, or if it simply can be overlooked, as is usually the case. A simple example to demonstrate more clearly Simon's claims can be a chemist's breath affecting slightly a reaction he is observing [Si78]. In the case of the FD-Experiments there has been considerable effort to minimise observer-caused effects by ensuring the following:

- *No variability among observers:* Even though variability among observers does not present a particular difficulty in principle [Si78], uniform user-trials cannot be guaranteed unless all observers follow the same 'ritualistic' practices (i.e. identical presentations to each individual before user evaluation, performed only either by the author or his supervisor in each and every instance).
- *Users have to be left alone:* Despite any user-frustration or user-request, the observer must not offer any more additional help than the standard already agreed.
- *All gathered data is equally important:* No coding sheet must be omitted from the statistical analysis. Once a person enrolls in the tests, unless he himself decides to quit, he will be allowed to complete it and therefore his results will be included in the study.
- *Estimations of Missing Data are not allowed:* Despite the fact that Simon [Si78] suggests that under many circumstances an empirical researcher may need a datum that is not available, in this particular FD-experiment this is not allowed to happen. It is the author's belief that estimations concerning missing data can be dangerous and biased. Therefore all required data must be validly gathered. The many devices for estimation of missing data often involve common sense [Si78]. Yet, common sense is neither always clearly defined nor scientifically acceptable.

Section 4.8. Test Equipment and Environment Considerations

Besides observers and subjects, there are other factors that can influence the experiment. Such factors are relevant to the equipment and environment in which the study eventually is performed. In order to keep them as constant as possible, one must ensure that:

- *The subject responses must be generated using identical equipment with similar settings.* One should not use different computer platforms even if running the same software application, otherwise the timings will be altered. Consider the case where a user is expected to produce timings using a conventional PC as opposed to a SUN terminal. To begin with, the relevant displays might be set to different resolutions. Furthermore, the two mice might move the pointers with different speeds (i.e. a conventional PC-mouse using ball-technology moves the pointer considerably faster than a SUN's optical mouse which is designed for increased accuracy when used with CAE packages).
- *Lighting conditions must be steady.* Poor lighting can affect user responses. Also, it might be interesting to mention that under normal daylight conditions (photopic vision) the eye is more sensitive to green and yellow spectral wavelengths while less sensitive to blue and red [Lon84].

Section 4.9. Some Sociological Considerations

The cornerstone of valid empirical research that sanctions trust to various outcomes, since they always assumed to be genuine, is related to Ethics. Nevertheless, although every scientific community operates on ethical premises, there is some danger that research results could be misinformative and inconsistent due to external factors, being admittedly difficult to control. Such factors can be related to sociological phenomena. To be more specific, many subjects during research experiments can give inconsistent answers [Si78] either because they want to please the observer and therefore deceive him or simply due to unexpected rationalisation and repression. Searching for the truth is not always easy, but fortunately psychologists have provided

specific guidelines to protect research outcomes from subjects who alter their behaviour during experiments. Before discussing some of the solutions, let us analyse the reasons for such unexpected behaviour.

Many subjects, when they participate in an experiment, do not wish to do what they have been asked to; instead they try to provide their own solutions to the problem [Si78]. However, by trying so, they alter their behaviour. The reason for this phenomenon could be that, unlike animals, humans always employ their own set of rules of thumb whenever they are forced to solve a problem.

Furthermore, Freud convinced sociologists that there is a conscious distortion of the truth. Much of this distortion is relevant to the various mechanisms of the human mind that are employed for recalling each time. He claimed that [Si78] distortion of truth happens merely for two reasons:

- The human mind comes into consciousness as a subconscious distortion of reality
- The human mind refuses to come into consciousness at all, because it is repressed

Even though much of the workings of the human mind are hidden, scientists still need to be able to study accurately a person's attitudes, desires or opinions in order to assist technological evolution. This can be achieved by employing behavioural psychology in order to identify possible instances in which respondents wish to provide deceiving answers. Behavioural psychology studies phenomena by watching the behaviour of subjects instead of simply believing subjects' answers based on their own distorted version of reality.

This study needs to take into account all these sociological issues and therefore its findings are to be based on the subjects' own answers as well as behavioural techniques (timings) that admittedly provide more objective results. The collection of two different sets of data (Likert-type scores as well as timings) in conjunction with careful observation of the subjects during the study ensures that the risk of collecting unreliable information is minimal. One point that might need clarification is that of deceiving answers. It should be clear that the adopted policy of this study regarding

subjects is that users are absolutely honest. Most timings are therefore expected to galvanise Likert-type answers instead of invalidate them.

Section 4.10. Some Unanswered Questions

It has been already said that Empirical Research is not meant to answer general questions that encompass philosophical aspects. That does not imply that Empirical Researchers do not have to answer philosophical questions during the course of their research. There are a number of issues that can lead to scepticism, some of which are listed below:

- ***How many variables to investigate?*** There are always some variables which are not going to be addressed during a particular experiment [Si78]. This happens because there is not the time or funding to investigate every single variable of any experiment. Besides, not all variables are promising enough to be worth the cost of being investigated. In this particular study, all the variables which are thought to play a fundamental role in the effectiveness of the FD-idea have already been identified. Variables such as screen-resolution, screen size, lighting conditions, machine input, noise etc, are not to be addressed since they are not considered relevant.
- ***Which object-shapes to investigate?*** The answer to this question may be very difficult. It is the author's belief that useful shapes which can provide a neat interface have to be less curved and more of a rectangular shape. Rectangles can be drawn adequately even with relatively low screen resolutions while they do not require complex calculations. They can also be deformed much more easily. Although FDV entails some artistic issues, the important prerequisite of any candidate object is its ability to perform effectively the required representations via deformation and not its appealing appearance.

Section 4.11. Conclusion

To conclude this chapter, the prominent goal of this analysis is to highlight some important issues with regard to conducting empirical research in order to estimate any potential benefit of FD-interfaces. This chapter discussed the scope and characteristics of empirical research while establishing the basic platform on which FDV experiments will be conducted and analysed. The main statistical tools to be used are the t-test and the *Pearson Product Moment Correlation Coefficient (PPMCC)*. The strategy to be followed at first is to analyse the collected data and hopefully prove statistical significance among the human responses that were generated using FD instead of conventional visual interfaces. Furthermore, PPMCC is to be used in order to discover any potential associations among the various variables that will assist in the formulation of the basic FDV principles. The chapter also considered sociological issues with respect to scaling human behaviour and presented the major instrument for the required data collection. Other various issues have been addressed as well, in order to ensure a well designed experiment that will provide adequate data to perform valid statistical inferences. Further action concerning the interaction with relevant user-groups is discussed in the following chapters.

CHAPTER

5

Prototype FD-Interface For Pilot Testing

Section 5.1. Introduction

The idea of Figural Deformity Visualisation was implemented into a prototype FD-interface for network performance display and used for preliminary testing with relevant user groups in order to provide data for a statistical assessment of the visual mechanism. The related program listing is written in MATLAB (version 4.2c for UNIX Workstations) and this decision was based on a number of reasons which are analysed in the following section.

Section 5.2. A Brief Introduction to MATLAB

As its creators put it [Mat96], MATLAB is a technical computing environment for high-performance numeric computation and visualisation. It integrates numerical analysis, matrix computation, signal processing and graphics in a user-friendly environment. The name MATLAB stands for Matrix Laboratory. MATLAB was originally developed to provide an easy access to matrix computation. It can be characterised as an interactive

system whose cornerstone is a matrix which doesn't require dimensioning. Compared to traditional programming languages such as C, Fortran, Pascal and Basic, MATLAB seems easier to use for mathematical computation.

Moreover, MATLAB features a set of application-specific solutions that are called **Toolboxes**, which are nothing more than comprehensive collections of script and function files, being able to extend MATLAB's environment to solve specific technical problems that would take a lot more time and effort to be solved in a traditional way [Mat96]. MATLAB is available for Unix and PC. The differences between the two versions are almost negligible and practically once an application is written, it can work successfully in both platforms with minor changes. Once MATLAB is invoked the user is presented with the command window. Any valid command can be typed and executed immediately from the prompt, since MATLAB's workhorse is not a compiler, but instead, a very fast interpreter which performs some basic form of compilation. Its creators characterise MATLAB as a compiler/interpreter technical environment [Mat96]. The command window itself can be edited but for long programs a separate editor must be used. Once a file has been typed, it can be executed by just calling it from the command window. However all MATLAB files must bring the extension **“.m”**. Due to this extension, such files are called **M-files**. An M-file consists of a sequence of normal MATLAB statements that possibly include reference to other M-Files. There are two types of M-files: **scripts** and **functions**. Scripts automate long sequences of commands. Functions provide MATLAB extensibility by allowing the creation of new functions. Additionally, while in scripts arguments cannot be passed, this is possible with functions. Another difference among them is that all variables declared in scripts have a global context and every other script following the present one can access them. On the contrary, function variables are mapped as local and once the function is executed they disappear from the workspace.

A third type of MATLAB files are the so called **“MAT”** files which are full precision binary MATLAB format files created upon the programmer's request to save the current workspace and relevant variables. Two MAT-files have been created and used for the animation purposes of this application.

Section 5.3. Program Presentation

The format of the created application is based on the idea of having two separate groups of users, the experimental and the control group (as already suggested in the previous chapter). The main console (fig.25) is equipped with two buttons which invoke various windows to provide additional information about the purpose of the whole research. These are the “DEMO” and “About Figural Deformity” (figures 26, 27 & 28) which summarise the main idea in a few lines of text and offer a brief demonstration so that users of both groups can have a general understanding about FDV. “Help” windows (i.e. fig.29) are also available in all sections of this program.

5.3.1 Control-Group Assessment

The relevant tests for the control-group users are designed to measure their responses with regard to network alarming situations, by employing current ways of visualising multiple variations (i.e. bar graphs). Once the appropriate pushbutton from the main console has been clicked, the user will be presented with 10 empty axes and an additional control panel with pushbuttons to either enroll in the testing, receive help or quit and return to the main console. The structure of the test is rather straightforward. Users were presented four consecutive times with ten different bar graphs. Each bar graph represents a particular network route (fig.30) and visualises four network parameters which are relevant to the Service Level Agreement (S.L.A) and namely are 1) the *maximum* packet delay, 2) the *minimum* packet delay, 3) the *average* packet delay and 4) the *average slowest 5%* of packet delays. The task of the user is defined as following: once he is presented with a new screen, he is expected to study all the relevant presented information as quickly as possible and click the “ready” pushbutton. Immediately one checkbox will appear next to each route and the user will be expected to identify and mark all unhealthy routes by simply clicking the respective checkbox. Once he completes this process, he has to notify the computer by clicking on the “Finished with Current Screen” pushbutton.

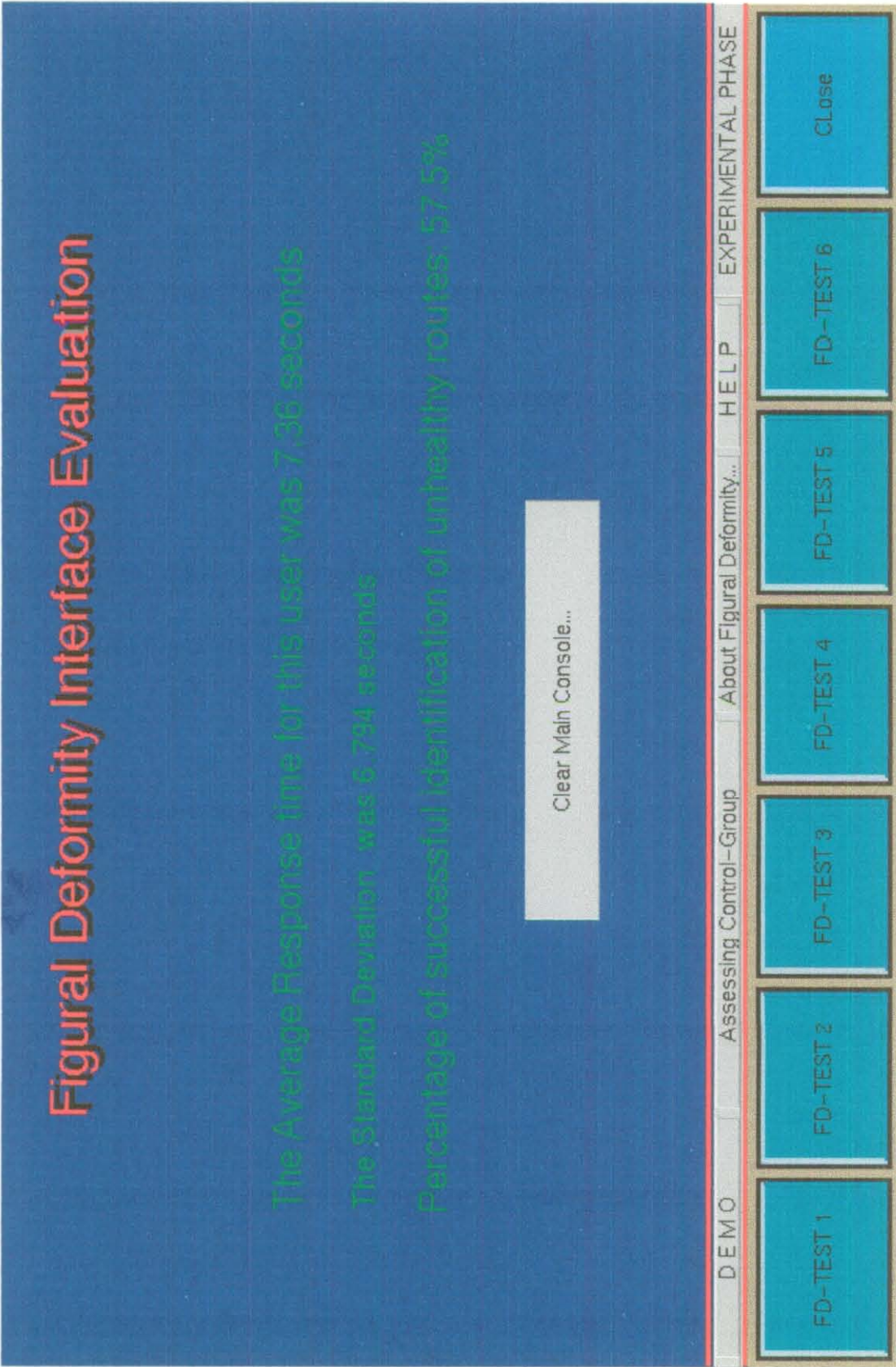


Figure 25: Main Console

Extra care has been taken in order to gather the relevant time measurements as validly as possible. First, the axes marking has been done manually and the auto-mode has been disabled. This enabled ticking of the threshold values only (nevertheless, a "Grid" button is available if the user should require further visual help). Second, the timing function is

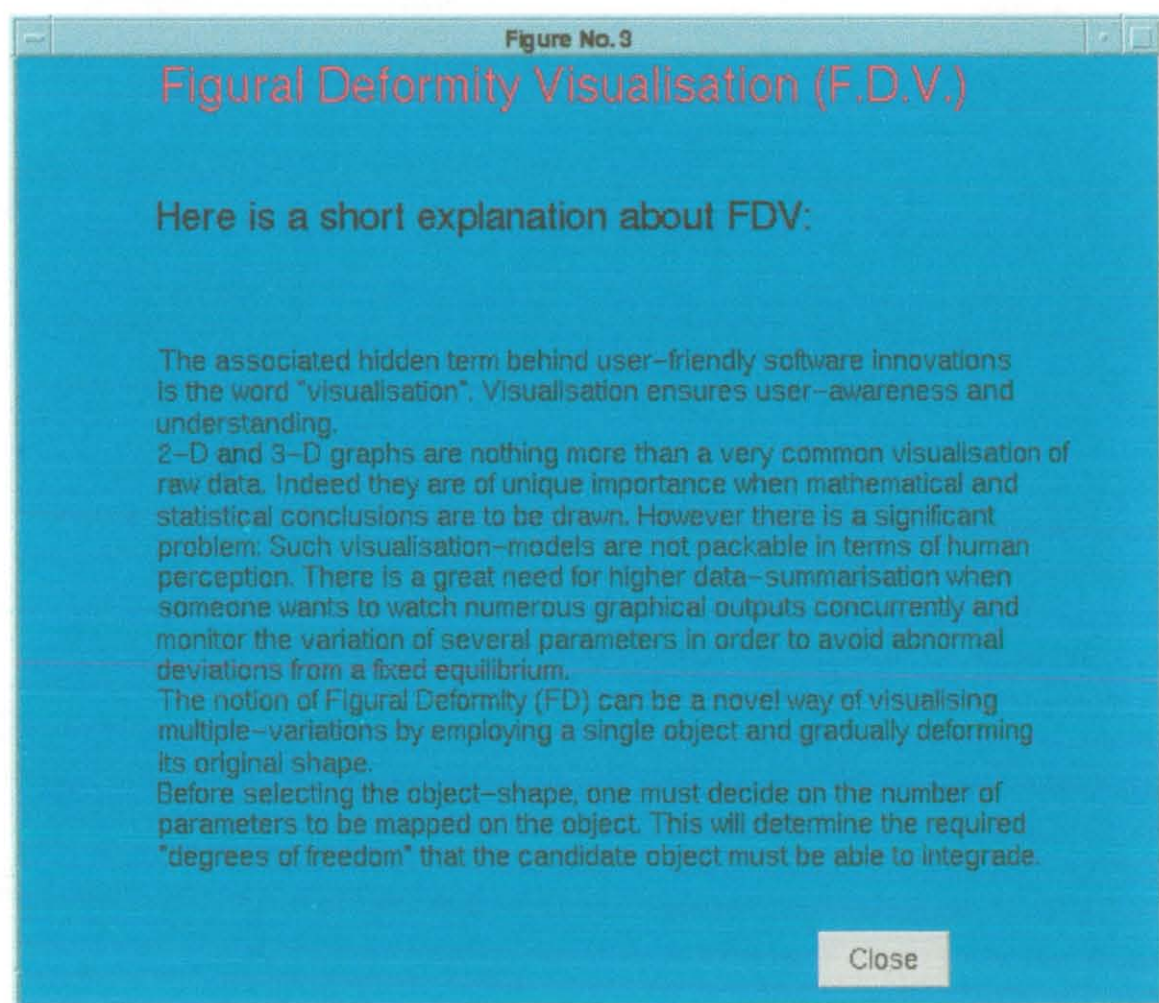


Figure 26: A short explanation about F.D.V.

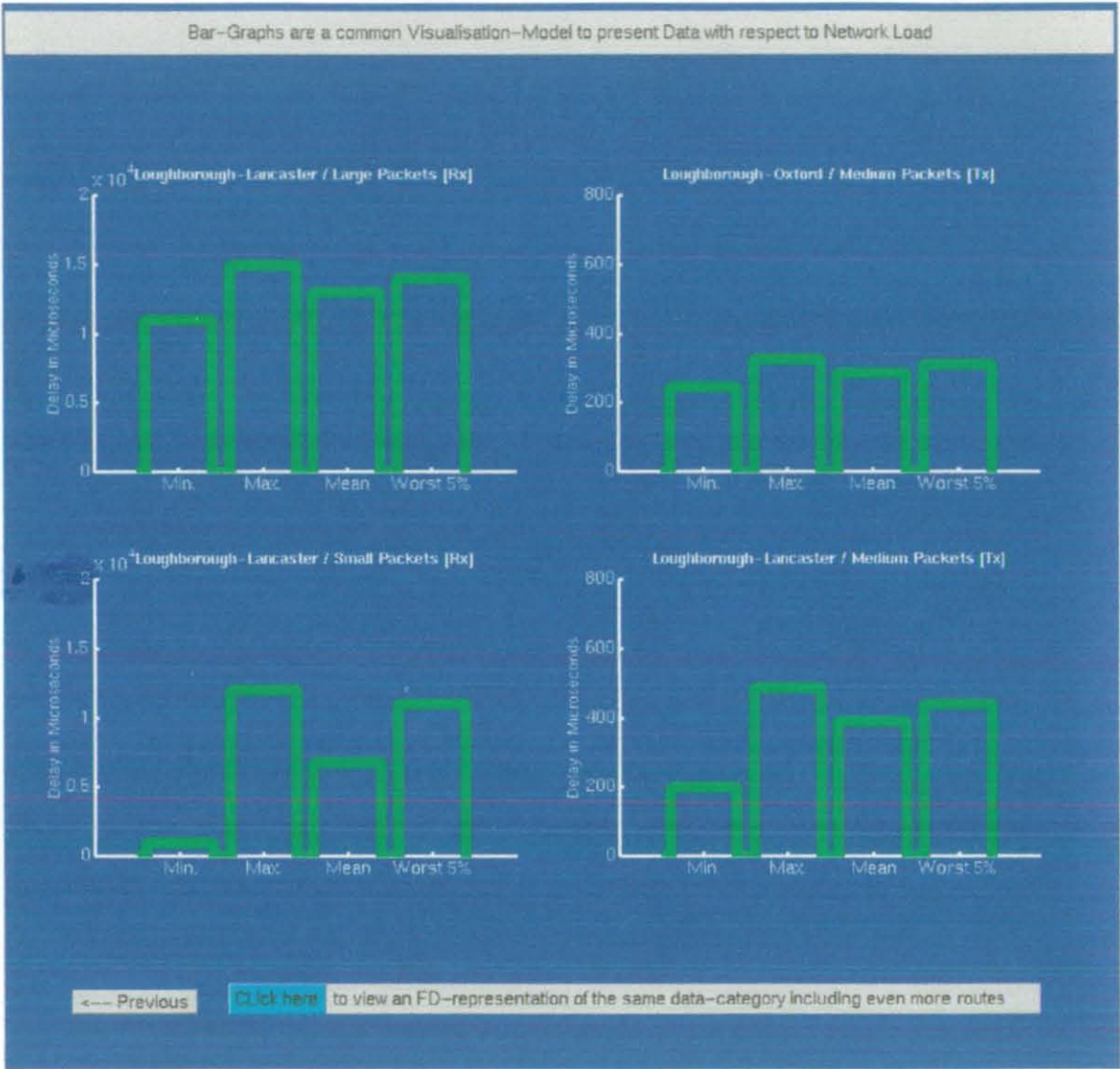


Figure 27: Demo Screen-I

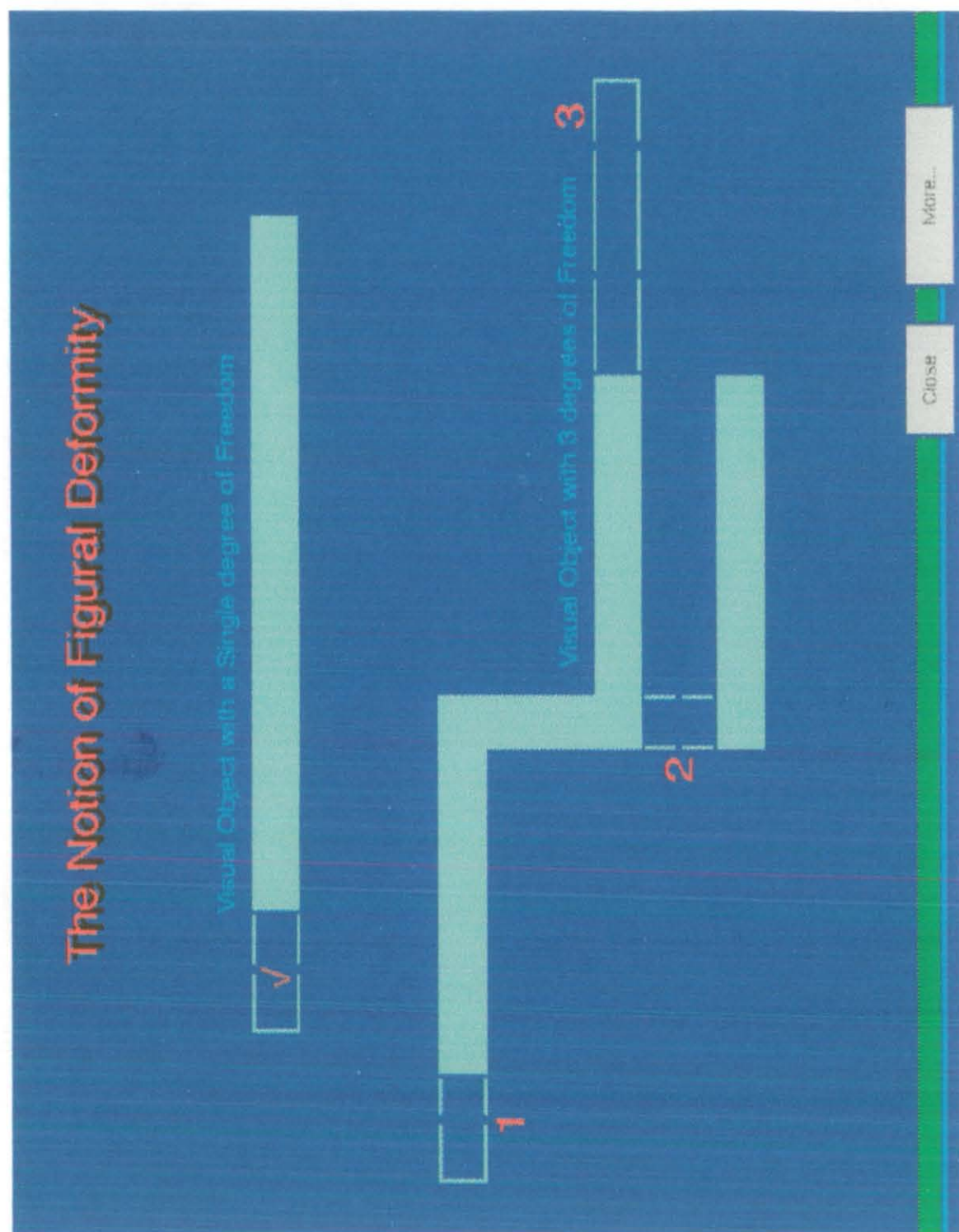


Figure 28: Demo Screen-II

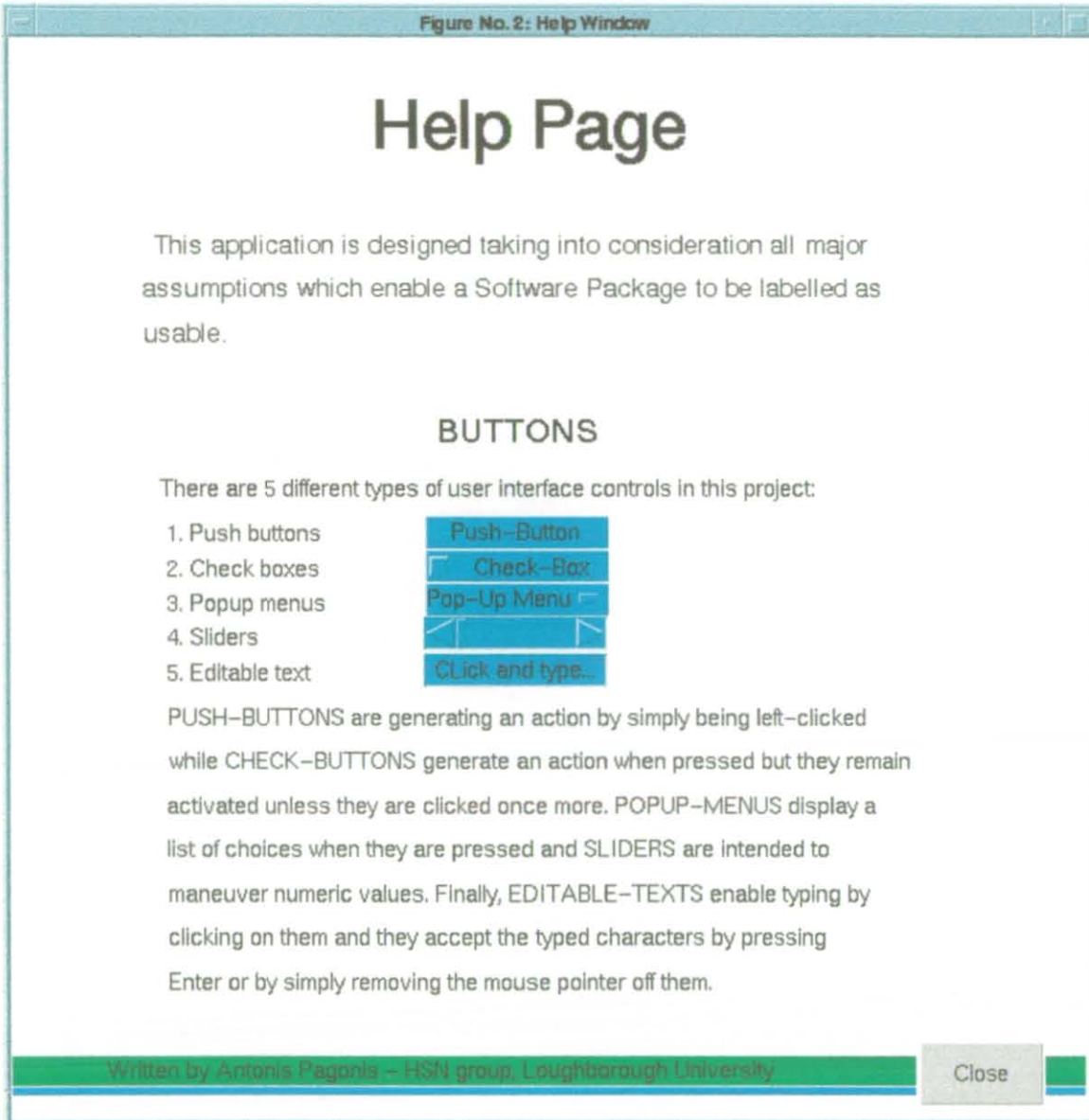
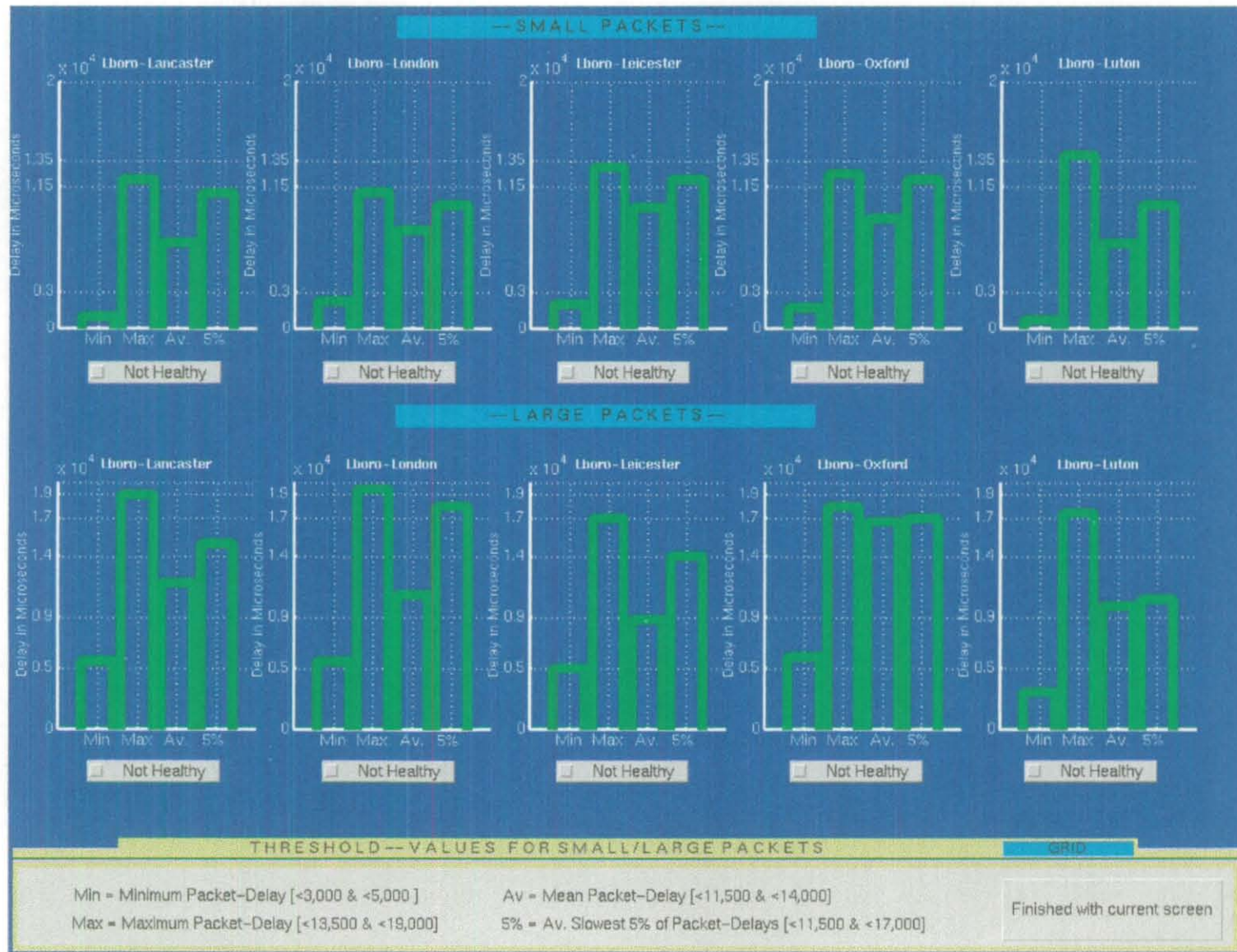


Figure 29:An example of the Help facility

Figure 30: Control-Group Screen



performed by using the MATLAB's **tic-toc** stopwatch timer command which is reasonably accurate and does not suffer from overflow situations like other MATLAB's timing commands (i.e. the **cputime** command whose return value may overflow the internal representation and wrap around [Mat96]). Third, the actual time measurement for each particular user is calculated by averaging the four separately gathered responses. Additional information is collected with respect to the standard deviation of the four responses and the percentage of successfully identified unhealthy routes.

5.3.2 Experimental-Group Assessment

In order to strengthen the validity of the comparison between FD-objects and Bar-graphs the structure of the tests for this group is identical to the previous one. As a result, the time measurements are collected after averaging the four separate responses of each user and the data to be represented in both cases is comparable. However, the objective in this case is not only to gather data for the direct comparison of the two groups, but to perform a correlation analysis of the six variables suggested in the previous chapter which as already mentioned are 1) the object shape [outcome variable **A**], 2) object colouring [outcome variable **B**], 3) calibration-method [outcome variable **C**], 4) parameter-mapping [outcome variable **D**], 5) number of included objects per screen [outcome variable **E**], 6) number of degrees of freedom per object [outcome variable **F**] and 7) response to alarm situations [outcome variable **Z**]. In order to collect the required data six different FD-Tests have been implemented and will be analysed separately below:

- ***FD-TEST 1:*** This is the most fundamental test which allowed collection of data with respect to variables C and Z. The test includes a carefully designed interface (fig.31) having most of the user interface controls in the right side and allowing for a large display area to accommodate the relevant FD-objects. The object calibration can be performed by using the relevant sliders to adjust object length, height and thickness while the combinations of colouring are almost infinite due to the fact that it is possible to control the three basic display colours (red - green - blue) separately and therefore, to create every possible chromatic combination.

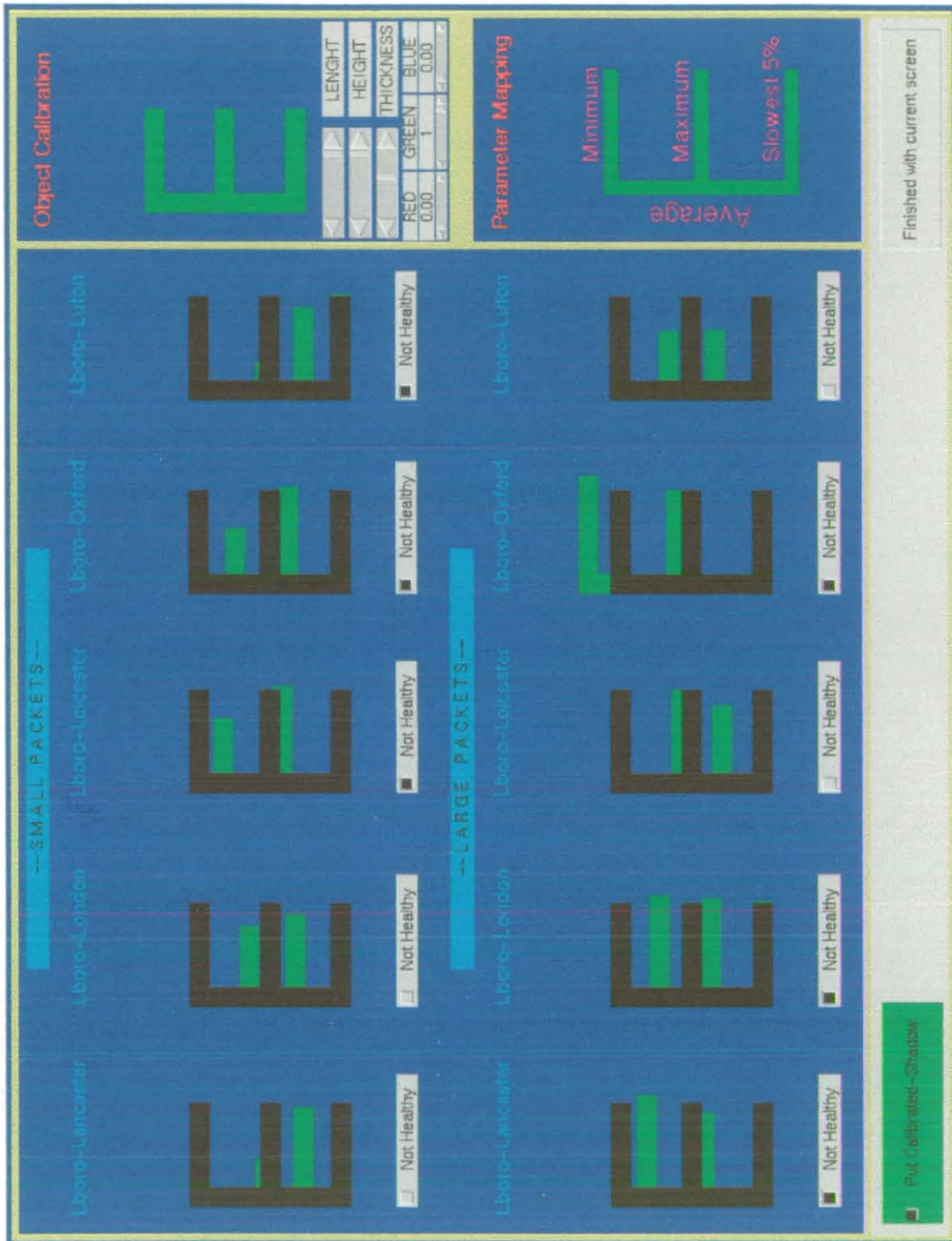


Figure 31: Experimental group (FD-TEST 1)

If the FD-object deformations are not easily seen, it is possible to click on an additional button which draws calibrated shadows on top of the visual objects. This action compensates for the lack of *grid*, an aid which is offered only during the control-group assessment. Calibrated shadows are basically a reproduction of the calibrated object, directly drawn on top of every deformed item. Their use enables the user to visualise the actual thresholds and therefore are meant to offer an instant appreciation of the network situation.

- **FD-TEST 2** : The test is relevant to outcome variable B. The aim of the trial is to present the user with four different screens on which the colour of the respective FD-objects will be red, green, white, and yellow (fig.32). The selection of the colours is arbitrary since the goal is simply to examine if the change of colour has any effect on user-responses with respect to time needed and integrity of the answers.

- **FD-TEST 3** : The aim of this section is to accumulate data for the outcome variable E. The number of FD-objects per screen was increased to twenty (fig.33). However only ten FD-objects among them are potentially deformed since the purpose of the test is to examine any potential correlation between the time required for a particular user to trace a set of intolerably deformed objects, and the actual size of the entire FD-collection. The adopted procedure to record the unhealthy routes is similar to the previous tests.

- **FD-TEST 4** : This test is associated with the outcome variable A. The user is presented with two different FD-object shapes: 1) The familiar by now “E” shape and 2) the “square” shape (fig.34). The number of visual objects per screen is still twenty but only a single object is deformed from the entire collection. The task of the potential user will be to look carefully and once he has traced the deformed object to click on the “Ready” pushbutton. Special effort has been given in order to ensure the same degree of deformation between the two different shapes.

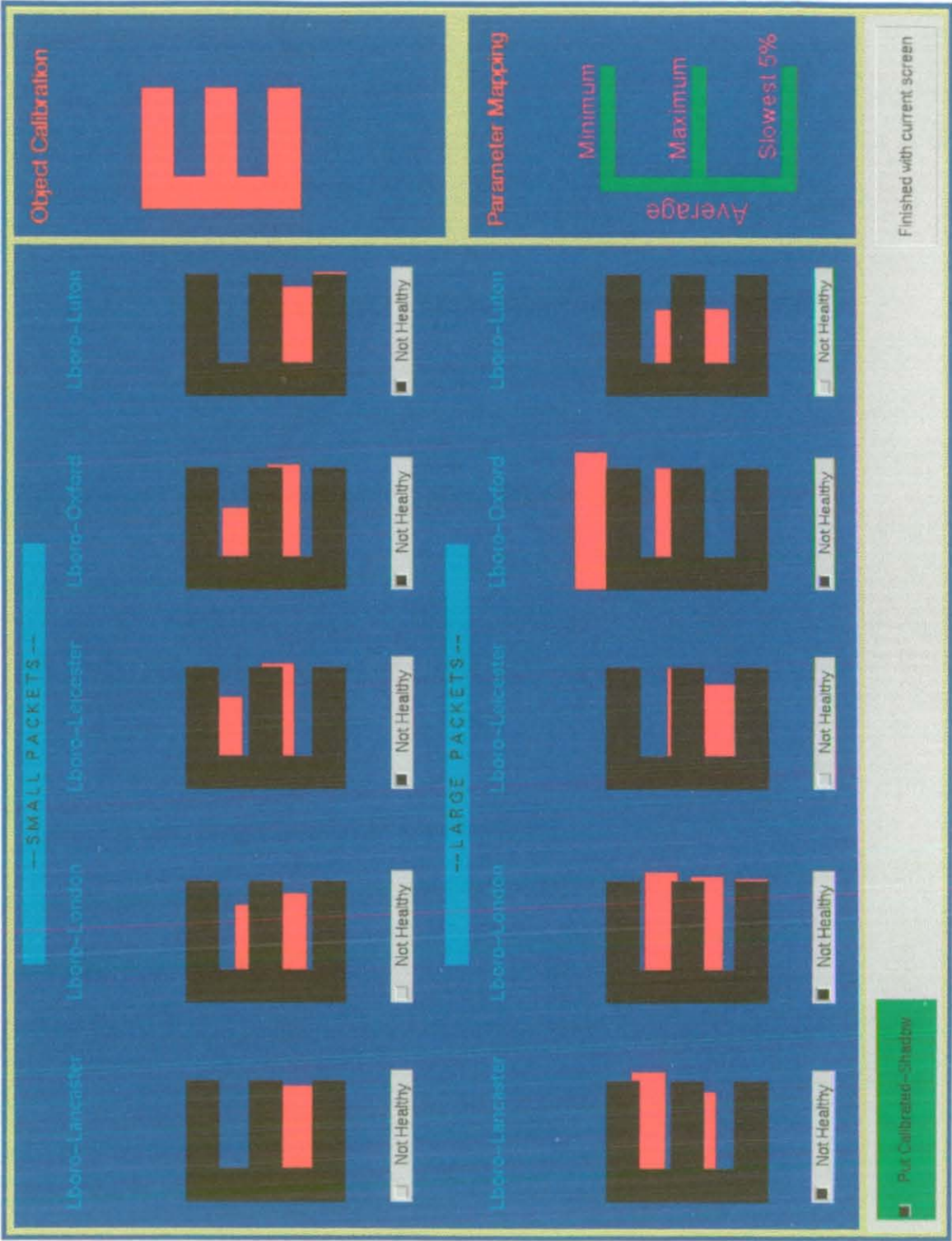


Figure 32: Experimental Group (FD-Test 2)

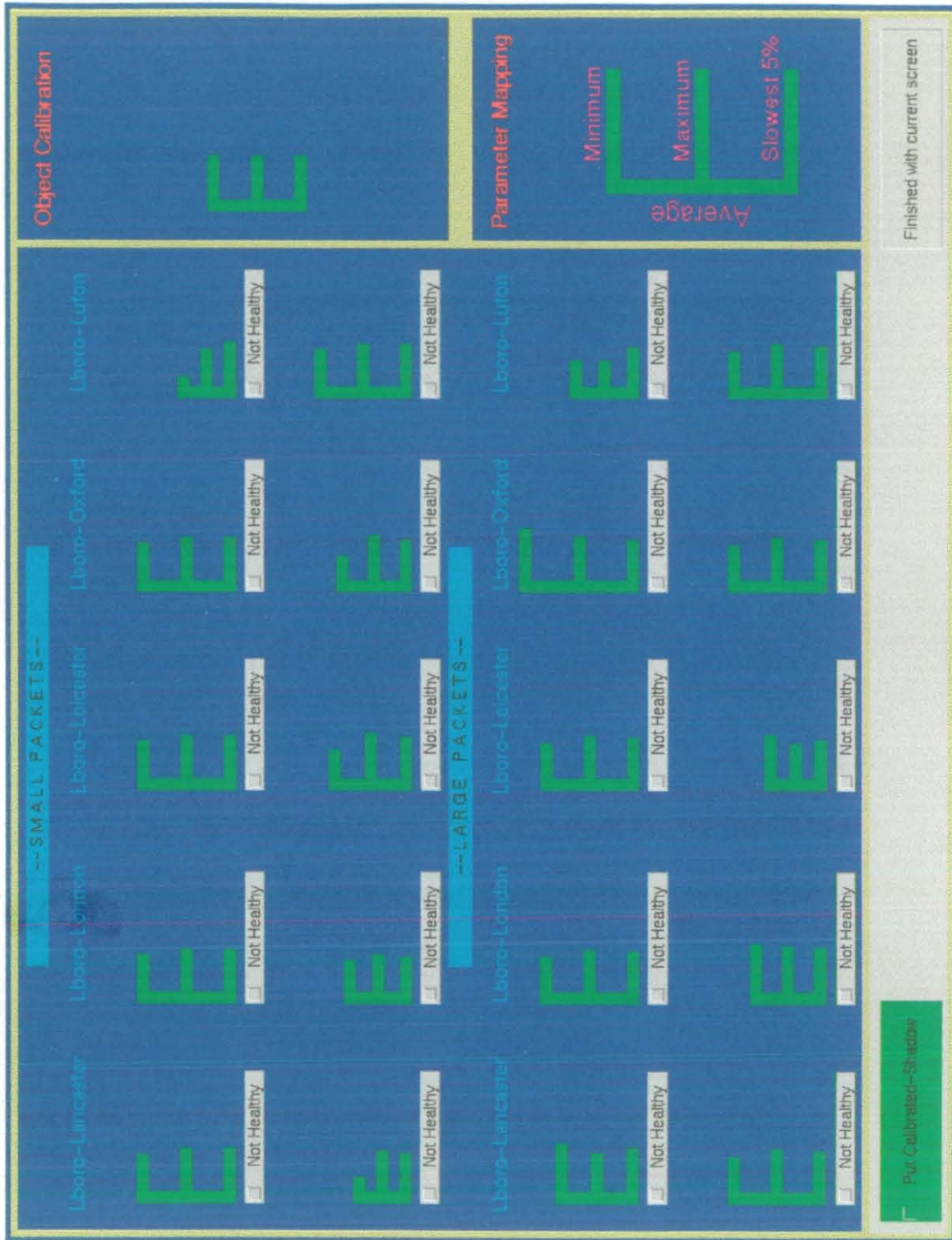


Figure 33: Experimental Group (FD-Test 3)

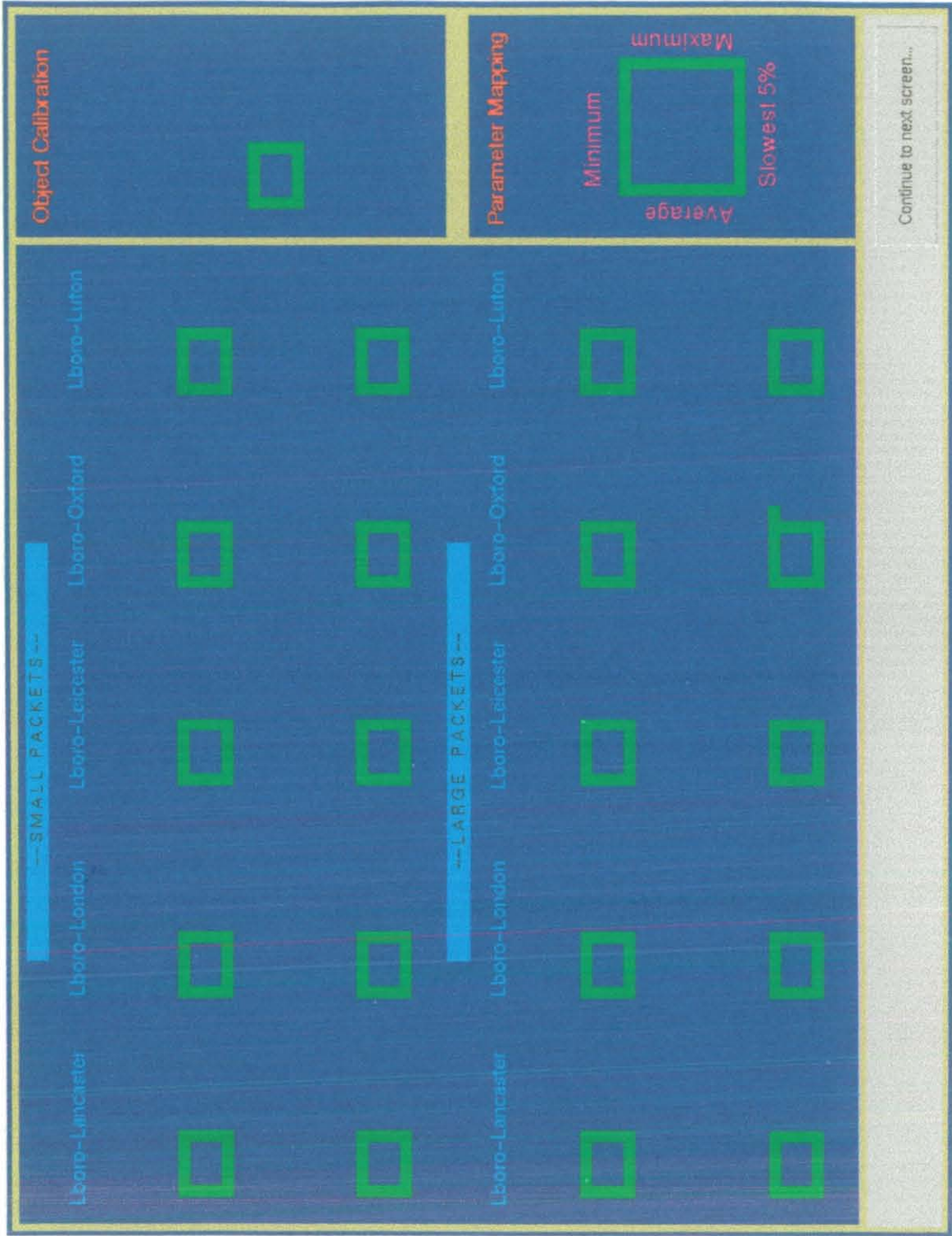


Figure 34: Experimental Group (FD-Test 4)

- ***FD-TEST 5*** : This section is potentially the most demanding one in terms of visual capabilities and is linked to outcome variable D. The trial consisted of two parts in which an animated FD-object of shape “E” is deformed extremely slowly with respect to its height and length. The user must watch carefully and once he suspects that the object is deformed either vertically or horizontally, he is expected to click on the relevant pushbutton. The purpose of the test is to directly compare vertical to horizontal deformation. The conclusions will help to establish some basic rules of hierarchical (instead of arbitrary) parameter mapping.
- ***FD-TEST 6*** : The measurements of this final test are relevant to outcome variable F. The test incorporates two different object shapes (fig.35 & 36) with eight and fifteen degrees of freedom respectively. Users are expected to trace the exact part of the FD-object which has suffered deformation, without the use of any calibrated shadows. The relevant gathered data will be processed in order to investigate any potential *statistically-significant* difference between time responses and number of degrees of freedom per visual object.

Section 5.4. Program Design Issues

Although the most important M-files are included in the Appendix-B, it would be useful to elaborate on specific programming issues at this point:

- All the created windows have properly defined *figure handles* and the automatic assignment to “ans” variable has been purposely avoided in order to keep the software code as neat as possible.
- All the created windows cannot be re-sized. This is done to control the experiment and to avoid accidental visibility fluctuations.

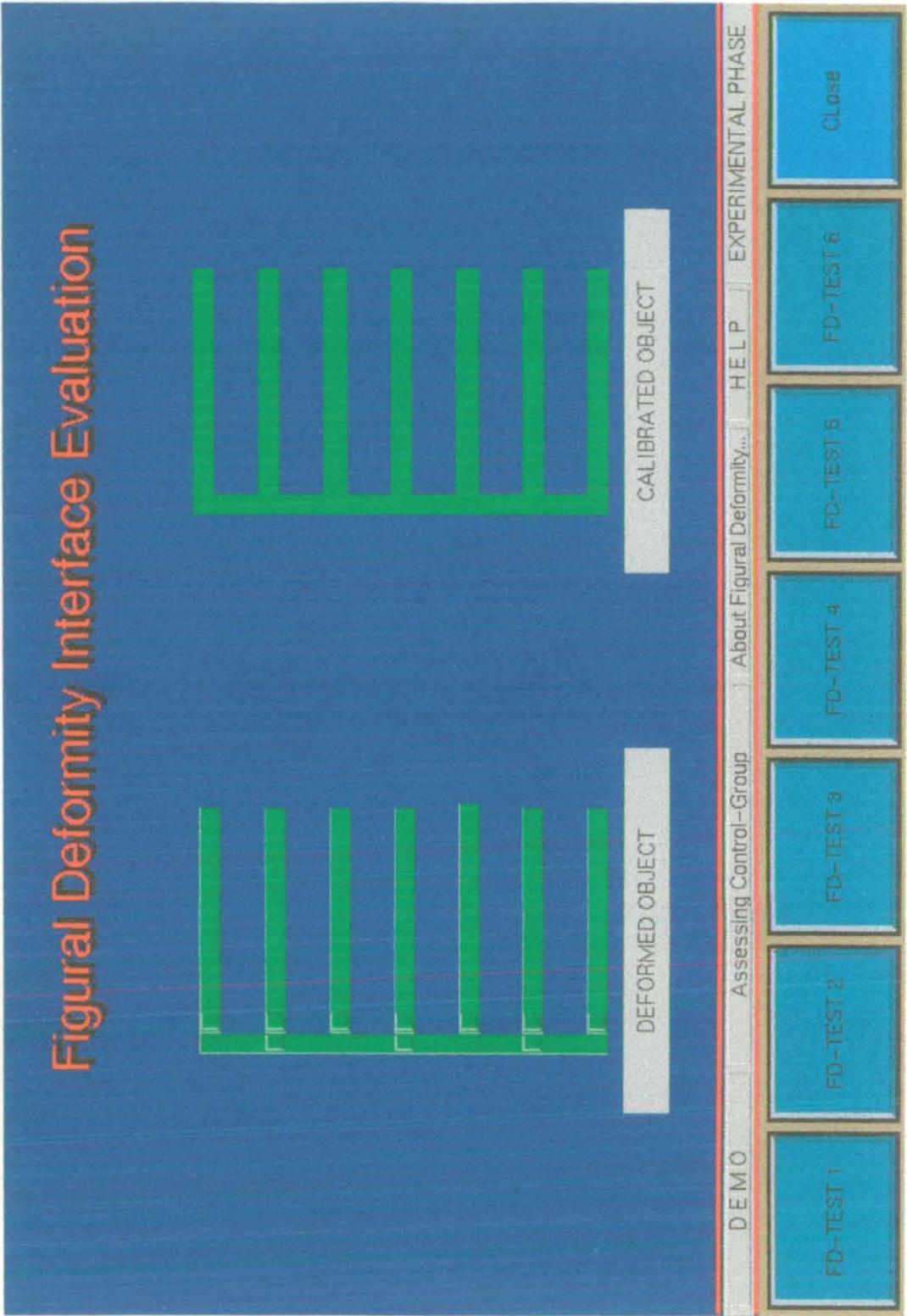


Figure 35:Experimental Group (FD-Test 6)

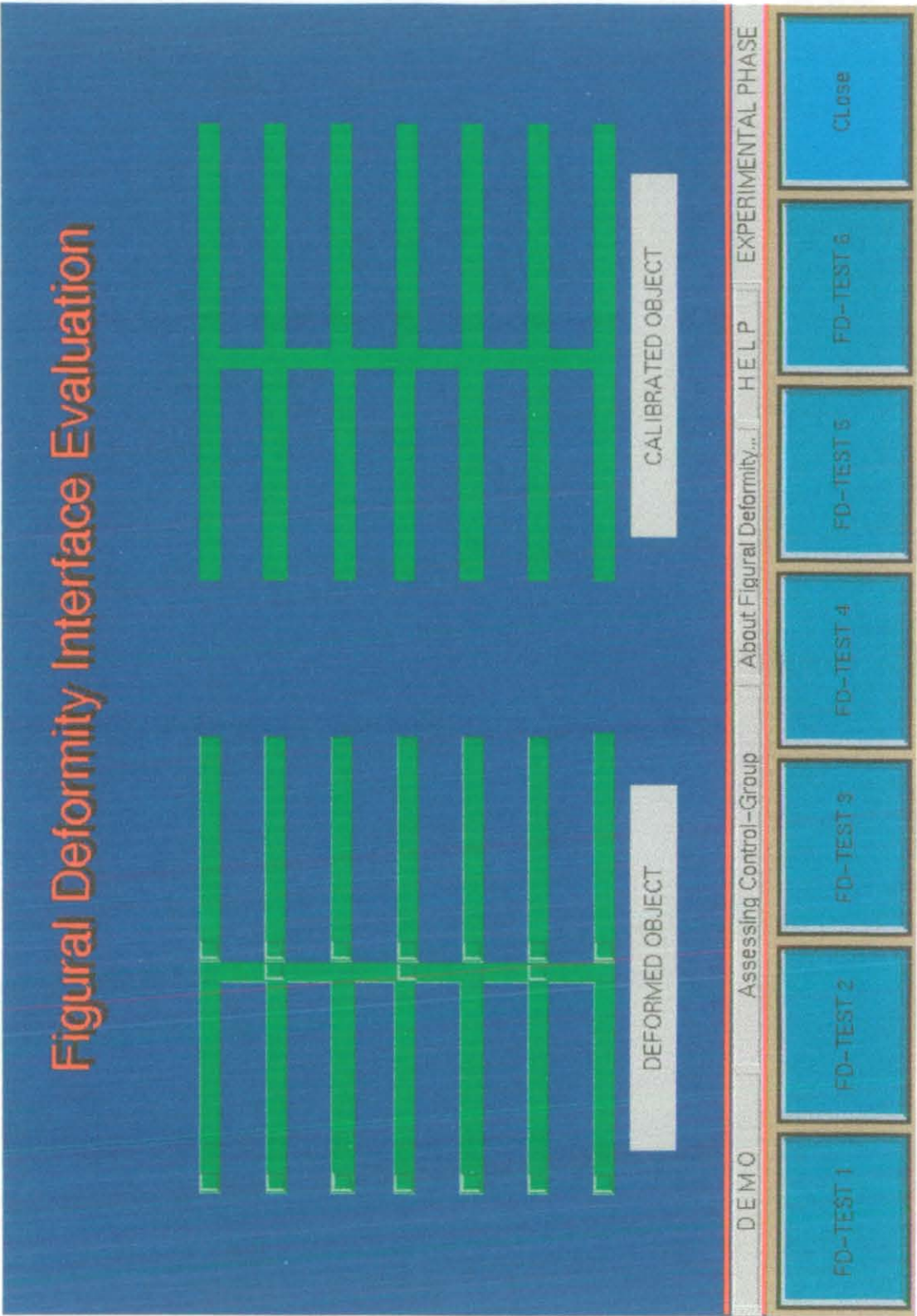


Figure 36:Experimental Group (FD-Test 6)

- The “uicontrol” command has been used to draw user interface controls and lines of various shapes (including FD-objects) and colours. The following part of the code demonstrates the creation of the “Close” 3D- pushbutton in the main console:

Code from M-file “FD PLATFORM.m”

```
% ##### PUSHBUTTON No. 1 #####
% *Shadow: 1*
sh1 = uicontrol('Style','Text','Position',...
    [780,5,115,83],'BackgroundColor',[0.002 .50 .50]);

% *Shadow: 2*
sh2 = uicontrol('Style','Text','Position',...
    [777,89,115,4],'BackgroundColor',[.60 .81 .81]);

% *Shadow: 3*
sh3 = uicontrol('Style','Text','Position',...
    [777,7,4,85],'BackgroundColor',[.60 .71 .81]);

% *Shadow for Left corner*
sh4 = uicontrol('Style','Text','Position',...
    [777,4,117,3],'BackgroundColor',[0 0 0]);
sh5 = uicontrol('Style','Text','Position',...
    [777,92,117,3],'BackgroundColor',[0 0 0]);
sh6 = uicontrol('Style','Text','Position',...
    [774,4,3,91],'BackgroundColor',[0 0 0]);
sh7 = uicontrol('Style','Text','Position',...
    [893,4,3,91],'BackgroundColor',[0 0 0]);
sh8 = uicontrol('Style','Text','Position',...
    [779,7,1.6, 2],'BackgroundColor',[.02 .50 .50]);
sh9 = uicontrol('Style','Text','Position',...
    [890,88,2 , 2],'BackgroundColor',[.02 .50 .50]);

% **Actual PushButton: 1**
But1 = uicontrol('Style','Pushbutton','Position',...
    [780,10,110,80],'Callback','close(gcf)',...
    'String','CLose','BackgroundColor',[0.002 0.7020 0.8020]);
```

- The Bar graphs have been declared as smaller separate windows under the main window and their handles are *children* of the main window. The following code demonstrates how to create and access a specific bar graph:

Code from M-file “fd_vs_bar.m”

```
Bar1 = axes('Position',[0.04 .24 .16 .26]);

set(Bar1,'title',...
    text(0,0,'Lboro-Lancaster', 'FontSize', [11], 'FontWeight','Bold'));
```

Separate Window



Code from "startnow.m"

```
subplot(Bar1);

set (Bar1,'YLim',[0,20000],'LineWidth',[3],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',{'Min';...
                                     'Max';...
                                     'Av.';...
                                     '5% '},...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 5000, 9000 14000, 17000, 19000]);

M = [1,2,3,4];
[Mb,Nb] = bar(M,N1(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.1 -----
Temp_Bar1 = Bar1;
Temp_Bar1 = plot(Mb,Nb);
set(Temp_Bar1,'Color','green','LineWidth',[8]);
```

Notice that controlling the line width of the bar-graph cannot be done directly. The only possible way is to create an auxiliary matrix to store the bar graph data and then to use the command *plot* which is equipped with the property *LineWidth* to perform the required task.

- The program is equipped with its own graphical functions for the drawing of the FD-objects. A characteristic sample of code is presented below:

Code from "fdDraw.m"

```
function v = fdDraw(a,b,c,d,A,B,C,D,T1,T2,T3,T4,E,F,G,x)

% --*-- Calibrated Size at 50% of window-area for Threshold values --*--
% -----
% -- Designing the "E" object ---
% -----

%      -- Designing the left side of "E" object --

v(1) = uicontrol ('Style','text','Position',...
    [a,...
    b,...
    x,...
    ((C * d/2)/T3)],...
    'backgroundcolor',[E,F,G]);
```


% -- Designing the bottom bar of the "E" object --

```
v(2) = uicontrol ('Style','text','Position',...
    [(a+x),...
    b,...
    ((B * c/2)/T2),...
    x],...
    'backgroundcolor',[E,F,G]);
```

% -- Designing the middle bar of the "E" object --

```
v(3) = uicontrol ('Style','text','Position',...
    [(a+x),...
    (b+(((C*d/2)/T3)/2) - (x/2) ,...
    ((D * c/2)/T4),...
    x],...
    'backgroundcolor',[E,F,G]);
```

% -- Designing the upper bar of the "E" object --

```
v(4) = uicontrol ('Style','text','Position',...
    [(a+x),...
    (b+ ((C * d/2)/T3 )-x),...
    ((A * c/2)/T1),...
    x],...
    'backgroundcolor',[E,F,G]);
```

Fig.37 presents some information on the logic behind the above code segment:

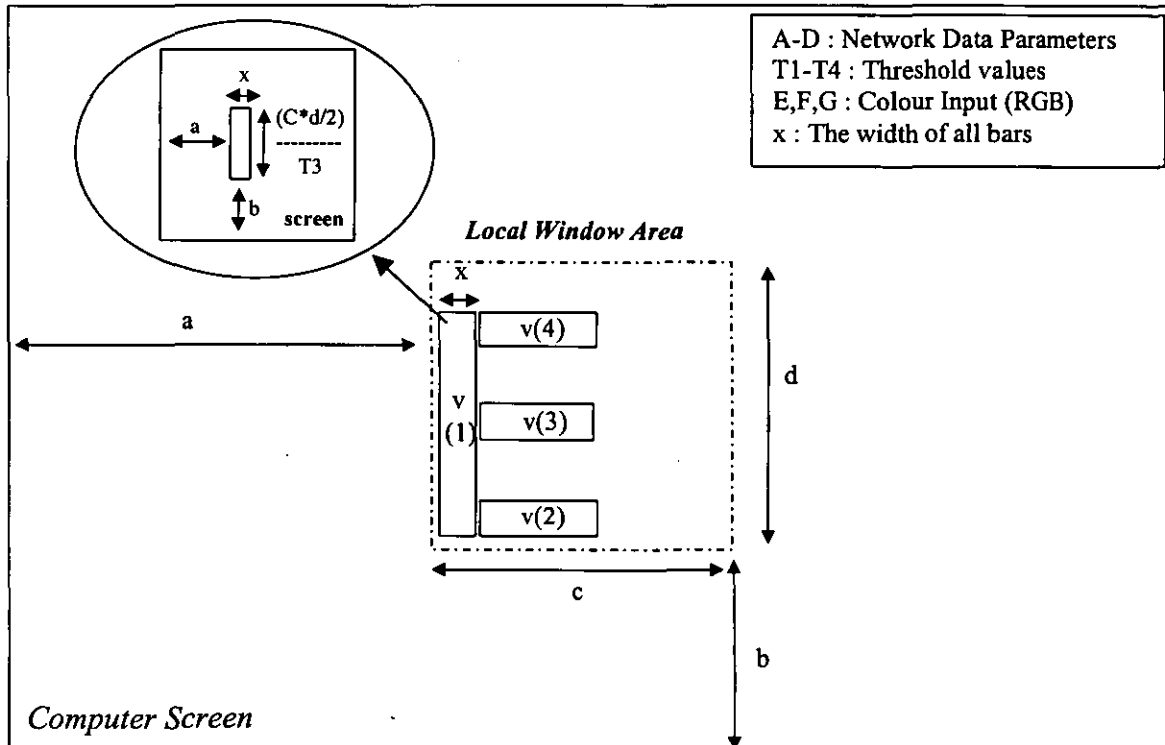


Figure 37: Drawing an FD-Object

- Throughout the program, all the required timing has been achieved by using identical timing loops. This suggests that all the gathered time measurements, ideally, are extremely accurate. In practice, even if some delay is incorporated, it is ensured that this does not change the final outcome since according to Bausell [Ba86], a constant can be added to either (or both) set(s) of scores without affecting the Pearson r correlation coefficient.

Section 5.5. Conclusion

Having implemented the idea of FDV in the current prototype FD-interface, the next step, which certainly was the most interesting, included user-trials with relevant users. The collected data will be statistically processed in the following chapter, according to the directives of chapter 4.

CHAPTER

6

Experimental Data Analysis

Section 6.1. Introduction

The prototype FD-Interface, described earlier in chapter 5, played a pivotal role in a series of user trials in order to give users hands-on experience, and therefore gaining valuable feedback concerning the effectiveness of Figural Deformity Visualisation. Each individual user testing session consisted of one of two possible session types (a separate type of test for each group of users), with a total of 50 network-management and communications related people. Following acceptable practice [Elli96], each session was divided into the following four sections: a) a short informative presentation, b) user testing with representative tasks, c) filling out questionnaires identical to the proposed one in chapter 4 and d) a small discussion with each user.

Each of the following sections of this chapter is dedicated to a specific domain of the statistical data set which was generated during the study. The relevant analysis is intended to offer a panoramic view of the gathered information. Where possible, the actual mathematical findings have been explained from a human factors point of view. Many sections are enriched with self-explanatory graphs.

Section 6.2. Variable –Z- analysis (Response to alarm situations)

The treatment of this particular test, depending on the interface type, involved the following:

- Monitoring user responses to alarm situations by visualising data without Figural Deformity Objects (control group)
- Monitoring user responses to alarm situations by visualising data with Figural Deformity Objects (experimental group)

Table-1 presents the raw data before statistical analysis.

CONTROL GROUP			EXPERIMENTAL GROUP*		
70.71	53.14	97.5	27.39	6.483	100
113.2	100.4	62.5	43.34	28.11	85
109.2	96.4	65	32.81	15.49	100
119.2	100.6	100	36.99	3.114	97.5
205	48.13	90	23.94	1.111	100
62.55	21.17	97.5	43.09	22.34	100
90.95	53.99	95	23.34	4.76	97.5
128.2	109.4	100	19.57	2.24	100
160.8	182.9	72.5	28.8	7.083	100
92.81	63.04	92.5	29.69	4.933	100
99.88	58.97	95	35.43	7.051	92.5
53.57	17.98	42.5	30.12	15.56	100
88.87	39.66	85	30.6	5.807	92.5
79.67	30.95	97.5	23.74	9.249	90
86.71	58.71	62.5	39.28	14.06	97.5
93.58	54.17	90	20.22	1.939	95
61.77	15.36	65	23.35	2.762	100
91.75	59.78	90	31.29	8.21	92.5
76.84	60.56	70	35.43	7.05	100
80.29	45.64	50	20.17	5.078	100
90.92	33.55	95	23.9	7.68	100
98.42	46.64	100	24.84	8.862	100
79.94	26.82	95	17.12	1.61	100
86.63	21.45	85	17.39	2.181	100
97.32	21.66	55	15.28	8.844	95

Av. Time (Sec) Av. Std. Dev (Sec) Av. Success (%) Av. Time (Sec) Av. Std. Dev (Sec) Av. Success (%)

Table 1: Data relevant to variable –Z-

*FD-Object Calibration approximately at
64% of Bar-Charts' height

As the reader may suspect after taking a quick look at fig.38, the findings show strong evidence in favour of the FD-Interfaces. The average response for the control group (Bar-graphs) is equal to 96.8 seconds against only 27.9 seconds for the experimental group (FD-Interfaces).

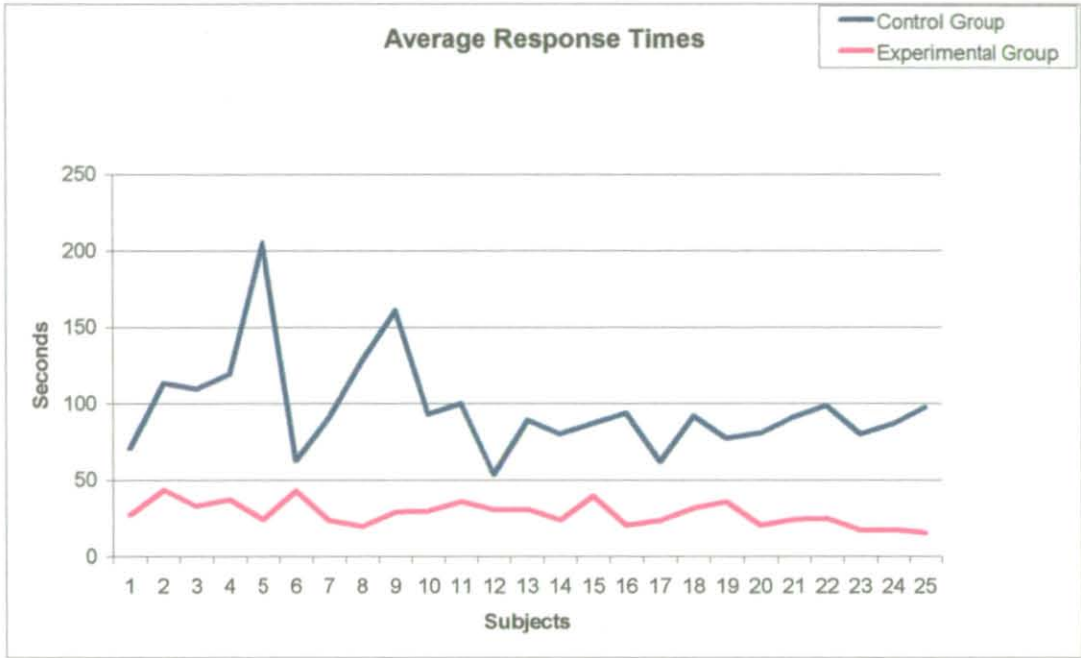


Figure 38: Graph of Average Response Times

The practical exegesis of these numbers leads to a t-statistic result of **10.5**, undoubtedly higher than the value of 1.7 as found in most published distributions of t-statistics at 24 degrees of freedom with the basis of the comparison being established at the 0.05 level of significance [Wal93,Moo93]. Therefore, the first null hypothesis (see section 4.5) can be **rejected** due to existence of statistically significant evidence.

Continuing the investigation, as fig.39 suggests, the average percentage of successful identification of un-healthy routes for the two groups are 82 and 97.4%. The t-statistic based on the particular data is equal to **4.228**, and therefore the rejection of the second null hypothesis is statistically justified.

6.2.1 Discussion

Besides the t-statistic results that enabled the official rejection of the two hypotheses, the reader might agree that FD-Interfaces not only enable users to perform their tasks

in a faster and more effective way, but also with greater ease. Every subject in the experimental-group commented that the required tasks appeared extremely easy and straightforward and that the experiment seemed to them rather like a “game”! The initial discomfort of meeting a new “challenge” was followed by vast improvement and positive remarks. With respect to the control-group subjects, the opposite scenario was nearly always predominant; people were eager to start without further explanation, however, the initial intimacy with bar-graphs was insufficient to yield competitive timings.

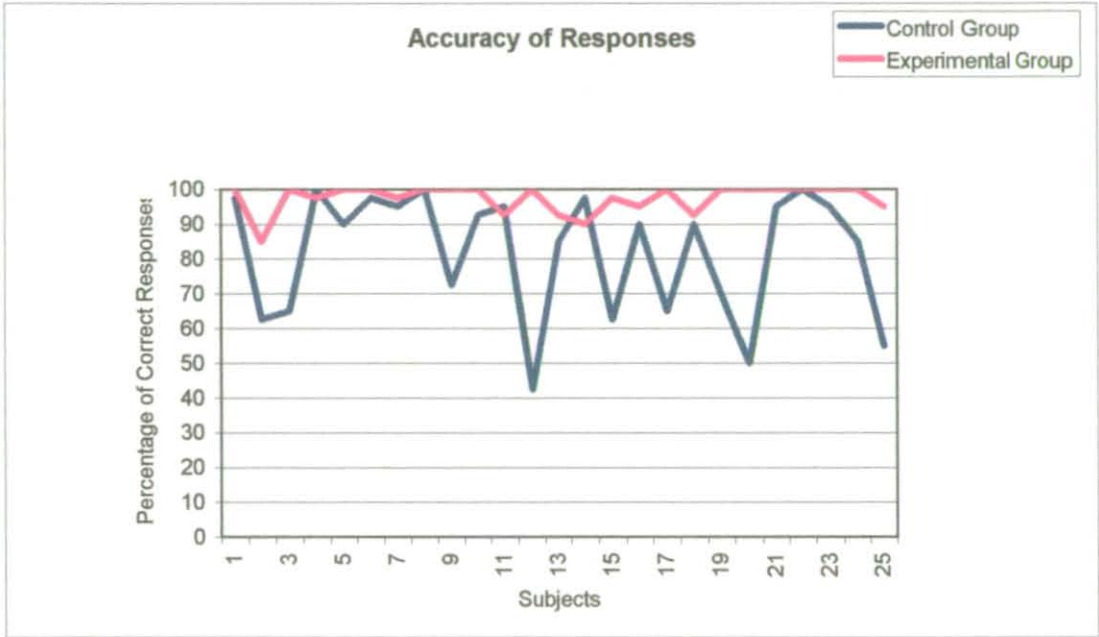


Figure 39: Graph of accuracy of responses

The fact that the standard deviation of the control and experimental group average responses are 31.82 and 7.95 respectively, consists evidence to the previous claim. After all, the standard deviation is the most often used measure of spread [Wal93] and is a measure of sample variability which can denote in this instance the level of easiness. As a result, FDV seems to provide a more uniform profile. Fig.40 presents graphically the two sets of standard deviations calculated individually from the four answers of each particular subject after taking the relevant tests (see chapter 5). In a way, the graph allows a microscopic view of the response-variation. The average standard deviation of the set of the collected standard deviations for the control group is 56.84 seconds against only 8.06 seconds with respect to the experimental group. From a human factors perspective, it is not surprising that the responses of the control

group suffer such great variation. The difference of the short-term memory characteristics of the various users appears to play a significant role in their performance when working with bar-graphs.

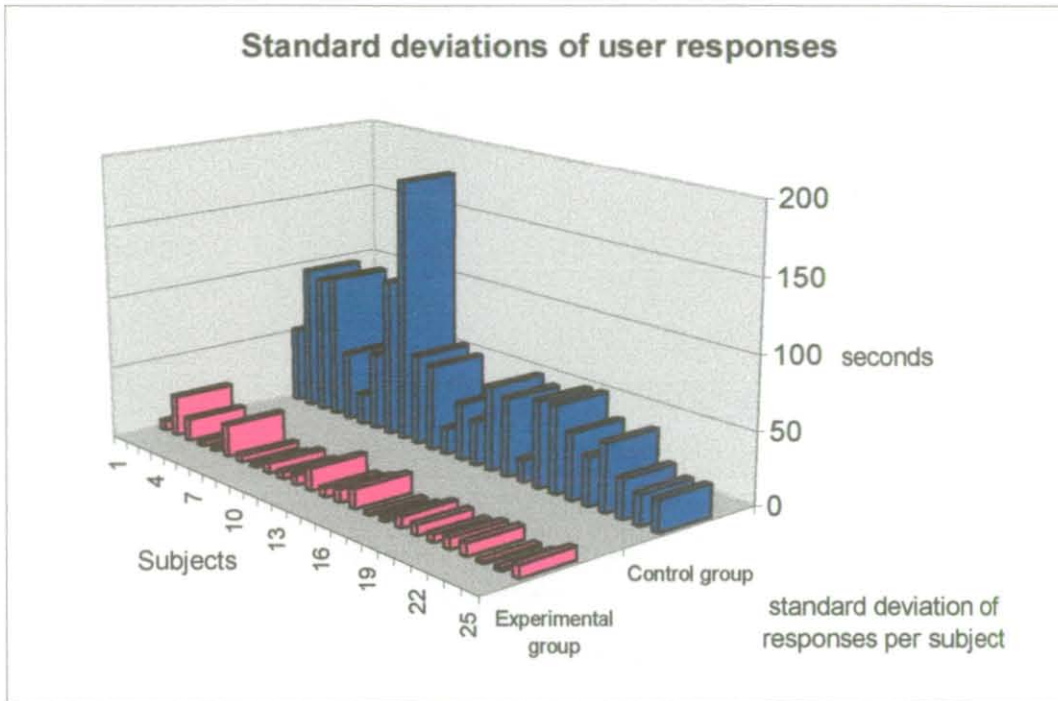


Figure 40: A more in-depth look of the variation of user responses

Section 6.3. Variable –A- analysis (Object Shape)

The treatment of this test, depending on the FD-Object shape, involved the following:

- Monitoring user responses to FD-Interfaces by utilising the generic “E” shape to visualise four network parameters per network route
- Monitoring user responses to FD-Interfaces by utilising a different shape, such as a simple square to accommodate the relevant four parameters per object

Most of the other variables were kept constant during this test (object calibration = 50% of the local window area, number of objects per screen = 20, object colouring = green (blue background), d.o.f. per object = 4 in both cases). Although it is impossible to use identical parameter mapping on two different shapes, variation was kept to a

minimum. The null hypothesis to be tested in this scenario was formulated as following:

“Different FD-Object shapes do not significantly alter user performance”

Table-2 presents the relevant collected information.

<i>Response time to ‘E’ (Sec)</i>	<i>Response time to ‘Square’ (Sec)</i>	<i>Subjective answer</i>
3.9	3.99	2
4.173	4.024	4
4.974	3.971	2
4.294	3.9	4
3.989	3.176	2
7.124	1.683	5
3.159	4.149	4
6.128	1.983	5
7.093	6.01	3
7.02	4.68	5
4.071	3.662	5
5.075	3.751	3
17.17	5.58	2
10.14	9	4
25.27	9.061	5
5.219	2.335	4
11.15	9.884	5
6.086	3.9	2
20.68	9.06	5
22	11.05	5
17.06	11.035	5
19.432	9.045	2
6.084	4.995	5
4.067	1.273	4
4	2.1	5

Table 2: Data relevant to variable –A-

Applying the t-test formula to both sets of timings yields a t-statistic result of **2.586**, therefore the null hypothesis can be rejected.

6.3.1 Discussion

Proving that different object shapes can significantly effect user promptness is in accordance with the fundamental principles of Figural Deformity Visualisation. The theoretical justification lies on the manner that a particular object-shape embraces the Gestalt laws (see section 3.5). With respect to the particular test, a square shape is

more 'symmetric' (the term is used within a gestalt law context) than a "E" shape and therefore, on average, is detected faster when deformed ($\bar{T}_{\text{square}} = 5.33 \text{ s}$, $\bar{T}_{\text{'E'}} = 9.17 \text{ s}$). The fact that the Pearson coefficient detected quite a high correlation between the two sets of timing data ($r = 0.818$) indicates that user responses share rather a similar profile in both object shapes.

Moving onto the subjective answers, it seems that seventeen out of twenty five subjects replied that either they disagree or strongly disagree with the statement "*Overall, I felt that the object-shape does not alter the effectiveness of FD-interfaces*". Nevertheless, six subjects strongly agreed with the above statement while two were undecided. Concluding therefore, it seems that 68% of the participants perceived correctly that object-shape is a rather important constituent of FDV that needs to be given proper consideration when wishing to maximise user performance.

Section 6.4. Variable –B- analysis (Object Colouring)

The work with regard to FD-Object colouring was to monitor user responses while using the following four colours:

- Red
- Green
- White
- Yellow

All other variables were kept constant during this test. The null hypothesis to be tested in this section was stated as:

"Different FD-Object colouring does not alter significantly user performance"

Table-3 & fig.41 presents the relevant collected information. The various column headings are explained below:

Tr, Tg, Tw and Ty: response time recorded for each treatment

Std.Dev.: the calculated standard deviation of the four different readings /user

Sr, Sg, Sw and Sy: % score for each treatment

Likert-Score: The corresponding subjective answer.

The average response time (seconds) for the four colours are:

23.9556 (red objects)

19.6124 (green objects)
19.1776 (white objects)
18.9278 (yellow objects)

The success scores (%) are:

98.4 (red objects)
99.6 (green objects)
97.2 (white objects)
97.2 (yellow objects)

Applying initially the t-test formula between the two sets of response times for red and yellow colouring will yield a t-statistic result of 3.12382 (note that the critical value = 1.7) therefore, the previous null hypothesis can be safely rejected.

Tr	Tg	Tw	Ty	Std. Dev.	Sr	Sg	Sw	Sy	Likert-Score
20.24	18.9	18.6	15.9	1.8189	100	100	100	100	2
20.68	19.45	18.68	15.83	2.059	100	100	100	100	3
32	14.58	18.41	15.51	8.08	100	100	100	100	2
26.2	22.42	26.63	24	1.967	100	100	90	100	2
17.25	14.96	14.01	10.24	2.92	90	100	100	60	4
27	17.6	18.9	25.39	4.66	100	100	100	100	5
23.2	17.18	13.89	4.095	7.99	100	100	90	100	5
24	18.1	18.9	20.02	2.617	100	100	100	100	4
17.68	22.11	25.98	22.75	3.415	100	100	100	100	4
25.6	18.2	18.9	23	3.497	100	100	100	100	5
41.57	28.78	23.04	22.7	8.818	100	100	90	100	5
18.28	18.96	15.99	14.39	2.104	100	100	100	100	5
23.76	20.62	20.82	21.85	1.438	90	100	90	90	1
33.49	20.09	20.31	22.1	6.394	100	100	100	100	5
33.59	28.23	21.35	25.48	5.133	100	100	90	90	5
29.65	19.75	15.94	20.58	5.807	100	100	100	100	4
16.98	17.07	19.96	16.63	1.543	100	100	90	100	5
20	22.08	19.03	18.9	1.469	90	90	100	100	3
20.2	22.42	25.8	24.2	1.7	100	100	100	100	5
23.41	17.41	17.94	15.44	3.414	100	100	100	100	5
23.11	22	26.2	16.1	4.22	100	100	100	100	5
20.32	19.2	17.9	17.8	1.194	100	100	100	100	5
25.83	19.05	13.77	15.67	5.3	100	100	90	90	5
15.99	14.09	12.59	24.04	5.102	90	100	100	100	2
18.86	17.06	15.9	20.58	2.053	100	100	100	100	4

Table 3: Data relevant to variable –B-

Although it has just been proven that there is significant difference between Tr and Ty timings, it might be interesting to find out whether Tw and Tg sets of values can generate significant t-statistics when compared to Tr values. In fact, $t(Tr,Tw) = 3.224$

and $t(Tr,Tg) = 3.021$, and as a result, all four colours played a statistically significant role on user promptness.

Moving the investigation to the actual set of scores concerning each colour, it can be shown that there is no significant difference between the Sw and Sy values ($t(Sw,Sy) = 0$), Sg and Sr ($t(Sg,Sr) = 1.41$) or even Sg and Sy ($t(Sg,Sy) = 1.385$).

However, the previous statement is not true when comparing Sg and Sw ($t(Sg,Sw) = 2.4$). Therefore object colouring seems to effect the integrity of user responses, in a noticeable but not highly significant manner.

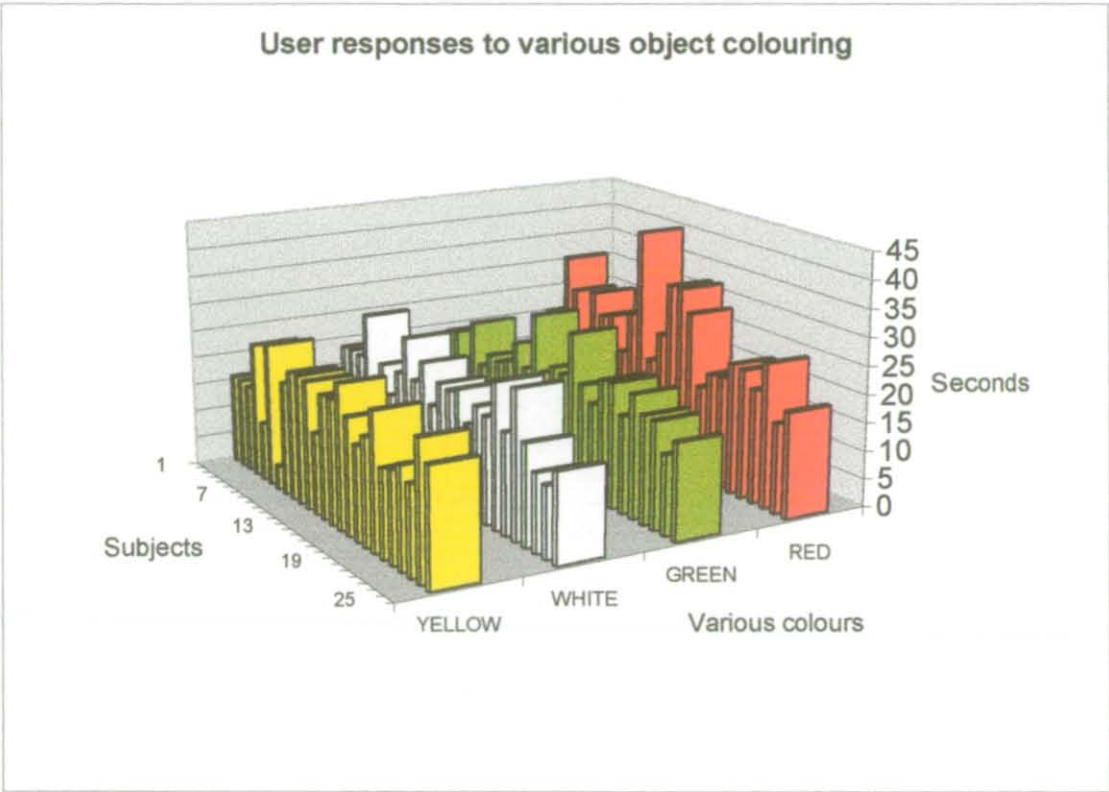


Figure 41: Graph of Tr, Tg, Tw and Ty values

The correlation analysis between Tr,Tg,Tw and Ty indicated a relatively strong similarity in the timing profiles of Tg and Tw ($r_{Tg,Tw}=0.652$). However, any relationship between the actual response times and their corresponding scores with respect to each colour is very weak. In all four cases, the correlation coefficients were below 0.3.

6.4.1. Discussion

One thing that should be clear by now is that the change of object colouring provided a statistically significant difference in user promptness and integrity of responses. Therefore, further analysis is needed in order to fully understand the previous findings. By now, the reader will see that the red colour seemed to have added the biggest delay in user response times and yet the corresponding average scores were not the worst ones. Instead, white and yellow objects yielded even lower percentages. The question therefore is twofold:

- Why did users take the most time to identify the unhealthy routes from the specific collection of red-objects (assuming all other factors were constant)?
- Why did users take the least time to identify the unhealthy routes, from the collection of white (or yellow) objects, and yet with the least success (assuming they were not in a hurry to abandon the experiment)?

The answer lies within the human colour perception, a highly subjective [HUSAT89] and complicated process.

6.4.1.1. Colour Perception

Since the time Newton first split white light into its components [Wil66] a lot of work has been done to attempt to fully understand how colours are perceived and how can they be best described. In general, colour is visible light which corresponds to a very small part of the electromagnetic spectrum. The main components of the colour spectrum and their corresponding wavelengths (nm) are [Lon84, URL2]:

- Violet (380-420)
- Blue (420-495)
- Green (495-566)
- Yellow (566-589)
- Orange (589-627)
- Red (627-780)

Colour vision is possible due to the existence of two types of eye-receptors, called “cones” and “rods”. Cone receptors function in high levels of illumination, and they form the so called ‘photopic’ visual system. Rod receptors function on low levels of illumination and they form the ‘scotopic’ visual system. While the photopic system is

more sensitive to colours towards the red-end of the colour spectrum, the scotopic system is more sensitive to movement, and colour vision is impaired with increased sensitivity only to colours towards the blue end of the visual spectrum. Therefore, under normal daylight conditions the photopic visual system operates and the properly adapted human eye is found to be most sensitive to yellow and green spectral wavelengths while least sensitive to blue and red [Lon84].

Bearing this in mind, it can be seen why blue and red should be avoided when presenting text on multicolour displays. In certain cases, red often appears to be closer than blue when combined together and the human eye can sometimes experience a 'beating' effect [Lon84] whilst trying to focus on both colours simultaneously. However, the use of blue is ideal for display backgrounds due to its possible tendency to recede [Wil66] and as such has been used in this experiment.

The previous paragraph provides a good explanation to the first part of the question, with regard to section 6.4.1. In other words, response times can be, indeed, maximised when having red objects on a blue background, however red objects might appear closer, therefore detection of deformities does not necessarily have to be affected in a negative manner. Indeed, that is why S_r scores are not the worst.

Concerning the second part of the question, it might be true that users answered quicker simply because they thought that the use of white or yellow objects eased their task. In reality, most of them possibly experienced the 'opponent process' mechanism [Lon84] which can be triggered by certain combinations of colours, such as yellow and blue that can cause an opposite and incompatible activity in the visual system, due to the eye's failure to detect mixtures of these colours in the same segment of light. The result of this can be 'shadowing' and colour reversal after-images. In general, combinations of colours of widely differing wavelengths (such as the white colour which embraces the whole visible spectrum) can cause what is known as 'chromatic aberration', that is, unpleasant visual sensations.

To conclude this colour analysis, it seems that the best colour combination that provided the fastest response with best integrity of results in this test, was green on a blue background. This is not surprising, since it seems to be indisputable within the

human-factors society that green is a very sharp colour [HUSAT89] to which the eye is very sensitive [Lon84]. However, one should know that observers can vary in their response to colour differences and their subjective influence can be very strong [Wil66].

6.4.1.2. Subjective answers

Passing onto the subjective answers, it seems that eighteen out of twenty five subjects replied that either they disagree or strongly disagree with the statement *“Altogether, the different colouring of the included objects did not alter my visual perception”*. Only a single subject strongly agreed with the above statement while four simply agreed and two were undecided. Concluding this section, it seems that 72% of the participants perceived correctly that object-colouring is an important constituent of FDV that needs to be properly considered when wishing to fine-tune user performance.

Section 6.5. Variable –C- analysis (Calibration-method)

The treatment in this section was to monitor user responses whilst using different object calibration techniques. The FD-Objects were calibrated at approximately 2cm high, occupying only 37% of the respective bar-charts height. All the other relevant variables were kept constant during this test. The null hypothesis to be tested in this section was stated as:

“Different object calibration does not alter significantly user performance”

Table-4 presents the relevant gathered information. Note that despite the excessive object minimisation, the overall average recorded timing for this treatment was 29.21 sec while the overall average standard deviation was less than 9.7 sec. The average success was 96.48%. Comparing this set of findings to the previously obtained ones (Table-1) in which the dedicated screen space per object was much higher, one can conclude that there is not statistically significant difference to reject the hypothesis (the relevant t-test values are 0.54 for time responses and 0.685 for % of successes). Furthermore, the relationship between the two sets is quite strong ($r=0.883$), yielding almost matching profiles.

Av. Time (sec)	Av. Std. D(sec)	Av.Success (%)	Sub. Answer
28.4	5.26	100	1
50	30.05	90	1
36.08	15	100	1
38.92	40	95	1
24	1.26	100	2
43.27	20.18	97.5	1
22.18	5.36	97.5	1
18.9	1.96	100	1
28.08	6.9	100	1
32	7	95	1
47	9.99	97	2
35.99	12.39	97.5	1
26.19	7.92	95	3
23.08	10	95	1
30.92	11.76	100	1
24.67	3.06	95	1
15.06	2.0611	75	1
34.91	7.97	90	1
36.01	8.16	97.5	1
19.75	6.59	100	1
29.09	5.56	97.5	1
26.88	7.895	100	1
20.01	1.11	100	1
22.95	3.087	97.5	1
16	10	100	1

Table 4: Data relevant to variable –C–

6.5.1. Discussion

The findings of this test prove in the best possible way the theoretical claims concerning the minimal short-term memory overheads that FDV requires as opposed to more quantitative visualisations. The ability of FD-objects to be literally squeezed within 37% of the respective bar-chart height (the object width was even more compressed) while still visualising the same bulk of information in a meaningful way, proves the superiority of the FD-mechanism when it comes to screen-economy.

Concerning the usefulness of the calibrated object, twenty two out of twenty five subjects strongly agreed with the statement *“I believe that the existence of a calibrated object eased visual detection of deformities”* while two simply agreed and only a single person was undecided. Clearly, most users have appreciated the presence of a calibrated object within the FD-Interface.

Section 6.6. Variable –D- analysis (Parameter-mapping)

The purpose of this section is to investigate user responses with regard to object deformation and ease of detection. The null hypothesis to be tested was formulated as:

“Vertical as opposed to horizontal object deformation does not alter user promptness”

Table-5 presents the collected timings and the relevant subjective answers.

T_{vertical} (sec)	$T_{\text{horizontal}}$ (sec)	Subjective Answer
3.091	2.274	2
3.001	1.792	4
3.101	2.729	3
2.055	1.937	5
3.589	3.346	3
2.446	2.255	5
3.893	2.466	5
2.09	1.785	5
2.127	1.912	4
2.962	2.88	5
4.278	2.599	5
4.072	2.56	3
2.092	2.096	2
2.371	1.64	1
2.624	2.161	2
3.519	2.467	4
3.756	2.158	5
3.605	2.82	4
2.06	1.94	5
2.581	1.434	5
1.914	1.783	5
1.61	1.998	4
1.994	1.739	5
2.539	2.186	5
3.288	4.14	3

Table 5: Data relevant to variable –D-

The average timings for vertical and horizontal detection of deformation are 2.826 and 2.283 (sec) respectively. Applying the t-test formula to T_{vertical} and $T_{\text{horizontal}}$ sets of values yields a value of 2.809. Therefore the null hypothesis can be comfortably rejected. As has been said previously (see section 5.3.2), this particular test was very demanding in terms of visual capabilities, therefore it was decided to investigate any

partial correlation among users age and their relevant time responses. The findings are the following:

- $r_{T\text{-vertical}, T\text{-horizontal}} = 0.583$ (between weak and strong, towards weak)
- $r_{T\text{-vertical}, \text{Age}} = 0.2364$ (between very weak and no correlation, towards very weak)
- $r_{T\text{-horizontal}, \text{Age}} = 0.5472$ (between weak and strong, towards weak)
- $r_{(T\text{-vertical } T\text{-horizontal}), \text{Age}} = 0.5578$ (between weak and strong, towards weak)

The previous labelling has been implemented based on the following onomatology (after consulting Moore's [Moo93] model example):

- (1) Perfect Correlation: $r=1$
- (2) Very Strong: $r=0.90$
- (3) Strong: $r=0.75$
- (4) Weak: $r=0.50$
- (5) Very Weak: $r=0.25$
- (6) No correlation: $r=0$

6.6.1 Discussion

Based on the previous statistical analysis concerning an "E" shape, we can conclude that users seem to detect any horizontal object deformation faster than vertical deformation. Therefore, important parameters should preferably be mapped on the horizontal sides of an FD-Object. Another important outcome seems to be that $T_{\text{horizontal}}$ values have a higher correlation with the age of subjects. The value of the partial correlation coefficient for the three sets of values (T_{vertical} , $T_{\text{horizontal}}$, Age) does not seem to decrease significantly when age is partialled out.

Regarding the relevant subjective answers, seventeen out of twenty five people replied that either they disagree or strongly disagree with the statement "*Vertical deformation of objects as opposed to horizontal one is perceived more easily*". Nevertheless, four subjects were undecided, three seemed to simply agree while a single subject strongly agreed. It seems that 68% of the participants perceived correctly that they managed to respond faster to any horizontal object deformation. Fig.42 summarises the findings of the subjective questionnaire with regard to the present and previous sections.



Figure 42: Summary of the subjective answers

Section 6.7. Variable –E- analysis (Number of Objects per screen)

The purpose of this section is to investigate user responses when the size of the object collection is doubled. The null hypothesis to be tested was formulated as:

“User promptness does not change as the number of included objects per screen increases ”

Table-6 presents the average timing, standard deviation and success per subject. Applying the t-test formula to the corresponding values of Tables 6 & 4 will give a result of 2.930 (timings) and 1.580 (integrity of responses). Therefore the null hypothesis can be rejected, keeping in mind that the relevant integrity of responses does not significantly change. Remember that the size of the objects used in obtaining the results of Table-4 was a lot smaller; this is done to avoid the potential criticism of obtaining statistical significance due to the smaller size of the current objects in use therefore causing decreased visibility.

Average Time (sec)	Av. Std. D (sec)	Av. Success (%)
29.79	5.443	100
26.98	1.402	95
40.981	10.1	95
25.02	1.603	95
37.05	9.809	87.5
31.23	6.241	92.5
41.19	8.89	95
33.53	4.891	97.5
28.74	3.169	92.5
31	6.421	92.5
59.9	7.29	92.5
46.09	2.56	95
35.69	2.674	77.5
37.38	4.098	100
47.12	4.998	90
37.42	4.996	95
30.72	2.221	97.5
40.9	10.46	92.5
38.39	5.98	100
33.91	5.895	100
28.51	3.179	90
30.5	4.179	100
38.7	3.399	85
38	8.852	97.5
36.69	7.933	97.5

Table 6: Data relevant to variable –E–

6.7.1. Discussion

When object size was considerably reduced due to different calibration, it was not possible to obtain significantly different user profiles (see section 6.5). However when the size of the collection doubled (even though object size did not change dramatically) it was possible to alter user readiness despite the fact that the validity of responses did not change. Therefore user promptness seems to be more vulnerable to screen loading than the integrity of the actual answers.

Section 6.8. Variable –F– analysis (D.O.F. per object)

Coming to the last part of the experiment, the purpose here is to investigate user responses when the number of degrees of freedom per object increases to eight and fifteen. The null hypothesis to be tested was formulated as:

“User responses do not change as the number of degrees of freedom per object change ”

The rate of deformation in both treatments was equal and very marginal to human visual acuity (almost approaching one minute of arc at a viewing distance of 500-700 mm, being regarded as the approximate limit of visual acuity in good lighting conditions [Wil66]) in order to maximise the test difficulty. Table-7 presents the collected timings for both treatments. The average responses are 27.66 and 17.98 seconds for $T_{dof(8)}$ and $T_{dof(15)}$ respectively. The t-test for these two sets of values yields a result of 1.542 therefore, the null hypothesis cannot be rejected.

$T_{dof(8)}$ (sec)	$T_{dof(15)}$ (sec)
9.55	6.65
23.18	4.251
10.19	3.951
80.73	34.87
16.25	19.42
78.07	56.37
4.121	4.282
37.75	5.594
16.24	7.301
6.27	14.08
5.382	58.72
8.239	8.39
106.1	21.6
10.04	15.93
40.48	41.14
19.83	7.71
13.69	5.52
43.44	8.519
11.04	16.9
52.34	16.78
36.94	18.34
34.94	10.34
3.649	4.53
11.91	39.31
11.2	19.22

Table 7: Data relevant to variable –F-

6.8.1. Discussion

Since it is not safe to fully accept the null hypothesis, one could safely assume that fifteen degrees of freedom per object might be a wise limit unless further experiments take place to explore the full potential of the mechanism.

Section 6.9. Conclusion

The primary aim of this chapter was to investigate the findings of the proposed experiment and to conclude on the usefulness of Figural Deformity Visualisation. As the relevant statistical analysis indicates, there are considerable benefits in store for human operators by employing FD-Interfaces to visualising network parameters and consequently monitor multiple variations. Based on the current findings, a brief summary of recommendations on how to maximise the potential of the relevant visual mechanism follows:

- The best chromatic combination to maximise user-promptness and integrity of responses is green object colouring within a blue display background.
- Symmetric shapes assist users to detect any object deformations faster.
- Important parameters should be accommodated on the horizontal parts of each object.
- The number of objects per screen has some impact on user performance which is not negligible.
- The maximum number of degrees of freedom per visual object is recommended to be fifteen.

 CHAPTER

7

Prototype FD-Interface for SMDS Alarm Station

Section 7.1. Introduction

To recap, one of the three research aims formulated in section 1.3 was to demonstrate the merits of the particular novel visualisation mechanism with a practical verification of the theoretical claims. Assuming that the theoretical claims have been verified, this chapter presents the very first implementation of a practical Figural Deformity Interface which has been evaluated in a major company of the British communications industry, British Telecommunications plc (BT).

In particular, this chapter describes a supplementary FD-Interface which has been recently designed for an SMDS Alarm Station Prototype Version 2.1 developed for BT. Its purpose is to acclimatise users with the idea of F.D.V. and enable them to assess the usability and usefulness of this particular approach in an industrial environment. The chapter concludes with an informal usability test conducted with users of the Alarm Station at Walsall Network Operations Unit.

Section 7.2. The SMDS Alarm Station

SMDS is a commercial service which enables users (i.e. banks etc.) to exchange large amounts of data on a non-constant or so called “bursty” basis. SMDS (Switched

MultiMegabit Data Service) and ATM (Asynchronous Transfer Mode) are the two main parts of SuperJanet which provides Wide Area Connectivity among UK academic institutions and furthermore a gateway to the wider Internet world [Phi96]. The SMDS network carries the major service of IP interconnect and is being continuously monitored by the SMDS monitoring system set up at Walsall NOU. The original SMDS monitoring system consists of a Monitor station running on a PC which is controlled by a Control Station running remotely on a SUN workstation. The monitor station is responsible for generating test-packets which are sent across the SMDS network and back to the monitor station via Megastream links connecting Walsall NOU to a number of nodes in the SMDS network [Spe98].

The monitoring system was originally designed as a pure research tool providing non real time network performance data collection. The gathered results were logged over a period of several months [Spe98] before being further processed to provide historical information on the performance of the network. However a more immediate analysis was requested by operations staff at Walsall and as a result an alarm station was designed to fulfil this role. The alarm station is engineered to receive and process network data in real time whilst providing various alarms and reports according to parameters specified by the user.

Section 7.3. Alarms and Reports

The alarm station produces the following reports and alarms:

- Raise an alarm if more than $x\%$ of packets for a particular test ID are delayed by more than D microseconds (D and x can be specified by the user). The alarm shows the time of detection, the route and packet size on which it occurred and the number of alarms generated by the specific test ID. It should be noted that the delays are calculated over a sliding window of time, the parameters of which can be specified by the user.
- Display a loss report for each packet that has been lost. This is possible by detecting a break in the packet sequence number for a particular test ID. The loss

report shows the time of detection of the packet loss, the route and packet size as well as the cumulative number of packets sent and lost on that network route in the last 24 hours. The cumulative totals can be reset manually on a per test ID basis.

- Packet loss reports are totalled shortly after midnight each day for the preceding day. A table shows the total number of packets lost on each test ID for each day.
- Periodically, at a user defined interval, the packet delays are summarised in two tables. The first table shows the average delays of all packets over the period while the second ones shows for each test ID the average of the highest y% of delays for that test ID over the same period (y can be specified by the user).
- Raise an alarm if the time between consecutive packet losses concerning a particular test ID is less or greater than a certain threshold, where the threshold can be set on a per test ID basis.
- Raise an alarm if there is a break of more than one minute between packets arriving at the alarm station.

Section 7.4. Synoptic program presentation

The alarm station is written in Microsoft Visual Basic, a Microsoft Windows programming system which offers full exploitation of the graphical user interface [Vis93]. According to Tim Spencer, the author of the original alarm station, the program requires an IBM-compatible PC running Microsoft Windows v3.1 and a Winsock-compliant IP stack [Spe98]. On start-up, the main screen is loaded and the relevant code is executed. It might be useful to mention that Visual Basic is an event-driven programming environment which allows windows (called 'forms' in the visual basic literature) to be created. On these windows various control objects can be placed such as labels, text boxes, frames, command buttons, check boxes, scroll bars, lines etc. Furthermore code can be created and attached to these objects, being executed

whenever a particular event occurs. Some snapshots of the main screen and the various reports / alarms of the program are available in fig. 43, 44 & 45. As the reader can see, the main form consists of three distinct areas: the status section, the packet delay alarm section and the packet loss report section. The menu at the top of the window offers access to the various sub-menus of the application.

Passing onto the actual program mechanics; first of all, a warning is generated in order to remind users that the application is still a prototype and, as such, it should be used for the evaluation of the concept of an on-line network-reporting tool. Two crucial files ensure successful program initialisation; these are the test ID file and the initialisation (SMDS.INI) file, which if not located at "c:\windows\" will cause the program execution to be problematic. The application automatically opens a UDP (User Datagram Protocol) port using all necessary parameters from the initialisation file and listens for incoming packets. Only upon reception of a packet further action will be taken.

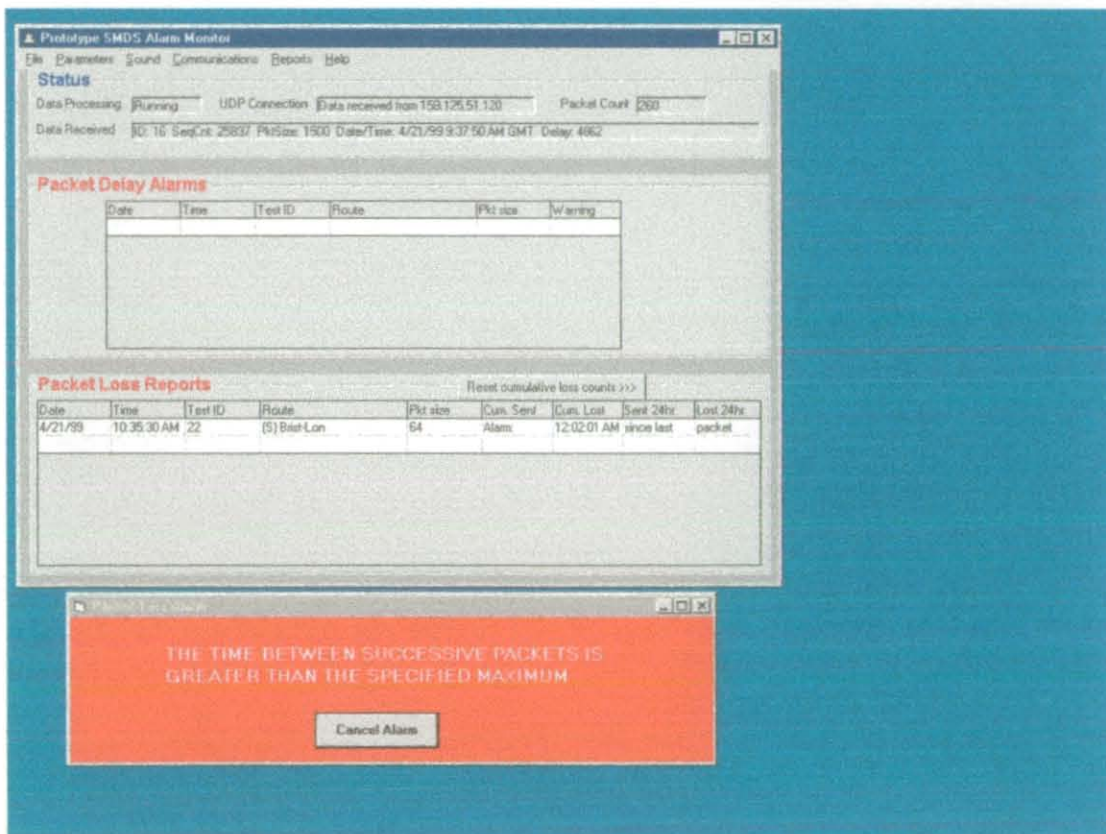


Figure 43: The main screen of the original Alarm Station

Daily Packet Loss Reports

Test ID: 13 Route: (L) Man-Edin Packet size: 64 Close

Test ID	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
4/10/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/11/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/12/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/13/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/15/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/16/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/17/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/18/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/19/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/20/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4/21/99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44: Snapshot of the daily packet loss report screen

Delay Reports for the last 3 hours

Average delay of all packets

Date	Time	Test ID	Route	Pkt Size	Av. Delay
4/21/99	12:00:00 PM	48	(L) Man-Lon	1500	4863
4/21/99	12:00:00 PM	47	(S) Man-Lon	64	4917
4/21/99	12:00:00 PM	46	(L) Man-Edin	1500	5632
4/21/99	12:00:00 PM	45	(S) Man-Edin	64	4846
4/21/99	12:00:00 PM	44	(L) Man-Brist	1500	4843
4/21/99	12:00:00 PM	43	(S) Man-Brist	64	4845
4/21/99	12:00:00 PM	42	(L) Man-Birm	1500	4832
4/21/99	12:00:00 PM	41	(S) Man-Birm	64	4872
4/21/99	12:00:00 PM	40	(L) Lon-Man	1500	5445
4/21/99	12:00:00 PM	39	(S) Lon-Man	64	4843

Average of worst 5% of delays

Date	Time	Test ID	Route	Pkt Size	Delay
4/21/99	12:00:00 PM	48	(L) Man-Lon	1500	5929
4/21/99	12:00:00 PM	47	(S) Man-Lon	64	6997
4/21/99	12:00:00 PM	46	(L) Man-Edin	1500	21114
4/21/99	12:00:00 PM	45	(S) Man-Edin	64	5830
4/21/99	12:00:00 PM	44	(L) Man-Brist	1500	5854
4/21/99	12:00:00 PM	43	(S) Man-Brist	64	5697
4/21/99	12:00:00 PM	42	(L) Man-Birm	1500	5461
4/21/99	12:00:00 PM	41	(S) Man-Birm	64	6057
4/21/99	12:00:00 PM	40	(L) Lon-Man	1500	16697
4/21/99	12:00:00 PM	39	(S) Lon-Man	64	5459

Close

Figure 45: Snapshot of the packet delay reports

While the program displays the main screen, the relevant data is loaded into memory for future use. The rest of the interfaces remain hidden. As long as the UDP port is open all test-IDs and relevant thresholds can be viewed but not altered (with the exception of the data retention period).

The program data is held on disk and loaded into memory each time the alarm station runs. Test-IDs, network routes and packet sizes are kept safe in the two files mentioned previously. Various global arrays are used to hold relevant data and are briefly detailed below:

aryTestIDs: holds a list of the active test-IDs and their relevant parameters such as network route, packet size, previous delay alarms raised by each test ID, number of packets sent / lost per testID, delay bound and some other information useful for the calculation of the various statistics (i.e. recording of the time of a particular packet loss, the number of packets sent and lost for each testID on the last 24 hours etc).

aryDelayAlarms: This is an array of arrays and holds all relevant arrays of the total and delayed packets for each testID, distributed on hourly bins.

AryAverageDelays: This data structure is used to calculate the average delay for each testID. It stores the total accumulated delay per testID as well as the number of packets this total is based on.

AryAllDelays: This is an array of arrays and holds lists of packet delays for each testID. In its original form, it was used to calculate the average of the worst delays for each testID. The reader may suspect that the recorded lists may vary in length, according to the number of packets received with regard to each particular testID.

Passing onto the format of the input data, UDP packets are transferred between the monitor and the alarm stations using PowerTCP UDP control [Spe98]. The actual data is stored in the monitor station as unsigned 32 bit integers and sent as 4 x 8 bit bytes. Each byte is represented as a single character. According to Tim Spencer, since there is no data type for an unsigned 32 bit integer in Visual Basic (only signed 32 bit integers), the first bit of each component can be safely ignored simply because the

expected numbers are not large enough to set this bit. The decimal equivalent can be obtained by taking the ASCII values of the characters and summing up as following:

$$\begin{aligned} \text{Value}_{10} = & [\text{ASC}(\text{1st char})] + \\ & [\text{ASC}(\text{2nd char}) \times 256] + \\ & [\text{ASC}(\text{3rd char}) \times 256^2] + \\ & [\text{ASC}(\text{4th char}) \times 256^3] \end{aligned}$$

The relevant fields transmitted are testID, send time, sequence count, packet length and receive time.

Section 7.5. From the user's viewpoint

In his book *Turn Signals*, Donald Norman who is professor and founding chairman of the Department of Cognitive Science at the University of California (San Diego), has an essay called “coffee cups in the cockpit” [Pr94]. With that essay Prof. Norman expresses his concern for the fact that even in expensive aeroplanes, pilots remind themselves of what they ought to do by improvising such schemes as taking an empty coffee cup and putting it over the flap handles so when they need to use them, the cup will remind them to switch off the excess electrical load before lowering the flaps. From his point of view, rather sadly, coffee cups provide the necessary safety backup within an unusually expensive, complex and highly sophisticated machine.

The previously described Alarm Station might be another good way of demonstrating the potential gap between engineers and users. In this instance, despite the fact that the particular application looks extremely dependable and provides an excellent view to the important network monitoring mechanics, all the relevant interfaces are text-based. Moreover, the only visual help for denoting alarming situations are plain red windows displaying messages of the type “the time between successive packets are greater than the specified maximum” (fig. 43). The current user interfaces require significant work on behalf of the user who is forced to flick through the various windows while searching items within rows and columns of numbers with the lack of

any visual structure. Moreover, the same network operator who will use the original alarm station will be bombarded every second with such messages as **"ID:16 SeqCnt: 25837 PktSize:1500 Date/Time: 4/21/99 9:37:50 AM GMT Delay:4862"** (see fig. 43) and consequently he will probably be unable to continue watching them after a few minutes.

After the relevant study period, the author reached the conclusion that the original alarm station's usability was rather weak, and therefore, there was significant room for improvements with regard to better utilisation of the collected data and developing visual mechanisms for displaying efficiently and effectively the **real-time** network performance measurements. The new objective was to manage to offer to users a global network picture in a fully qualitative (rather than quantitative) manner.

Section 7.6. Deciding the course of action

After the relevant analysis, it was concluded that the BT-Alarm Station provided an excellent chance to test the effectiveness of F.D.V in a real-time system. Subsequently, the author was faced with two options:

- To re-write the entire application using a different programming environment taking advantage of any existing code with regard to FDV (i.e. such as MATLAB)
- To implement FDV in the existing application

For reasons of dependability, simplicity and cost it was decided to proceed with the second option.

7.6.1. Deliverables

In order to justify its *raison d'être*, the supplementary FD-Interface should manage to:

- substitute the red screens with a less panicking and more informative alarm mechanism
- diminish the work involved whenever the user wishes to gain an immediate appreciation of the entire network operation

- enhance the system's real-time operation in the eyes of its users who are presented with only text-based reports.

Section 7.7. Engineering the supplementary FD-Interface

A brief presentation of the development stages can be found in the following sub-sections. The actual programming details have been kept to a minimum.

7.7.1 Deciding DOF, Object Shape and Parameter Mapping.

Taking into consideration that the number of testIDs to be represented summed up to forty (more details in 7.7.2), the issues of **what** and **how** to present on the computer display were paramount. Finally, it was decided to represent the following information (updated every second):

- The longest ever received packet delay per testID
- The shortest ever received packet delay per testID
- Each incoming packet delay
- The average delay per testID being constantly updated with each incoming packet

Having four degrees of freedom to cater for, the generic "E" shape was finally chosen. It should be noted that despite any changes, the core of the application remains unchanged and therefore all the previously mentioned features are still available. Also, most of the calculations are still done over a sliding window of time therefore at regular intervals defined by the step size all data are cleared. Consequently, whenever this happens the FD-Interface gets reset. The accommodation of the parameters is as following: a) maximum delay {upper horizontal element}, b) average delay {middle horizontal element}, c) minimum delay {lower horizontal element} and d) incoming packet delay {vertical side}.

7.7.2 FD-Object Manipulation and related details

The object manipulation is accomplished by employing two control arrays, **FD_TEST_ID** (actual objects) and **FD_TEST_ID2** (calibrated shadows). A control array is a group of controls that share the same name and type [Vis93].

Passing onto the actual mechanics, both control arrays dictate the appearance of 320 separate lines which form 80 Es; that is 40 Es on two separate layers (fig.46). Each layer is a 5 x 8 matrix. The mapping of the control array subscripts is done in a fairly simple manner. Given **A** is a valid row subscript from **aryTestIDs** and **B,C,D,E** are the corresponding subscripts for the four lines of any particular E letter (**B**:upper, **C**:vertical, **D**:lower and **E**: middle line), then the following is true:

$$B = (A \times 4) - 3, C = (A \times 4) - 2, D = (A \times 4) - 1 \text{ and } E = (A \times 4)$$

It has been already mentioned that the number of the testIDs to be represented are forty. This results from the fact that there are five network points between which tests are conducted, namely 1)Birmingham, 2)Bristol, 3)Edinburgh, 4)London and 5)Manchester resulting in $\{^5P_2 = 5! / (5-2)!\}$ [Stro91] twenty network routes each carrying packets of 64 and 1500 bytes. Every testID is a four-digit number whose first digit is always '2'. The remaining three ones declare the **Ingress Point** (from where the packet comes), the **Egress Point** (to where the packet goes) and the packet size.

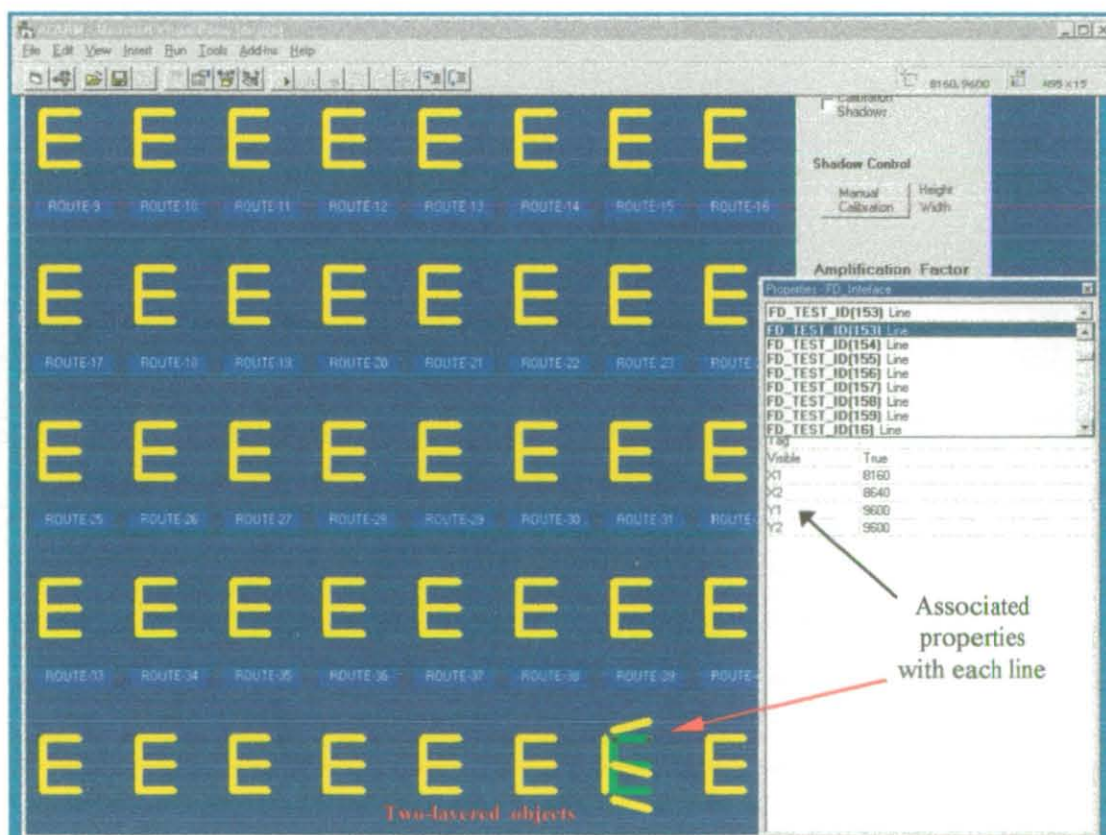


Figure 46: An insight to the developed prototype during the late stages of its creation

As an example consider the testID 2010 (packet arriving from Birmingham [0], going to Bristol [1] while carrying 64 bytes of data [0]).

7.7.3 Relevant Structure Diagrams

The following Jackson Structure Charts (fig.47a,b,c) briefly describe the real-time operation of the supplementary FD-Interface.

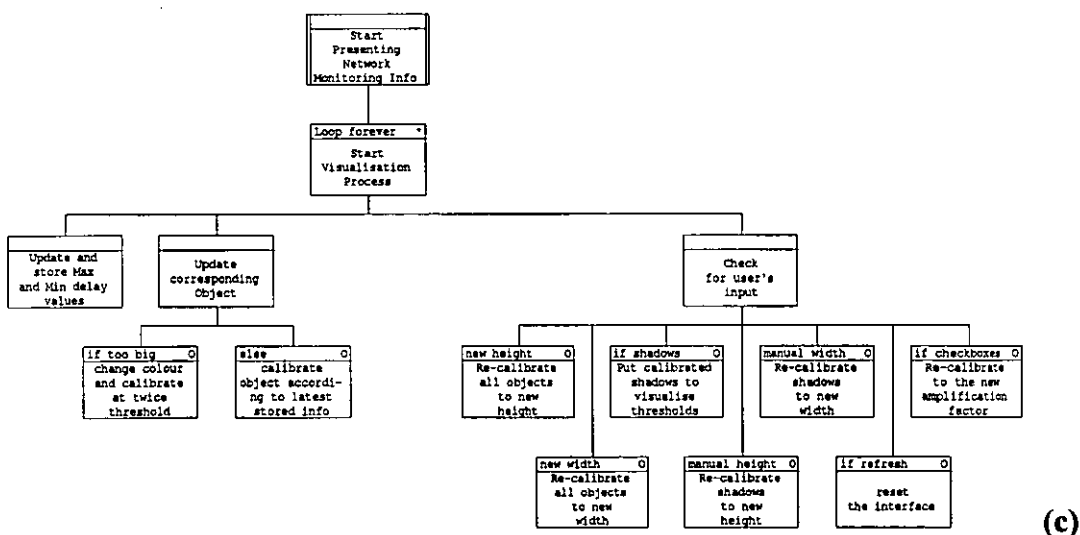
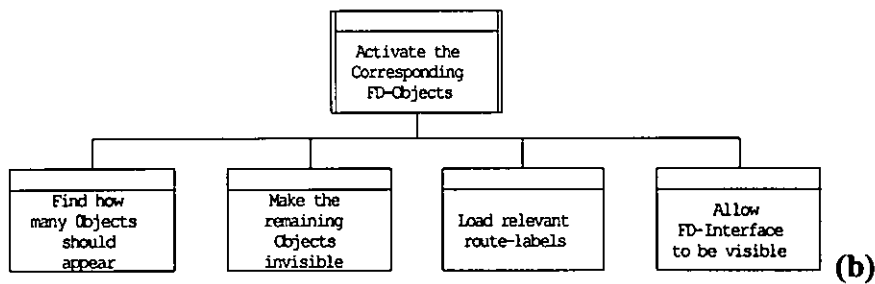
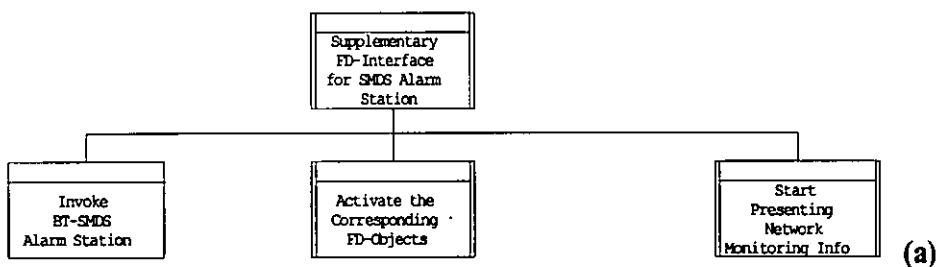


Figure 47: Jackson Structure Charts of the prototype FD-Interface for SMDS alarm station

7.7.4. Relevant code segments

With regard to the previous charts, two samples of the code developed in Visual Basic are listed below:

Segment No.1.

code segment details: GLOBAL9.BAS (filename)

General (object)

LoadActiveTestIDs (procedure)

```

.
.
.
' Open FD-Interface and activate respective Visual Objects
  FD_i = 0
  Do While (40 - FD_i) > NumTestIDs
    FD_Interface.Label(40 - FD_i).Visible = False
    FD_Interface.FD_TEST_ID(4 * (40 - FD_i)).Visible = False
    FD_Interface.FD_TEST_ID(4 * (40 - FD_i) - 1).Visible = False
    FD_Interface.FD_TEST_ID(4 * (40 - FD_i) - 2).Visible = False
    FD_Interface.FD_TEST_ID(4 * (40 - FD_i) - 3).Visible = False

    FD_i = FD_i + 1
  Loop

  ' Load Respective Labels with the Proper Active Route String
  FD_i = 1
  If (NumTestIDs > 40) Then ' Error Message
    MsgBox ("This version of FD-Interface doesn't support more than 40 routes")
    NumTestIDs = 40
  End If

  Do While (FD_i <= NumTestIDs)
    FD_Interface.Label(FD_i).Caption = aryTestID(FD_i).Route
    FD_i = FD_i + 1
  Loop
  FD_Interface.Show
.
.
.
.

```

Segment No.2.

code segment details: MAIN18.FRM (filename)

UDP2 (object)

OnRecv (procedure)

```

.
. Previous Stage (see fig.48)
.
' Updating activated FD-Objects
Height = FD_Interface.SliderHeight.value
Width = FD_Interface.SliderWidth.value
MostRecentDelay = UBound(aryAllDelays(Row).aryDelaysPerID) - 1
' Restoring Y1 for each Object to Left-Bottom Coordinate Y2
' Updating Objects with Already Received Packets Delays
' Calculating Object Height (Parameter: packet delay)
FD_Interface.FD_TEST_ID((Row * 4) - 2).Y1 = FD_Interface.FD_TEST_ID((Row *
4) - 2).Y2 Stage A

Difference = (Height * _
    aryAllDelays(Row).aryDelaysPerID(MostRecentDelay) * HowSensitive /
    aryTestID(Row).DelayBound)

If (aryAllDelays(Row).aryDelaysPerID(MostRecentDelay) >
    (2 * aryTestID(Row).DelayBound)) Then

    'inform User with a change of color to the respective bar
    FD_Interface.FD_TEST_ID((Row * 4) - 2).BorderColor = &HC000C0
    Difference = 1300

    'Prohibiting Massive Objects in order to preserve interface-elegance
Else
    FD_Interface.FD_TEST_ID((Row * 4) - 2).BorderColor = &HFF00&
End If

FD_Interface.FD_TEST_ID((Row * 4) - 2).Y1 =
    FD_Interface.FD_TEST_ID((Row * 4) - 2).Y1 - Difference Stage B
FD_Interface.FD_TEST_ID((Row * 4) - 3).Y1 =
    FD_Interface.FD_TEST_ID((Row * 4) - 2).Y1 Stage C
FD_Interface.FD_TEST_ID((Row * 4) - 3).Y2 =
    FD_Interface.FD_TEST_ID((Row * 4) - 3).Y1 Stage D

FD_Interface.FD_TEST_ID(Row * 4).Y1 =
    FD_Interface.FD_TEST_ID((Row * 4) - 2).Y2 - (Difference / 2) Stage E
FD_Interface.FD_TEST_ID(Row * 4).Y2 =
    FD_Interface.FD_TEST_ID(Row * 4).Y1 Final Stage

'Calculating object Width relevant to Middle Bar (Parameter: Average Delay)
FD_Interface.FD_TEST_ID(Row * 4).X2 =
    FD_Interface.FD_TEST_ID(Row * 4).X1 'restoring width

```

$$\text{Difference} = (\text{Width} * (\text{aryAverageDelay}(\text{Row}).\text{TotalDelay} / \text{aryAverageDelay}(\text{Row}).\text{NumberOfPkts}) * \text{HowSensitive} / \text{aryTestID}(\text{Row}).\text{DelayBound})$$

```
.  
.   
.   
FD_Interface.FD_TEST_ID(Row * 4).X2 =  
FD_Interface.FD_TEST_ID(Row * 4).X2 + Difference  
_Interface.FD_TEST_ID((Row * 4) - 1).X2 + Difference  
  
.   
.   
.   
DoEvents
```

Fig.48 presents a pictorial representation showing the various stages which take place prior to an object being updated with respect to its new height.

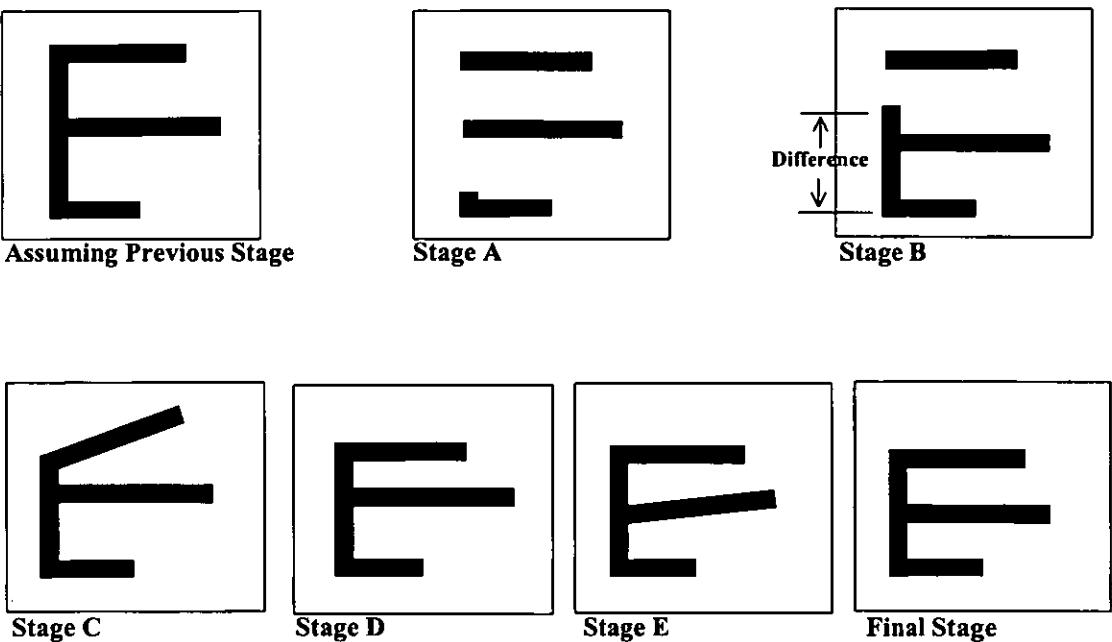


Figure 48: A pictorial representation of how an object's height gets updated

7.7.5. The finished Interface

After implementation of the required changes and various improvements, the final interface layout can be seen in fig.49 & 50. Each time the application starts, the FD-Interface pops up automatically. While the application runs, the user can click on the various sliders to re-calibrate the visual objects by either changing their height or

width. All relevant information is updated on the receipt of a new packet. As can be seen, the advantage of the prototype is the absence of any updateable text-based information and as a result a quick look at the screen provides an immediate appreciation of all routes. Various features such as shadow calibration and object sensitivity to different delay measurements are also available supporting customisation of the interface according to the current screen space and user-preferences. In order to avoid unwanted inconsistencies, it is not possible to re-calibrate simultaneously both shadows and objects.



Figure 49: Finished Prototype

The arrangement of the objects and their corresponding route labels is done in such a manner that every column lists objects sharing the same packet size while those in the same row have the same ingress point.

By using such a configuration, the effect of a malfunction or extensive load is visualised a lot easier. Also general conclusions with regard to network traffic can be visibly verified.

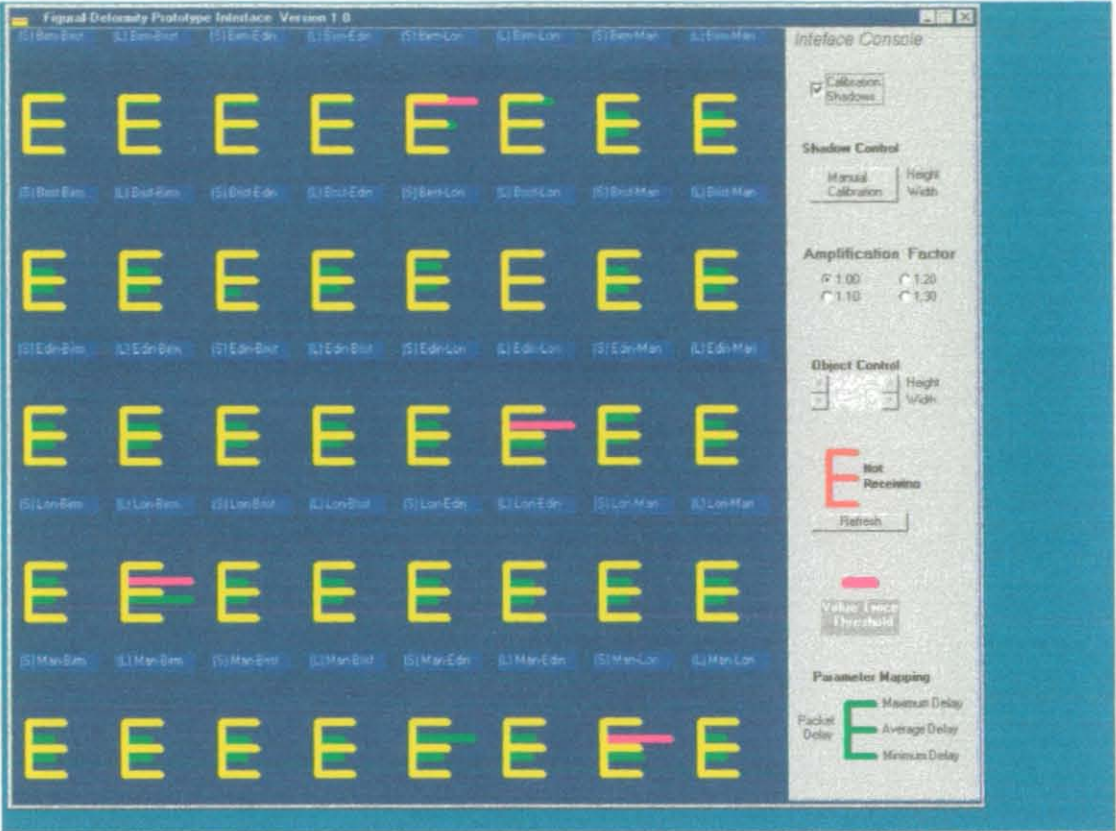


Figure 50: Snapshot with activated shadows visualising the relevant thresholds

Section 7.8. Prototype Evaluation

Once the interface development was completed, the supplementary FD-Interface was presented at Walsall NOU as part of an informal usability test with seven people from Walsall’s operations staff. In order not to create any anomalies on BT’s network monitoring equipment (since the current version of the FD-Prototype is not intended for deployment at the moment) the monitor station was configured to duplicate and send out extra packets for the additional machine which was used for the evaluation.

The evaluation process consisted of two parts: a) the presentation of the new interface visualising real data from BT-SMDS accompanied by a brief introduction on Figural Deformity Visualisation and b) asking staff to answer three likert-type questions as well as strongly encouraging them to give verbal feedback.

7.8.1. Users' reaction

Users' reaction was very positive; all users commented that the new visual mechanism seemed to be very useful and they noticed how much easier is to work with the new FD-Interface instead of the text-based interfaces of BT's original Alarm Station. It was made clear that the prototype was only an initial implementation of the author's idea built only for evaluation purposes at this time. Nevertheless, they all seemed to fully appreciate the usability of the proposed visualisation without any signs of hesitation and despite the lack of any previous adaptation to such graphical user interfaces.

7.8.2 Users' answers

The questionnaire that was passed to Walsall's operations staff is shown on the following page. With respect to the first question, all seven operators answered they strongly agree that the supplementary interface is able to summarise vast amounts of data in a single screen. Passing onto the second one, three simply agree while four strongly agree that the lack of any text-based information proved beneficial to their perceptual capabilities. Regarding the third question, five out of seven strongly agreed and two simply agreed with the remark that calibrated shadows constitute a strong visual aid. The results are graphically demonstrated in fig.51.

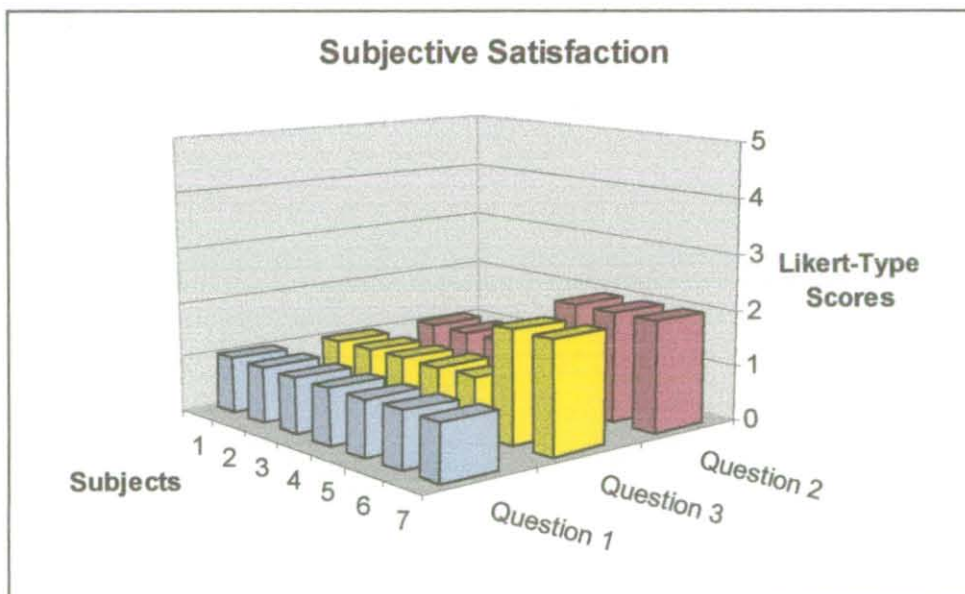


Figure 51: Summary of the subjective answers

FD-INTERFACE EVALUATION CODING SHEET

Identification:

DATE ____/____/____

Age of Subject: ____

Computing Experience: ____ Years

Study Conditions:

____ Experimental Setting ____ Regular Working Environment (tick as appropriate)

Likert-Type Scales:

(Please answer the following Likert-Type scales by checking the appropriate number)

- A. Overall, I felt that the supplementary FD-Interface improves significantly the efficiency & effectiveness of the SMDS Alarm Station offering the potential to summarise vast amount of data in a single screen.

1	2	3	4	5
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree

- B. Altogether, the complete absence of any text-based information increased by perceptual capabilities.

1	2	3	4	5
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree

- C. The provision of calibrated shadows proved to be a strong visual aid which allowed easy and fast detection of packet-delay measurements

1	2	3	4	5
Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree

7.8.3 Suggestions

Two operators mentioned on their written comments that the calibrated shadows could be transparent instead of solid. A few others commented on the possibility of mapping more parameters per object.

Section 7.9. Conclusion

As Ellis has mentioned [Elli96], the level of effort to convert an interface technique into an actual product design is significant. The process of designing and activating the supplementary FD-Interface consumed considerable time and several different attempts before reaching a satisfactory solution. There may be a few more minor issues to be resolved with the existing prototype but what is most important is that the people for whom FD-Interfaces were originally conceived and developed seem to fully appreciate the advantages of the proposed visualisation.

CHAPTER

8

CONCLUSION

Section 8.1. Discussions & Conclusions

The evolution of massive internetworking has lead to the formalisation and standardisation of critical areas of network management such as network configuration, service utilisation, security issues, fault detection and network performance measurement. Some of these aspects involve human-computer interaction. Consequently, HCI issues become relevant to the subject matter of communication network management.

User satisfaction should not remain an open question when it comes to network management issues. Improving computer performance has been the main objective of many developers. On the other hand, the philosophy behind designing and developing user interfaces as part of the software components of communication networks seems to remain unchanged and entirely based on the generic waterfall model which caters very little, if any, for user needs. As a result, users are expected to do as they are told and if they simply fail doing so, they are the ones to be blamed. The main ingredient which is missing from the Waterfall model is called “user-centred design”. The philosophy of network management tool designers should change in such a way that can better allow for the catering of usability issues.

Network performance monitoring can generate large amounts of raw data. Experts have mentioned that effective visualisation techniques are critical to make sense of various data sets [Claf99]. However, there are many instances where expensive network monitoring software with promising and highly visualised user interfaces (as the creators put it) are still using text-based screens to present various network data. In the best possible scenario, simple bar-graphs, histograms and, rarely, conventional tree-structures are employed to deliver “high visualisation” for “high-end functionality” implying that such visual mechanisms are the state-of-the art when in fact they are not; as already demonstrated in this thesis.

The presentation layer is usually the intermediate level in the entire performance monitoring process and deals with user-interface design. Its existence is intended to allow users to pass their requests to the monitoring system and accordingly to translate on their behalf the system’s responses. Various tasks are closely linked with the presentation layer of the performance monitoring system (i.e. reports on historical data, alarm reports etc.)

In this thesis, it has been argued that, in certain cases, the user interfaces of current network monitoring products are inadequate to cope with user needs and can be dramatically improved by employing the author’s novel visualisation aid called FDV. The philosophical framework behind FDV is that a successful interface should not be only useful but usable as well, and in order to be so, it needs to be highly visualised. In other words, a successful UI must inflict user awareness and full appreciation of what has been shown. In order for the previous to happen, raw data should give way to graphics.

Considerable experimental work has been done to prove the high efficiency of FDV when implemented in network monitoring environments as opposed to their best rival, bar-graphs. The screen economy and speed of the relevant user interaction while using FDV is seen to significantly increase. Furthermore, the evaluation of the developed prototype FD-interface for BT-SMDS Alarm Station by a small but representative sample of Walsall’s operations staff generated very positive feedback.

Section 8.2. Recommendations for further work

This last section presents some recommendations and potential developments for the continuation of this work.

8.2.1 Detailed information Vs Speed of interaction

It has always been argued in this thesis that FDV enables users to gain an immediate appreciation of a complex situation with just a quick look at a screen. FD-Interfaces have been characterised as a qualitative rather than quantitative tool. However, if one wishes to extend the use of the proposed visual mechanism and completely eliminate conventional text-based screens, this is plausible. Whether or not the overall performance of FDV would dramatically deteriorate is an issue which remains open for discussion and further research will have to take place in this direction before a final conclusion can be drawn. One may hypothesise that it may be reasonable to expect satisfactory performance, even if text is added on the FD-objects, provided that the user is given the option of eliminating parts, or even the whole, of the additional information whenever he experiences information overload.

8.2.2 Combining FDV with existing visualisation innovations

Another modification of the mechanism could be to implement a differential process of object deformation based on data filtering. The proposal can be implemented by making use of some existing visualisation novelties (i.e. the range-selection slider). In this scenario FD-Objects should be allowed to change shape when representing merely data which lies within a user-specified range of values; such a range may constitute the unacceptable deviation from the relevant thresholds.

8.2.3 Additional applications of FDV

FD-Interfaces have a considerable capability of representing graphically large amounts of numerical information with high space-economy. Consequently, this attribute makes them ideal for many other applications related to displaying information not necessarily relevant to networking monitoring. Some potential uses of FDV may involve database management systems in which scanning and seeking

items from a defined range of values includes extensive user input of various queries. Introducing hierarchical presentation of the various FD-Objects might give life to a hybrid interface able to combine all modern visualisation developments such as Tree-maps, Range-selection sliders and FD-Interfaces.

REFERENCES

- Ba86 Bausell R. Barker,
 “A Practical Guide to Conducting Empirical Research”, Harper & Row
 Publishers Inc., New York, 1986
- Bar79 Barton A. Mark, Glass V. Gene,
 “Integrating Studies That Have Quantitative Independent Variables”,
 Annual Meeting of the American Educational Research Association,
 San Francisco, 1979.
- Bas98 Bashir Omar,
 “Management and Processing of Network Performance Information”,
 Doctoral Thesis, Department of Electronic & Electrical Engineering,
 Loughborough University, 1998.
- Bas98b Bashir O., Phillips I., Parish D., Adams J.L. and Spencer T.,
 “The management and processing of network performance
 information”, BT Technology Journal, Vol.16, No.4, October 1998.
- Bla92 Black Uyles,
 “Network Management Standards: the OSI, SNMP and CMOL
 protocols, McGraw-Hill, New York, 1992.
- Car95 Carr D., Jog N., Kumar H., Teittinen M. & Ahlberg C.,
 “Using Interaction Object Graphs to Specify Graphical Widgets”, HCI
 Laboratory, University of Maryland, Maryland, College Park,
 September 1995, CAR-TR-734.
- Ce97 Cedric M. Festin, Stuart Clayman & Søren-Aksel Sørensen,

- "Jitter Analysis", Proc. Fourth Communication Networks Symposium, The Manchester Metropolitan University, Manchester, 7 July 1997, pp. 121-126.
- Cla94 Clarke Brynly,
"The Way Microsoft Excel for Windows works", Microsoft Press, 1994.
- Claf99 Claffy K.,
" *but some data is worse than others*: measurement of the global Internet", National Laboratory for Applied Network Research (NLNR), San Diego Supercomputer Center, University of California, March 1999.
- Da96 Datta S.,
"Some Human Factors Issues in Speech Recognition Systems", Proc. Conf. & Workshop on New Approaches in Computing, Part 1, University College Suffolk, Ipswich, 13 December 1996, pp. 26-31.
- Elg94 Elgot-Drapkin Jennifer, Miller Michael, Gordon Diana, Kraus Sarit, Nirkhe Madhura and Perlis Don,
"Calibrating, Counting, Grounding, Grouping", Proc. American Association for Artificial Intelligence, Fall Symposium on Control of the Physical World by Intelligent Agents, 1994.
- Eli96 Eliot T.S.,
"Challenges in Information Visualisation", BT Laboratories, British Telecommunications plc, 1996.
- Elli96 Ellis Jason, Rose Anne & Plaisant Catherine,
"Putting Visualisation to Work: Program Finder for Youth Placement", HCI Laboratory, University of Maryland, Maryland, September 1996, CS-TR-3692.

- En96 Englander Irv,
 "The Architecture of Computer Hardware and System Software – An Information Technology Approach", Chichester: Wiley, New York, 1996.
- Enr72 Enrick N. L.,
 "Effective Graphic Communication", Auerbach Publishers, 1972.
- Ha89 Hartson Rex,
 "User-Interface Management Control & Communication", IEEE software, 1989, pp. 62-70.
- HUSAT89 Human Sciences & Advanced Technology Research Centre,
 "Colour and Screen Design", Loughborough University, Loughborough, 1989.
- Ja91 Jain R.,
 "The Art of Computer System Performance Measurement", John Wiley & Sons Inc., 1991
- Je96 Jerding D. F. & Stasko J. T.,
 "The Information Mural: Increasing Information Bandwidth in Visualisations", Graphics, Visualisation & Usability Center, College of Computing, Georgia Institute of Technology, Atlanta, October 96.
- Jo91 Johnson Brian & Shneiderman Ben,
 "Treemaps: a space-filling approach to the visualisation of hierarchical information structures", Proc. of the 2nd International IEEE Visualisation Conference, San Diego, October 1991, pp. 284-291.
- Ju92 Jungmeister Walter-Alexander & Turo David,
 "Adapting TreeMaps to Stock Portofolio Visualisation", Human-Computer Interaction Laboratory, University of Maryland, Maryland, 1992, CS-TR-2996.

- Ka96 Kandogan Eser & Shneiderman Ben,
"Elastic Windows: Improved Spatial Layout and Rapid Multiple Window Operations", Proc. Advanced Visual Interfaces (ACM AVI'96), Gubbio, Italy, May 1996, pp. 27-29.
- Kar98 Karkaletsis E., Spyropoulos C. & Vouros G.,
"A knowledge-based methodology for supporting multilingual and user-tailored interfaces", *Interacting with Computers: the Interdisciplinary Journal of Human-Computer Interaction*, Vol.9, No.3, January 1998.
- Ku94 Kumar H., Plaisant C., Teittinen M. and Shneiderman B.,
"Visual Information Management for Network Configuration", HCI Laboratory, University of Maryland, Maryland, June 1994, CS-TR-3288.
- Li92 Liao H. S., Osada M. & Shneiderman B,
"Browsing Unix Directories With Dynamic Queries: An Evaluation of Three Information Display Techniques", University of Maryland, February 1992, CS-TR-2841.
- Lia91 Liao Tie, Seret Dominique,
"Information Representation and Presentation for Integrated Network Management", Proc. of the 1991 Singapore International Conference on Networks, September 1991, Singapore, pp. 143-148.
- Lon84 Long T.,
"Human Factors Principles for the Design of Computer Colour Graphics Displays", *British Telecom Technology Journal*, Volume 2, No.3, July 1984.
- Maha96 Mahajan Rohit & Shneiderman Ben,

- “Visual & Textual Consistency Checking Tools for Graphical User Interfaces”, HCI Laboratory & Institute for Systems Research, University of Maryland, Maryland, May 1996, CS-TR-3639.
- Mat96 MATLAB,
“User’s Guide”, The MathWorks Inc., 1996.
- Moo93 Moore David & McCabe George,
“Introduction to the Practice of Statistics”, Second Edition, W.H. Freeman & Company, 1993.
- Osa93 Osada Masakaru, Liao Holmes and Shneiderman Ben,
“AlphaSlider: Searching Textual Lists with Sliders”, Department of Computer Science, University of Maryland, Maryland, April 1993, CS-TR-3078.
- Pa97 Pagonis A. & Parish D.,
“On the Creation of a Highly Visualised Interface for Network Performance Data Pre-Analysis”, Proc. Fourth Communication Networks Symposium, The Manchester Metropolitan University, Manchester, 7 July 1997, pp. 117-120.
- Phi95 Phillips Iain, Parish David & Rodgers Clive,
“Performance Measurement of the SMDS”, IEE Colloquium on SMDS, London, October 1995.
- Phi96 Phillips Iain, Tunnicliffe Martin, Parish David & Rodgers Clive,
“On the Monitoring and Measurement of Quality of Service of SuperJanet”, 13th Teletraffic Symposium, Strathclyde, March 1996.
- Pr94 Preece Jenny,
“Human-Computer Interaction”, Addison-Wesley Publishing Company, Wokingham, 1994.

- Pre96 Prestwich Steven & Kusalik Anthony,
 “Programmer-Oriented Parallel Performance Visualisation, Department
 of Computer Science, University of Saskatchewan, Saskatchewan,
 Canada, February 1996, TR-96-01.
- Ro91 Robertson K. Philip,
 “A Methodology for Choosing Data Representations”, IEEE Computer
 Graphics & Applications, Vol. 11, 1991.
- Rob99 Robinson A. & Flores P.,
 “Novel Techniques for Visualising Biological Information”, EMBL
 Outstation, The European Bio-informatics Institute, Wellcome Trust
 Genome Campus, Cambridge, 1999.
- Ros94 Rosis F., Carolis B. & Pizzutilo S.,
 “User-Tailored Hypermedia Explanations”, Workshop held in
 conjunction with UM’94 (Fourth International Conference on User
 Modeling), Hyannis, Cape Cod, Massachusetts, USA, August 1994.
- Rot97 Roth S.F. , Chuah M.C., Kerpedjiev S., Kolojejchick J. A. & Lucas P,
 “Towards an Information Visualisation Workspace: Combining
 Multiple Means of Expression”, A journal of Theoretical, Empirical &
 Methodological Issues of User Science & of System Design, Laurence
 Erlbaum Associates, Mahwah, New Jersey, 1997, Vol.12.
- Shne91 Shneiderman Ben,
 “Visual User Interfaces for Information Exploration”, Proc. of the 54th
 Annual Meeting of the American Society for Information Sciences,
 Washington DC, October 1991, vol. 28, pp. 379-384.
- Shne96 Shneiderman Ben,
 “The Eyes Have It: A Task by Data Type Taxonomy for Information
 Visualisations”, Dept. of Computer Science, Human-Computer

Interaction Laboratory and Institute for Systems Research, University of Maryland, Maryland, 1996, CS-TR-3665.

- Si78 Simon J. L.,
"Basic Research Methods in Social Science, The Art of Empirical Investigation", Second Edition, Random House, New York, 1978.
- Sli99 Slidel Timothy,
"Distributed Computing in the Life Sciences", Lead Article, BioInformer, No.3, EMBL, Outstation Hinxton, The European Bioinformatics Institute, Genome, Campus, Cambridge, 1999.
- Smi79 Smith Lee Mary, Glass V. Gene, "Relationship of Class-Size to Classroom Processes, Teacher Satisfaction and Pupil Affect: A Meta-Analysis", Far West Laboratory, San Francisco, 1979.
- Spe98 Spencer Tim,
"SMDS Alarm Station Design", Broadband & Data Networks, Network Transport Engineering Centre, BT Laboratories, IPSWICH, June 1998.
- Stro91 Stroud K. A.,
"Engineering Mathematics", Third Edition, MACMILLAN EDUCATION LTD., 1991.
- Su91 Sung L., Keong L., Cheow L. & Khiang K.,
"NTUnet Campus Network Statistics & Configuration", Proc. of the 1991 Singapore International Conference on Networks, pp. 42-49.
- Tan98 Tanin Egemen, Lotem Amnon, Haddadin Ihab, Shneiderman Ben, Plaisant Catherine & Slaughter Laura,
"Facilitating Network Data Exploration with Query Previews: A study of User Performance and Preference", Department of Computer Science, University of Maryland, Maryland, February 1998, TR98-S.

- Thim91 Thimbleby Harold,
 "User Interface Design", Addison-Wesley Publishing Company,
 Wokingham, 1991.
- Tur92 Turo David & Johnson Brian,
 "Improving the visualisation of hierarchical Treemaps: Design Issues
 & Experimentation", Proc. of the IEEE Conference on Visualisation,
 October 1992.
- URL1 WWW site:- <http://www.wins.uva.nl/search>
 Jargon Lexicon, Faculty of Mathematics, Computer Science, Physics
 and Astronomy, University of Amsterdam, 1999.
- URL2 WWW site:- <http://serendip.brynmawr.edu/bb/neuro/neuro97/week09/>
 Week 09 coursework by Karyn Myers, elaborating on the difference
 between additive and subtractive color mixing, (fully applauded by
 Prof. Paul Grobstein), Serendip Web Site, Bryn Mawr College,
 Pennsylvania, March 1999.
- US86 U.S. Dept. of Commerce, National Technical Information Service,
 "Guidelines for Designing User Interface Software", Bedford, August
 1986.
- Vis93 Microsoft Visual Basic,
 "Programmer's Guide", Microsoft Corporation, 1993.
- Wa96 Wallgren A., Wallgren B., Persson R., Jorner U. & Haaland J.,
 "Graphing Statistics & Data, Creating Better Charts", Sage
 Publications, 1996.
- Wal93 Walpole Ronald & Myers Raymond,
 "Probability and Statistics for Engineers and Scientists", Fifth Edition,
 Macmillan Publishing Company, a Division of Macmillan Inc., 1993.

BIBLIOGRAPHY

David Nancy & Stewart G.W.,

“Hypothesis Testing with Errors in the Variables”, Research Institute of Michigan,
University of Maryland, Maryland, November 1986, TR-1735.

MATLAB,

“External Interface Guide”, The MathWorks Inc., 1996

MATLAB,

“Building a Graphical User Interface”, The MathWorks Inc., 1996

Microsoft Visual Basic,

“Language Reference”, Ver. 3.0, Microsoft Corporation, 1993.

Microsoft Visual Basic,

“Professional Features Book 2”, Ver. 3.0, Microsoft Corporation, 1993.

Nachszunow G.,

“Development of Telecommunications & International Organisations (ATU-ASCO-
PATU-APT-CITEL)”, Publisher: Creteil / Nachszunow, 1989.

Ravindran K. & Bhat K.,

“Performance Engineering of End-systems for High Bandwidth Multimedia
Communications, Dept. of Computing and Information Sciences, Kansas State
University, Manhattan, USA, TR-95-10.

Tanenbaum Andrew,

“Computer Networks”, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

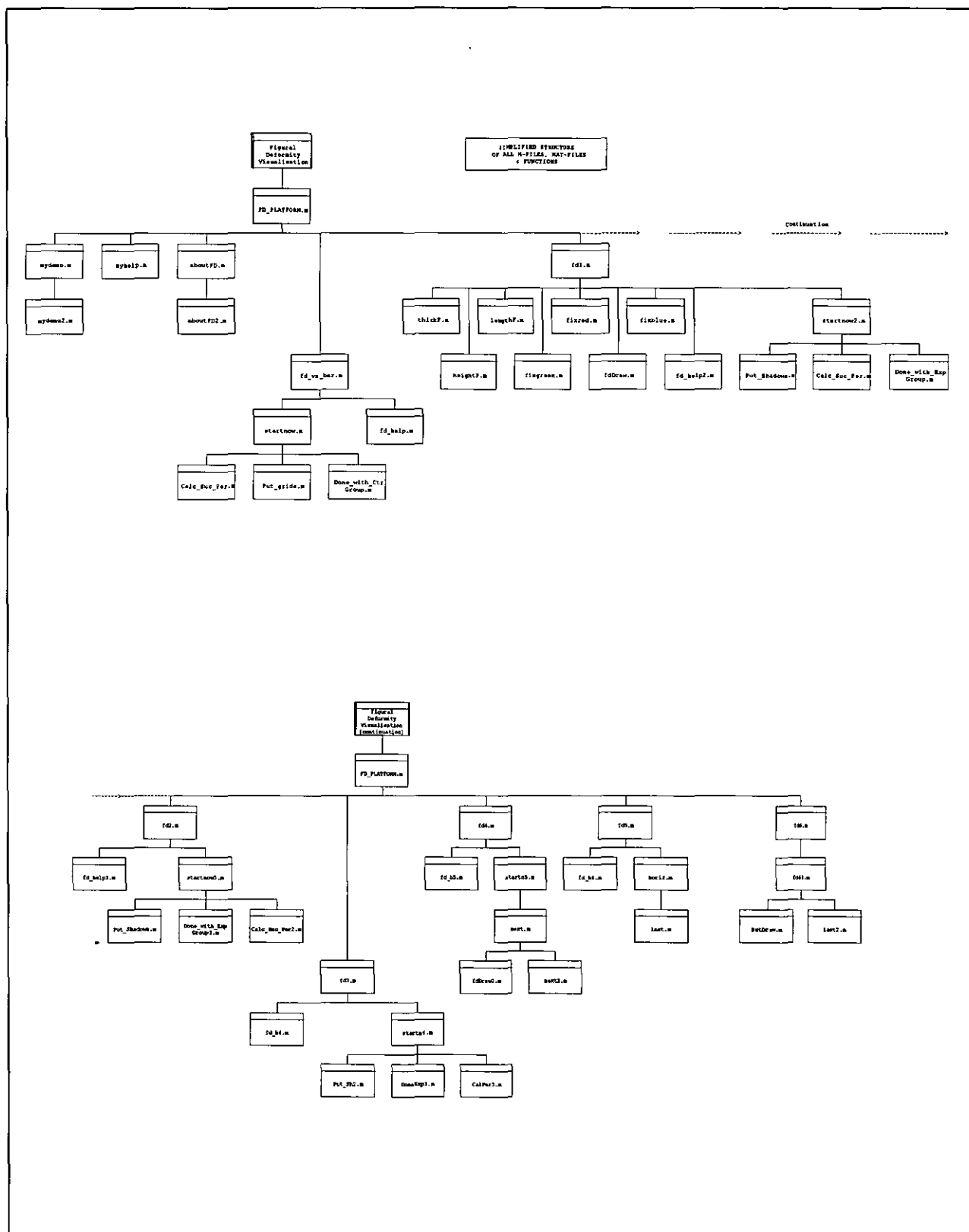
APPENDIX –A

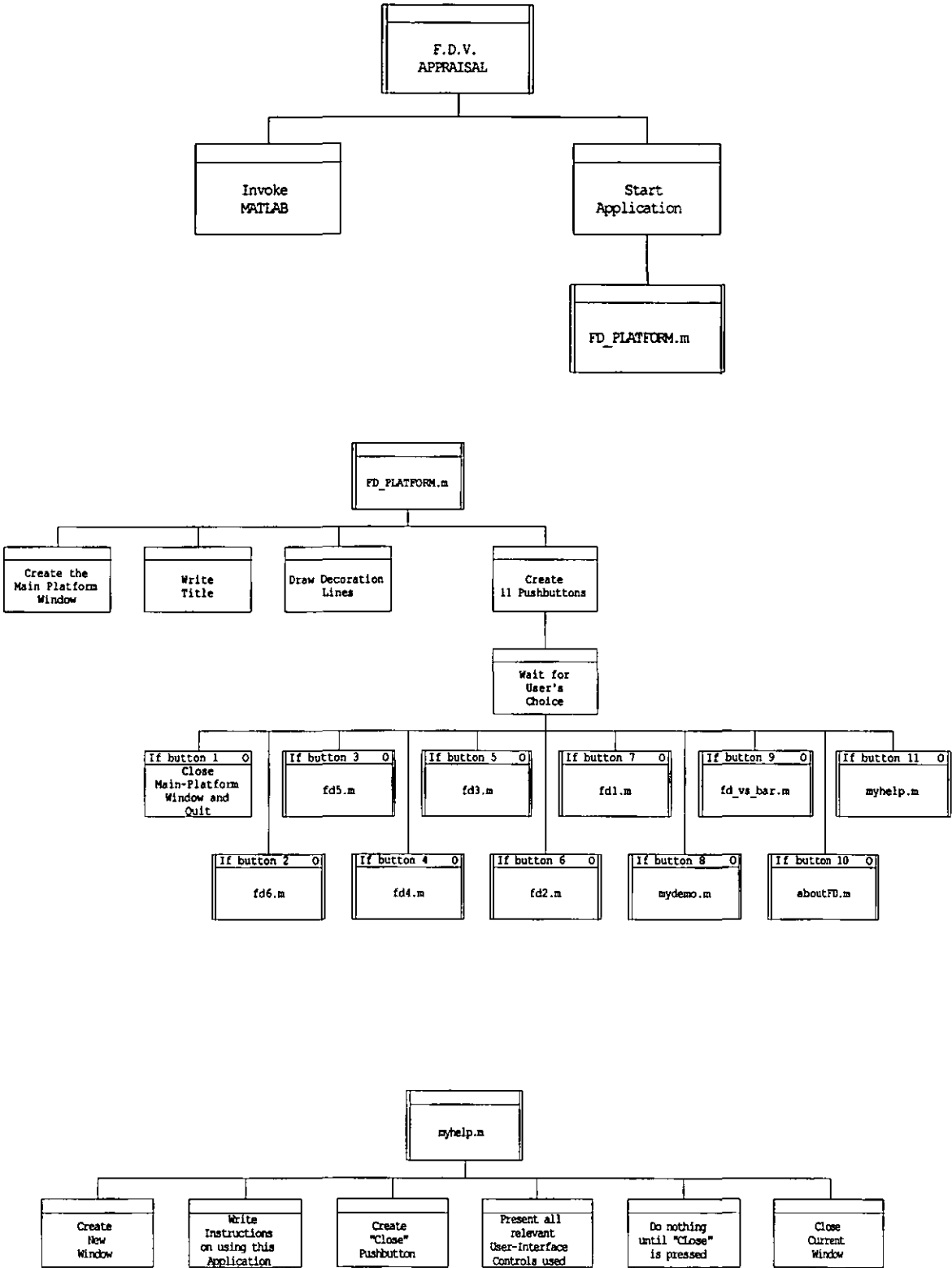
Section A.1. Introduction

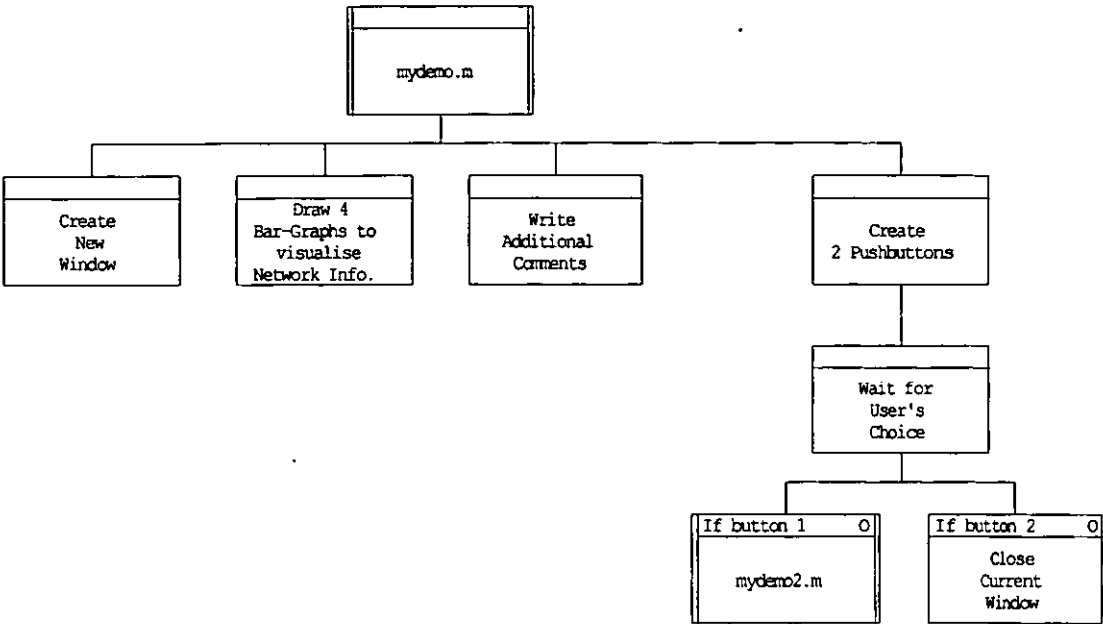
Appendix-A includes all the relevant structure diagrams for the M-Files of the Prototype FD-Interface, discussed earlier in chapter 5. In order to assist the reader, the next page demonstrates the complete M-File *family tree* which allows easier searching through the various charts.

The software-engineering tool which has been used to produce the required diagrams is the SELECT/YOURDON. The created charts are of the ‘Jackson Structure Charts’ type.

GENERAL MAP

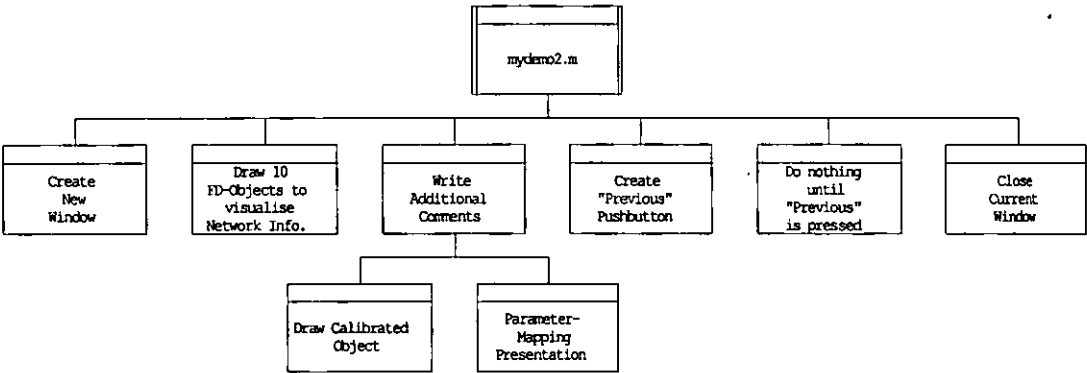


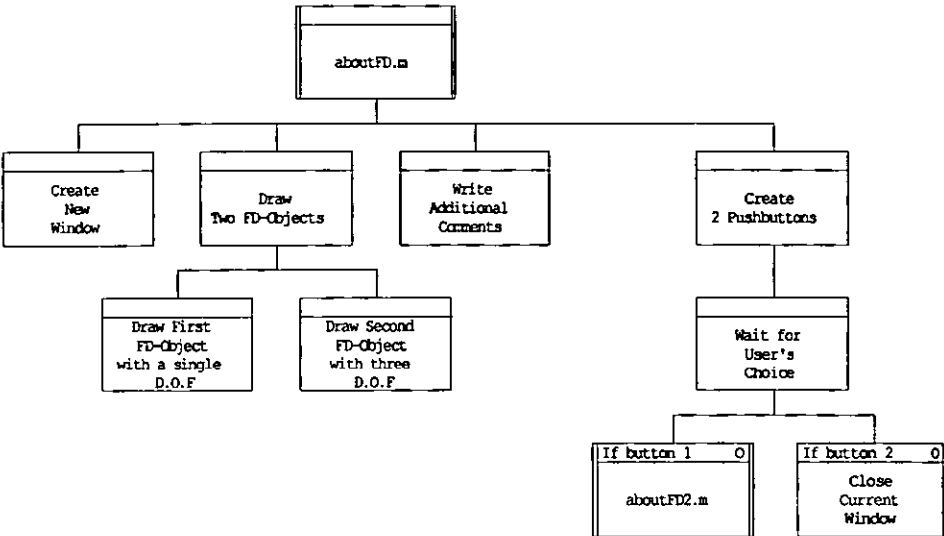




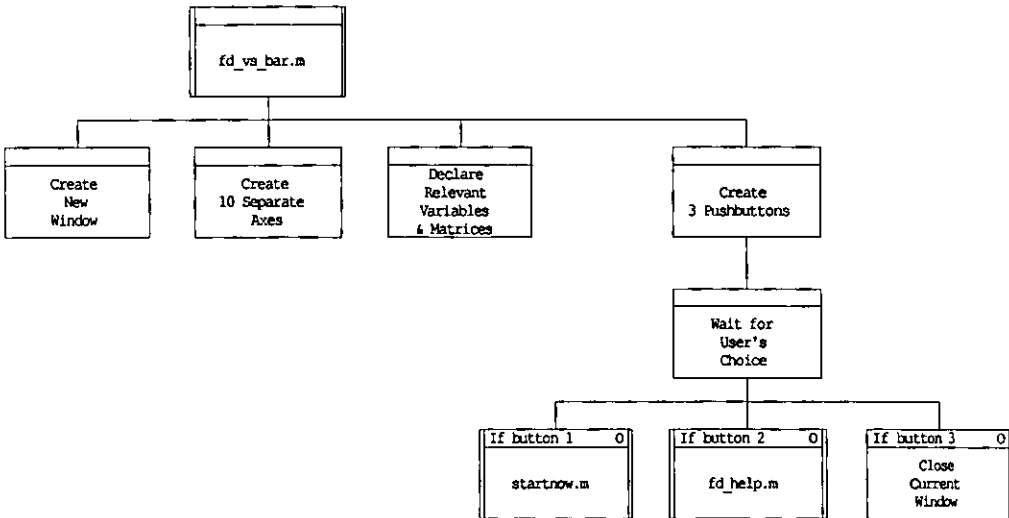
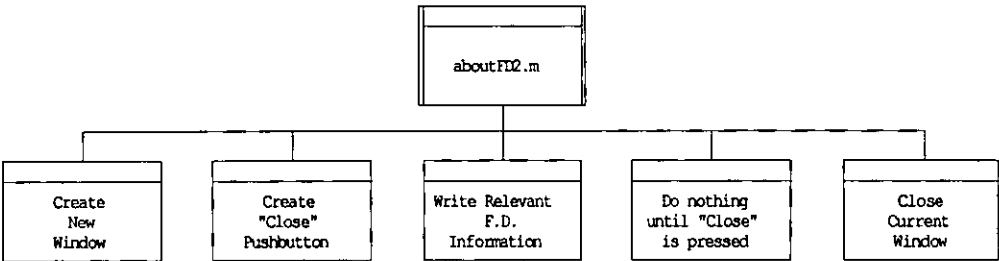
INFORMATION WITH RESPECT
TO NETWORK ROUTES

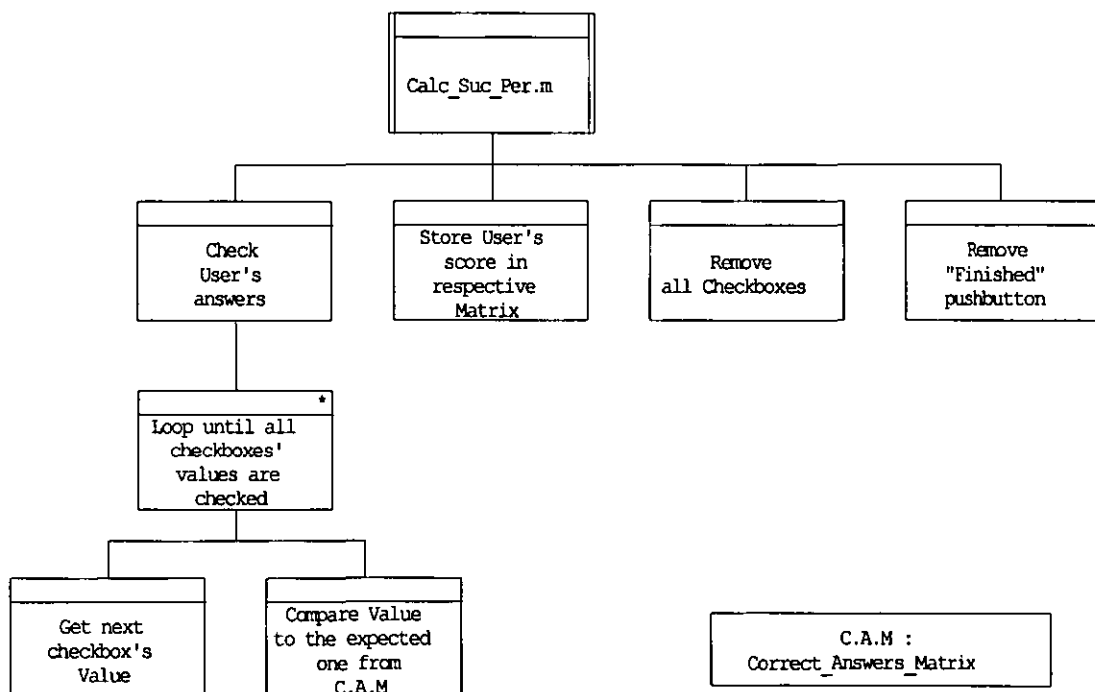
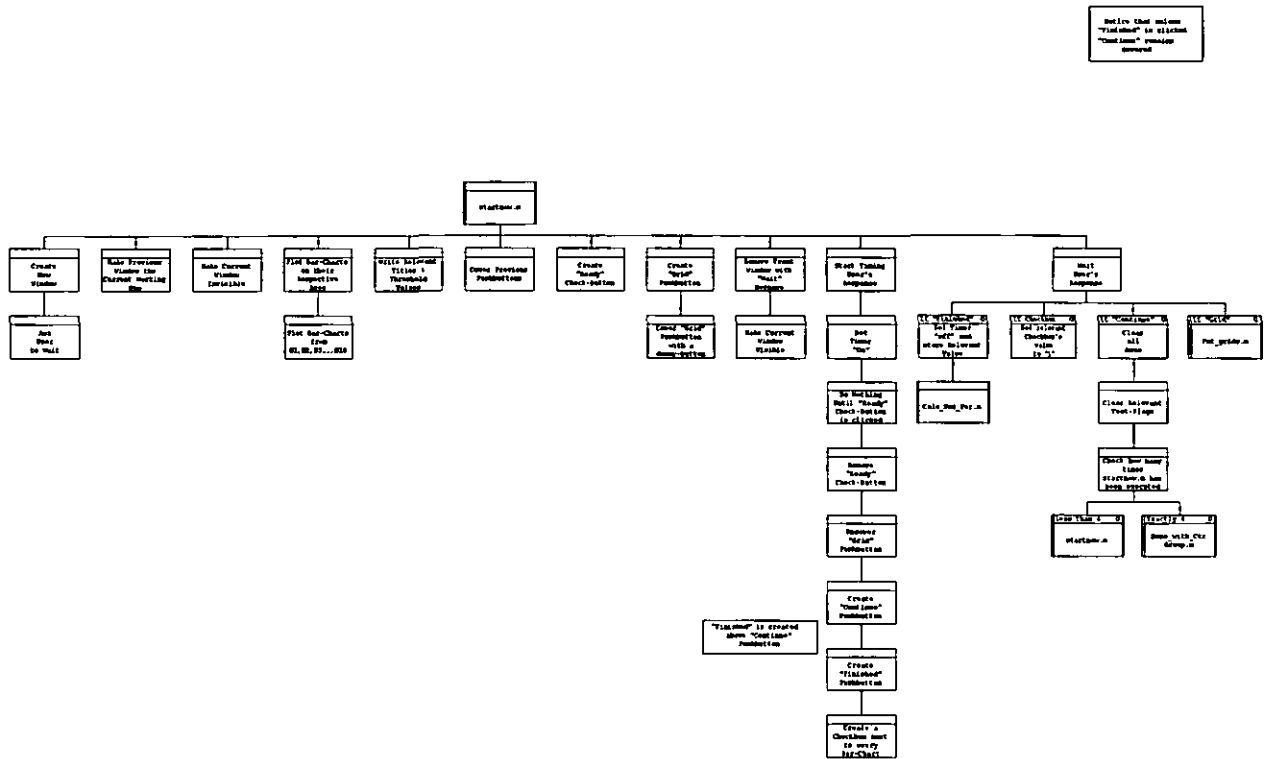
Minimum Packet Delay
Maximum Packet Delay
Average Packet Delay
Average Slowest 5%

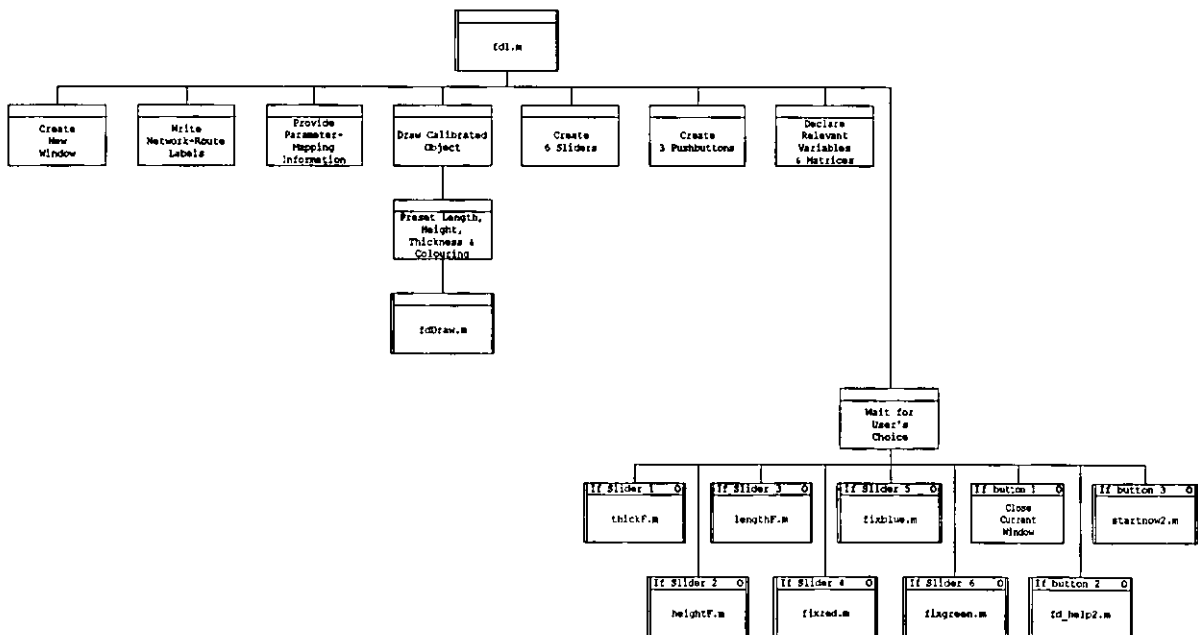
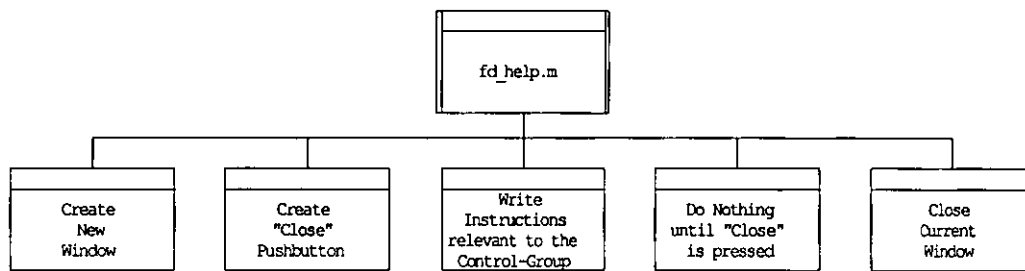
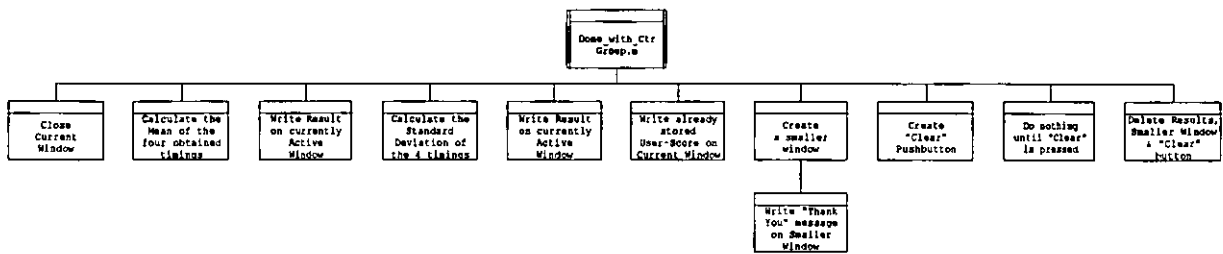
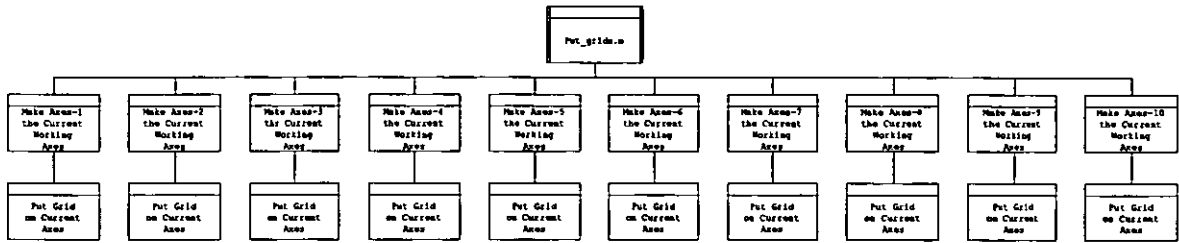


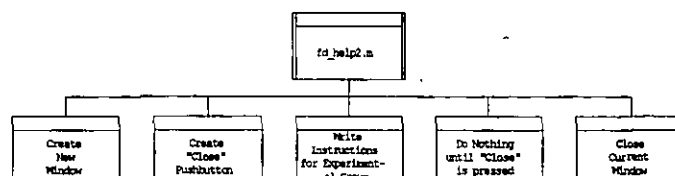
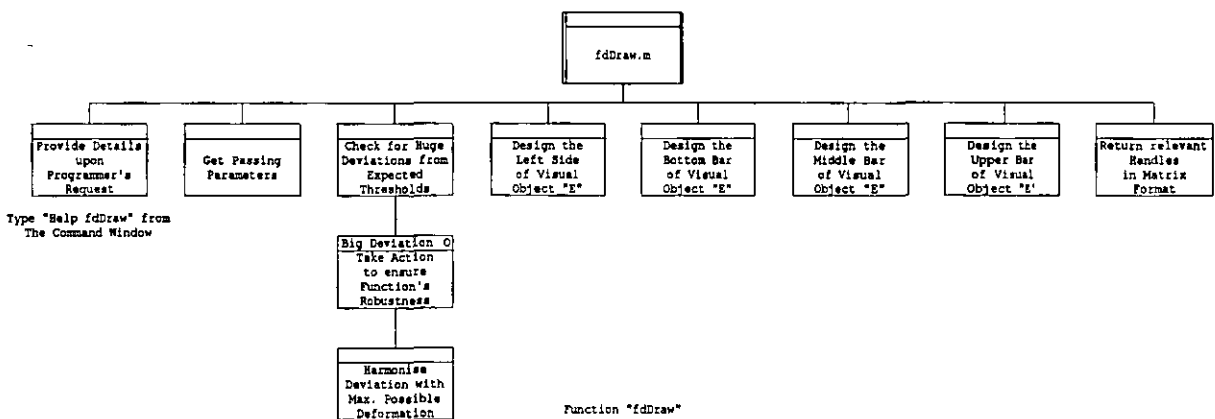
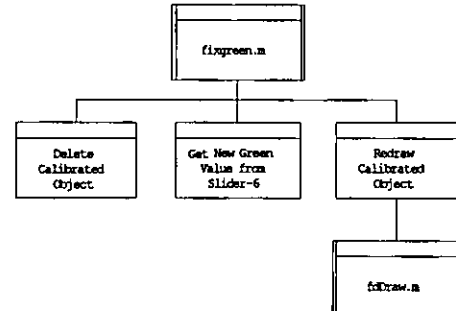
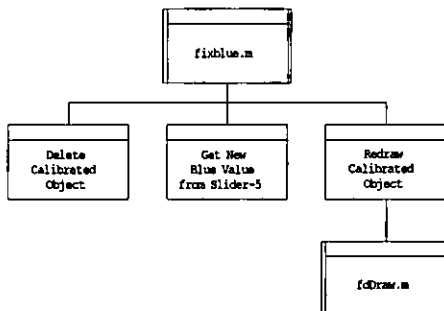
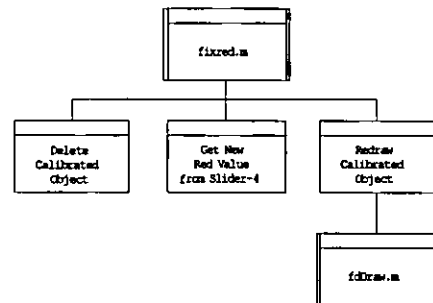
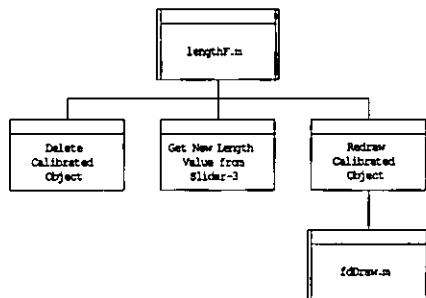
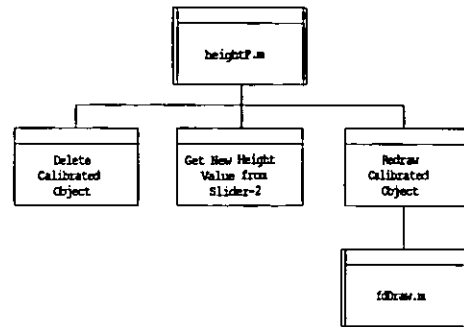
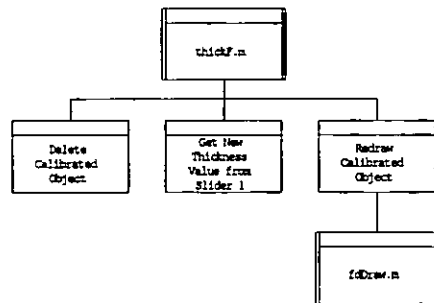


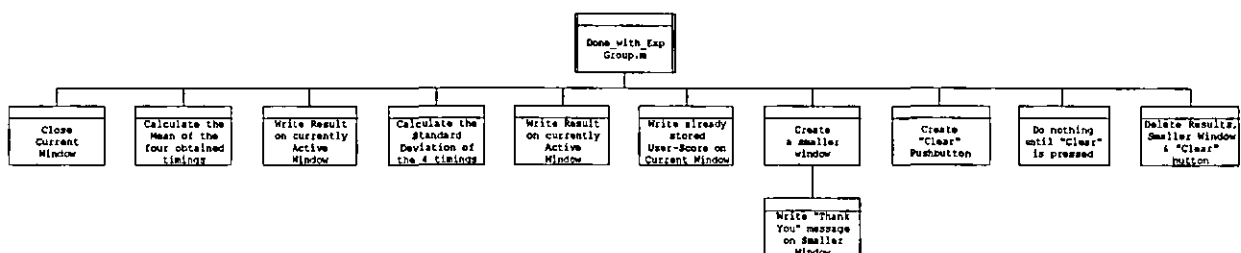
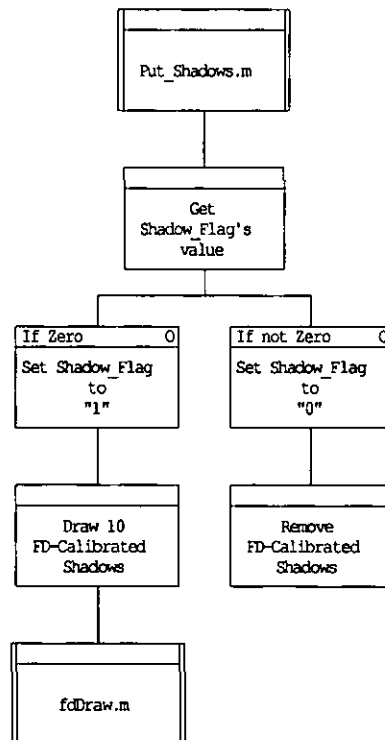
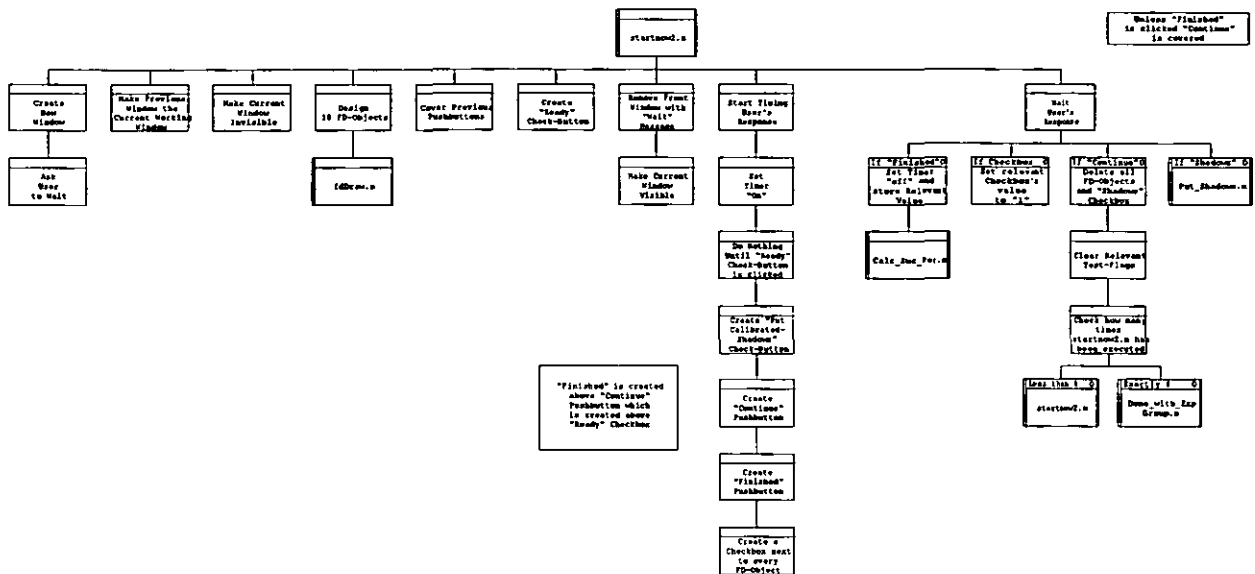
D.O.F : Degrees of freedom

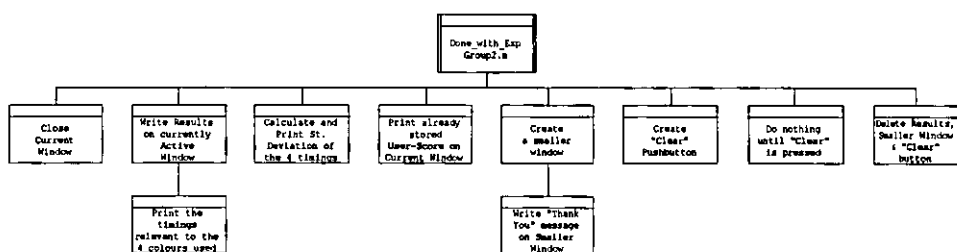
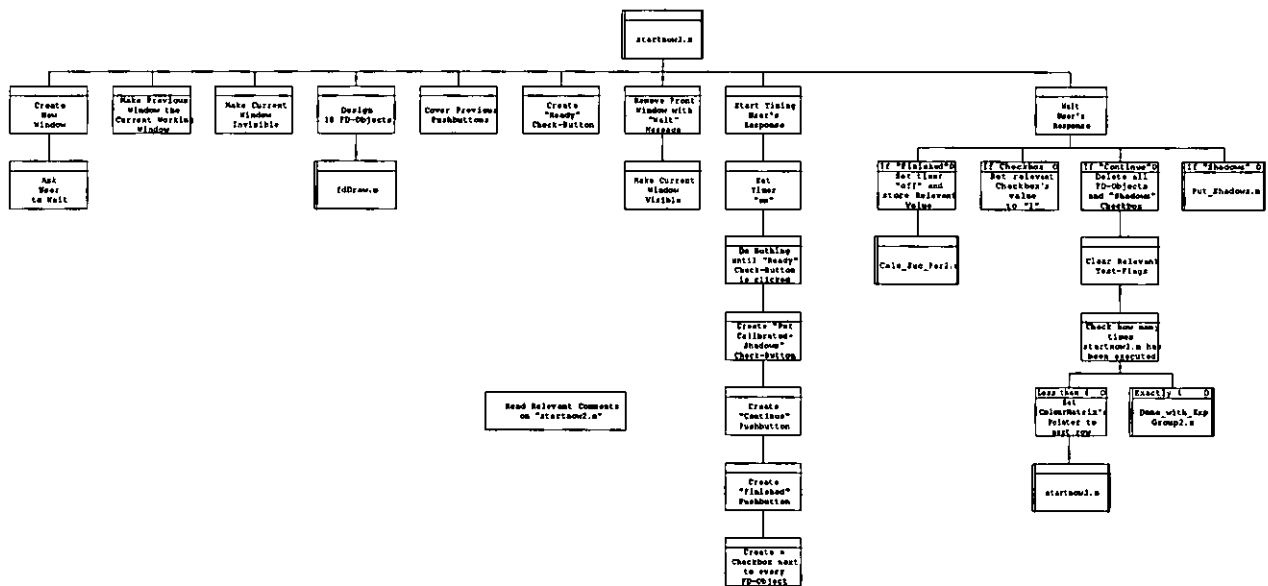
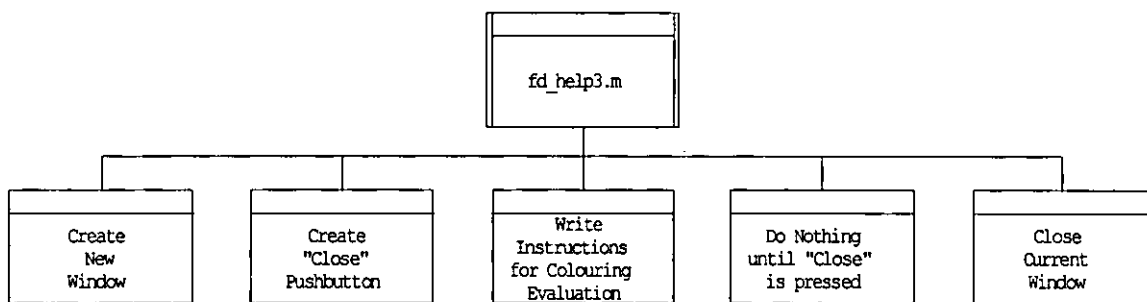
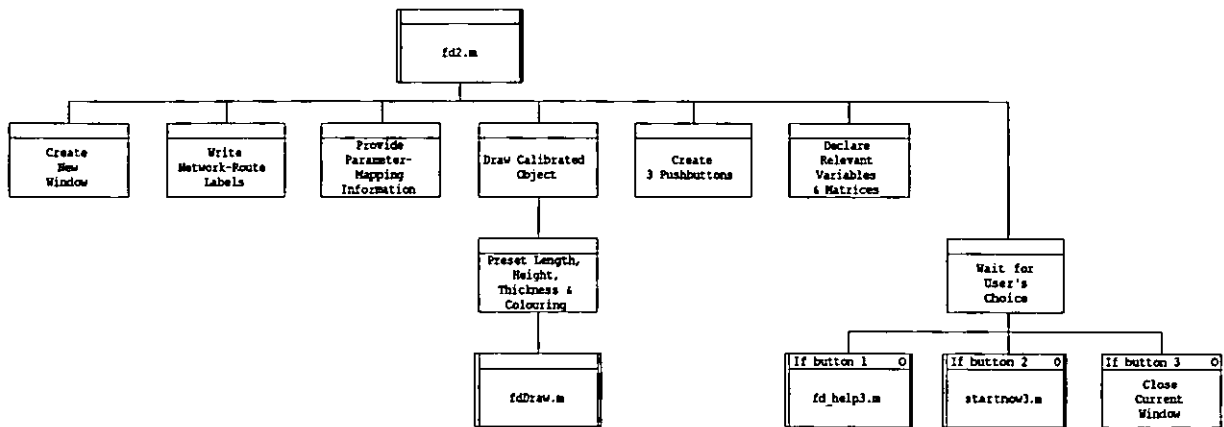


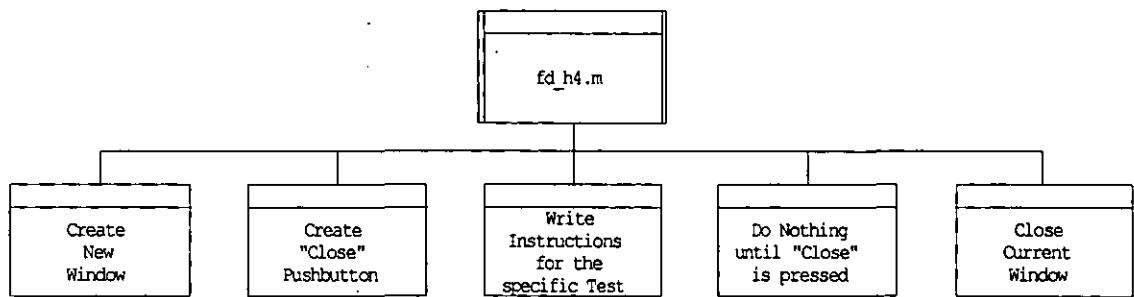
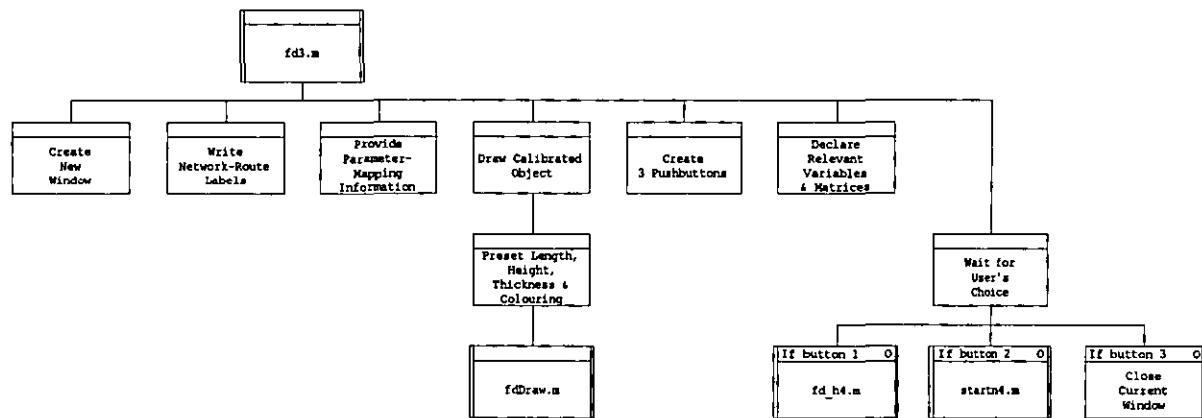
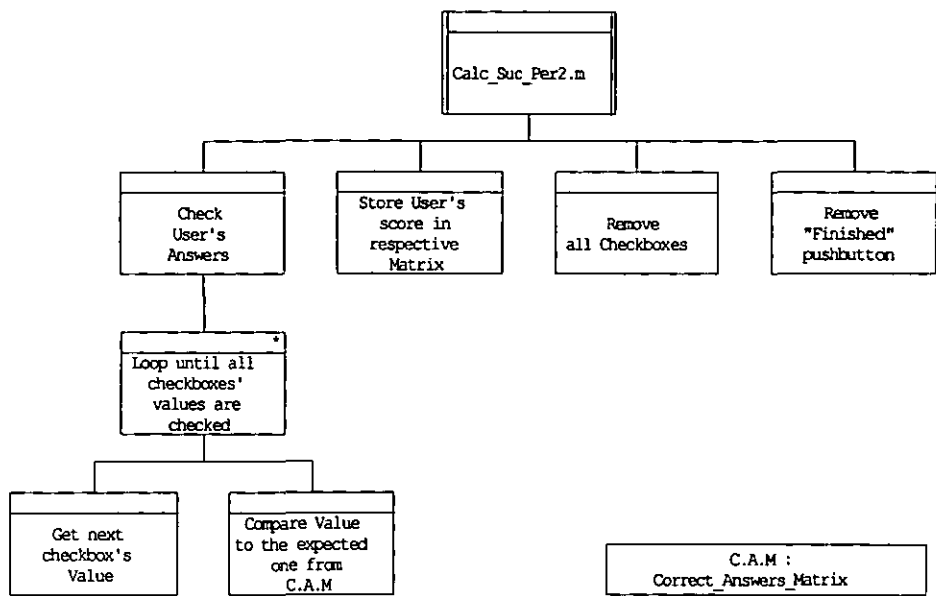


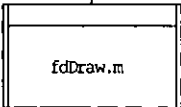
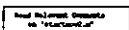


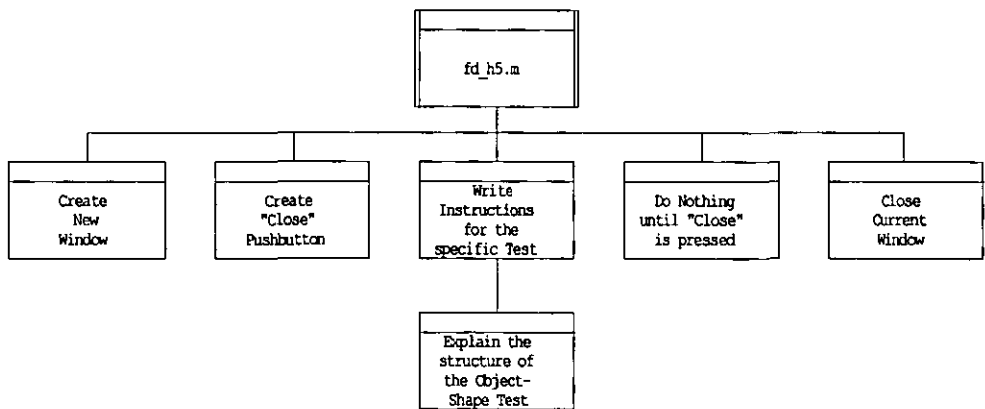
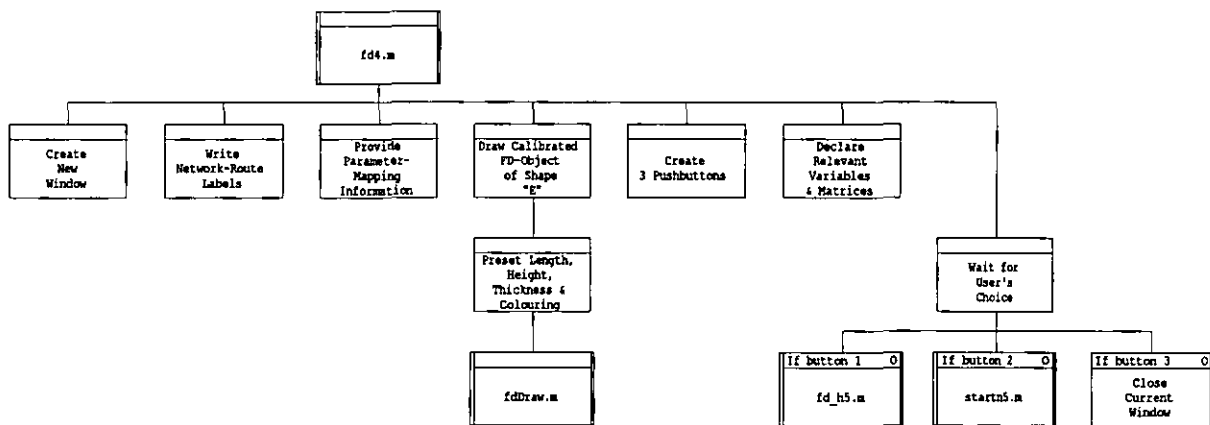
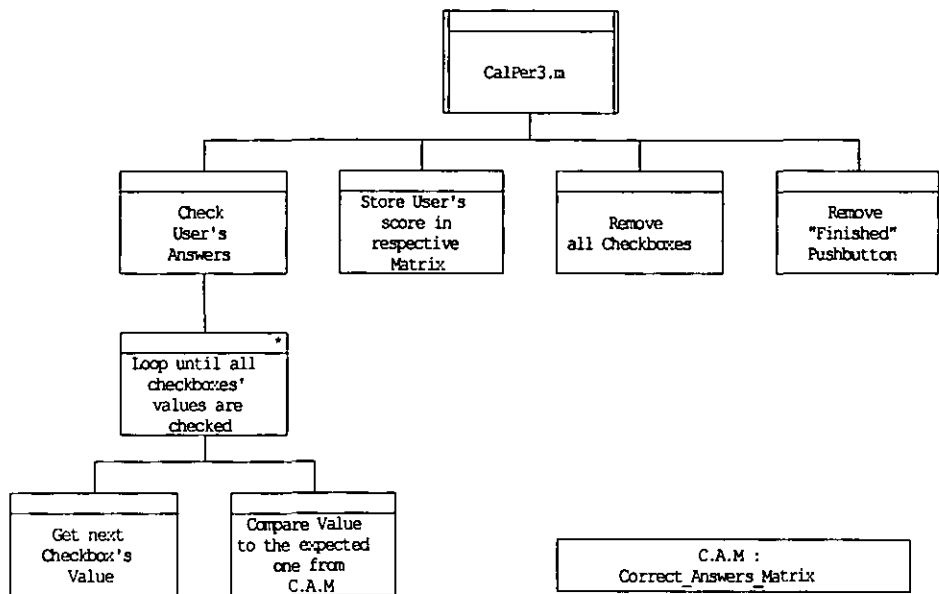


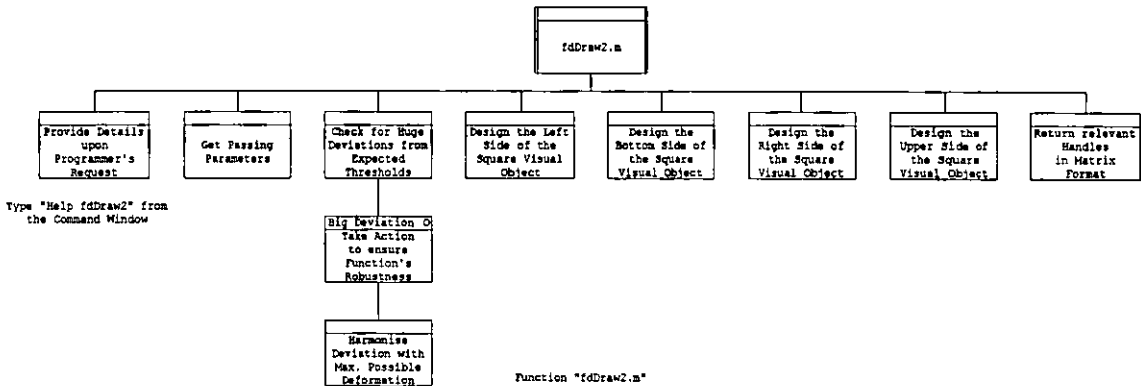
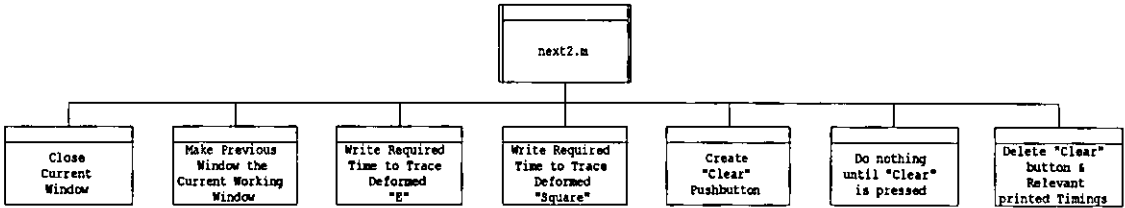
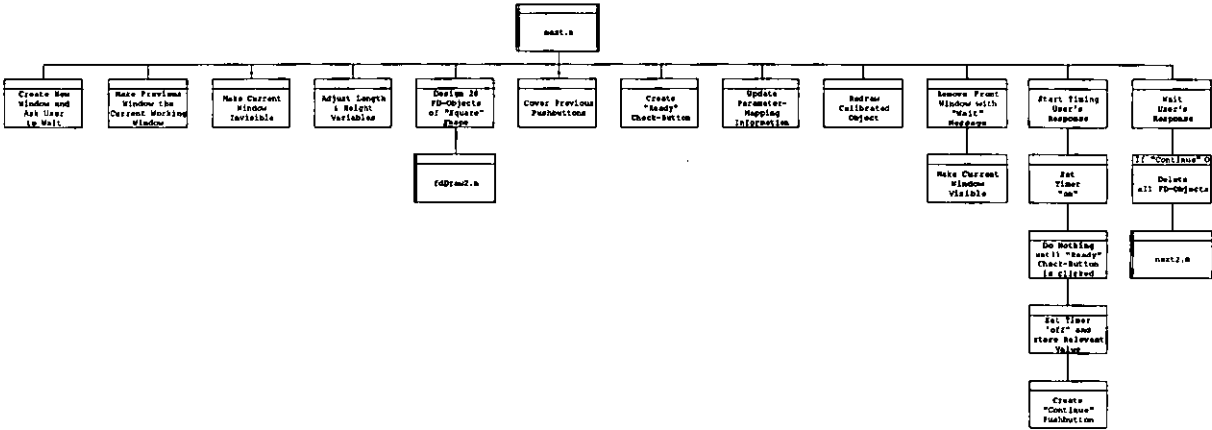
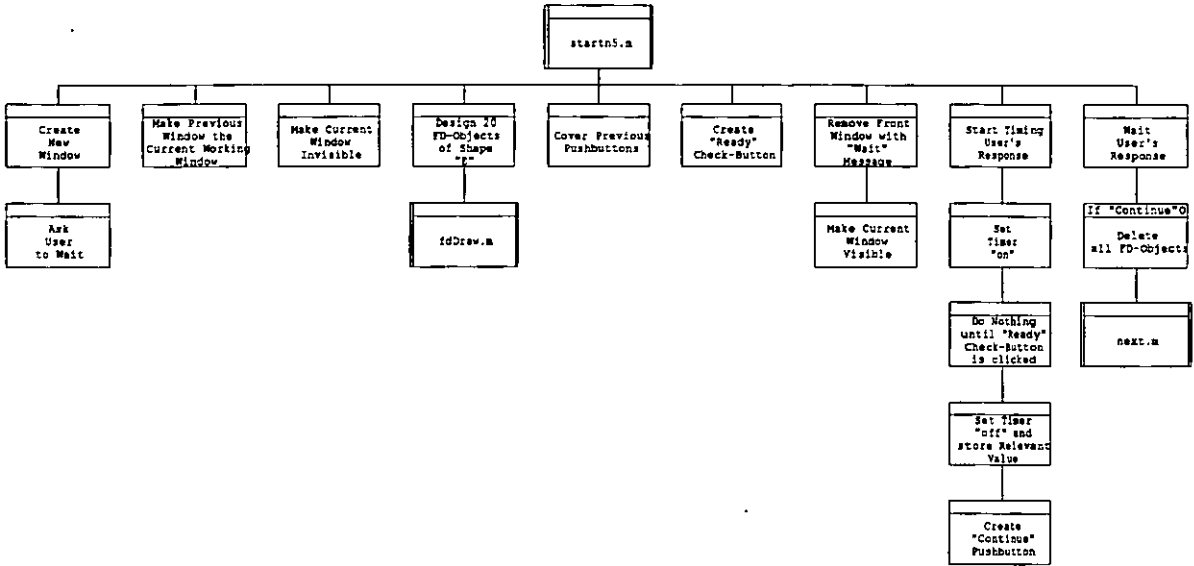


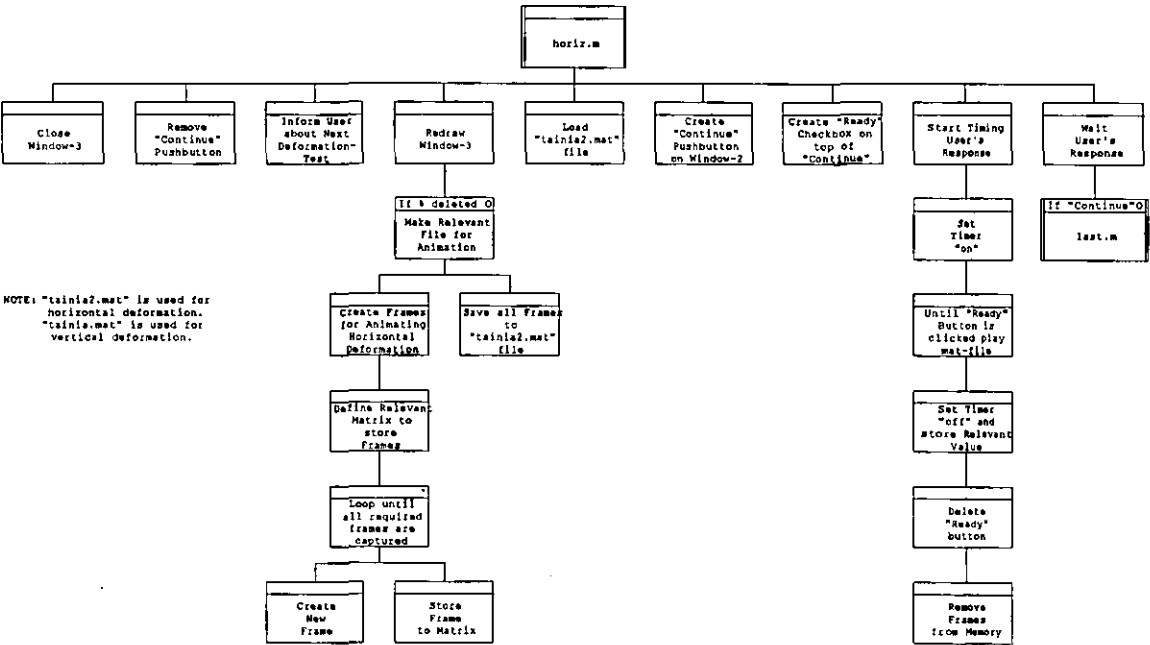
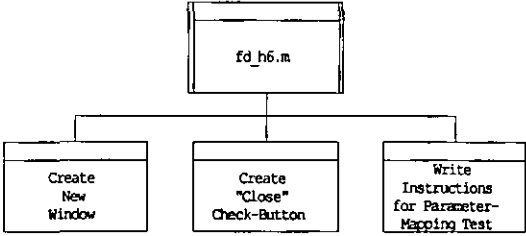
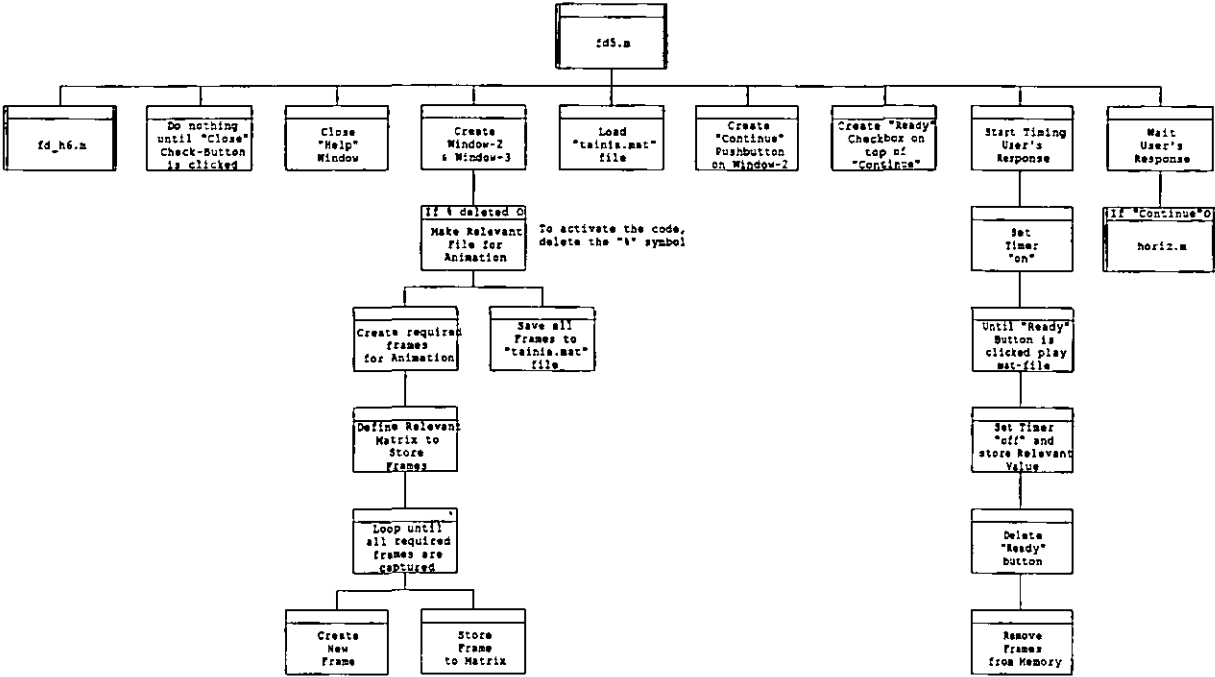




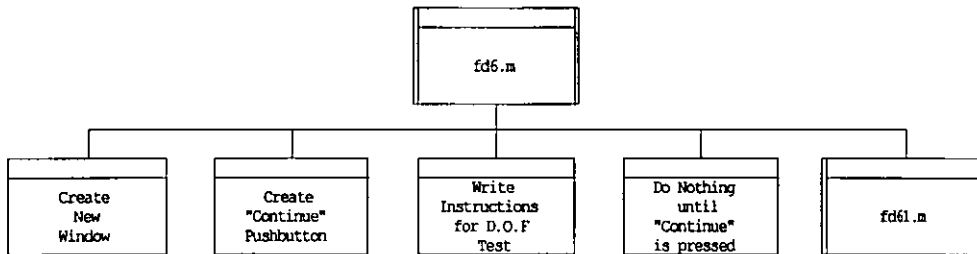
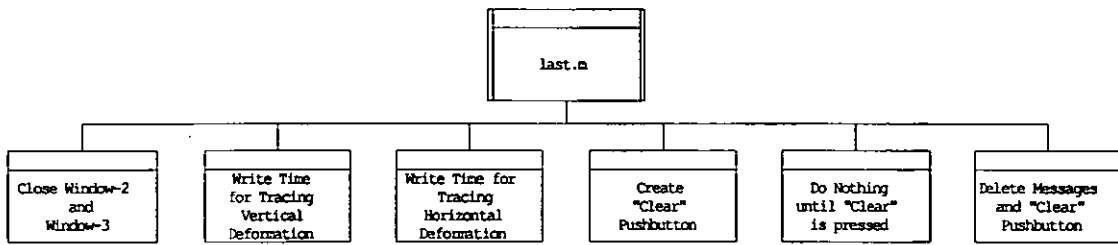




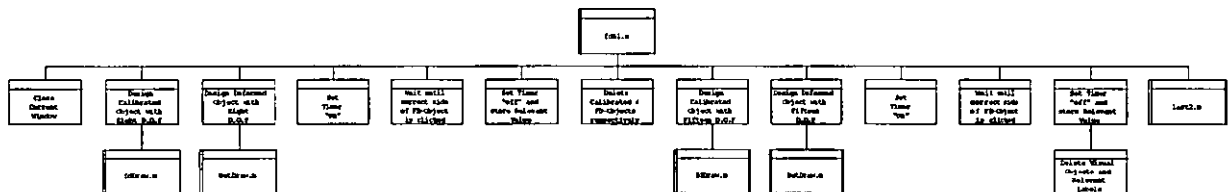




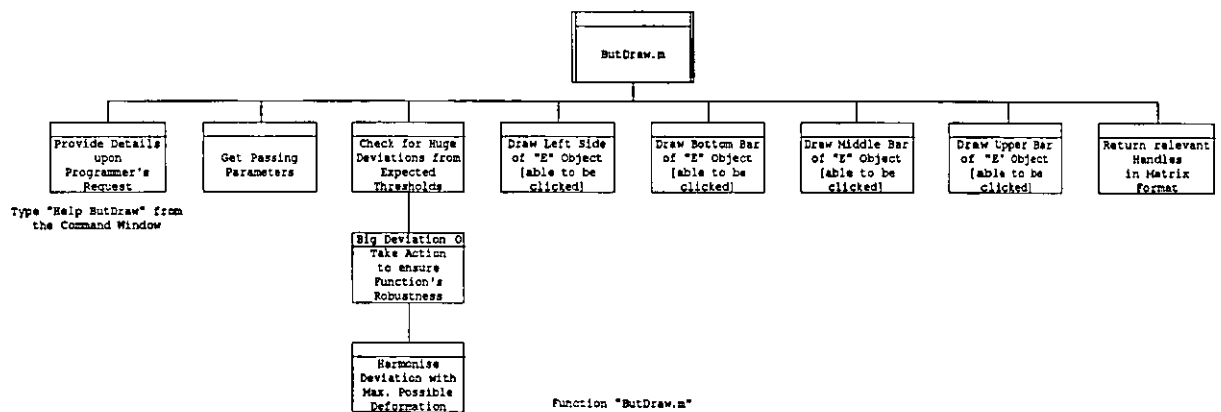
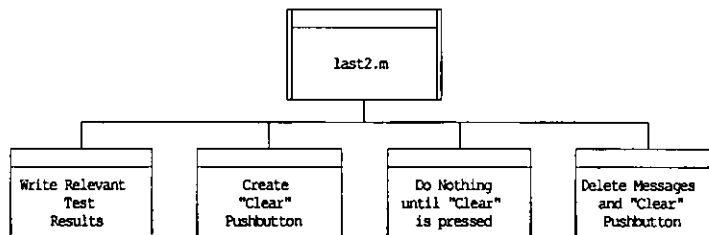
NOTE: "tainia2.mat" is used for horizontal deformation.
"tainis.mat" is used for vertical deformation.



D.O.F : Degrees of Freedom



Notes: The difference of the 2 P-Objects is kept minimal in each case.



APPENDIX –B

Section B.1. Introduction

Appendix-B includes various parts of the written source code for a) the Prototype FD-Interface described earlier in Chapter 5 and b) the FD-Interface with the telescopic antenna visualisation already presented at chapter 3.

Section B.2. Source code for the prototype interface for pilot testing

B.2.1. fd_vs_bar.m

The following m-file creates a user-interface with 10 bar-charts.

```
% *****
% *   FD-INTERFACE EVALUATION           - PART 1-   *
% *****

%   -- Written by Antonis Pagonis --
%   ** M-File-Name: fd_vs_bar.m **

echo off
evaluation = figure('Position',[15,100,1120,860]);

axis('off');
set(evaluation,'color','b');
set(evaluation,'Name','Evaluation Screen','Resize','off');

%-----
%   C R E A T I N G       T E N       B A R - G R A P H S
%-----

uicontrol('Style','text','Position',...
    [350,832,380,21],...
    'string',' -- S M A L L   P A C K E T S --',
    'backgroundcolor',...
    [0.002 0.80 .9]);

uicontrol('Style','text','Position',...
    [350,480 ,380,21],...
    'string',' -- L A R G E   P A C K E T S --',
    'backgroundcolor',...
    [0.002 0.80 .9]);

% Left & Down
Bar1 = axes('Position',[0.04 .24 .16 .26]);
set(Bar1,'title',...
    text(0,0,'Lboro-Lancaster',...
    'FontSize',[11],...
    'FontWeight','Bold'));

% Left & Up
Bar2 = axes('Position',[0.04 .66 .16 .26]);
set(Bar2,'title',...
    text(0,0,'Lboro-Lancaster',...
    'FontSize',[11],...
    'FontWeight','Bold'));
```

```

% Second-Left & Down
Bar3 = axes('Position',[0.23 .24 .16 .26]);
set(Bar3,'title',...
    text(0,0,'Lboro-London',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Second-Left & Up
Bar4 = axes('Position',[0.23 .66 .16 .26]);
set(Bar4,'title',...
    text(0,0,'Lboro-London',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Third-Left & Down
Bar5 = axes('Position',[0.42 .24 .16 .26]);
set(Bar5,'title',...
    text(0,0,'Lboro-Leicester',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Third-Left & Up
Bar6 = axes('Position',[0.42 .66 .16 .26]);
set(Bar6,'title',...
    text(0,0,'Lboro-Leicester',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Fourth-Left & Down
Bar7 = axes('Position',[0.61 .24 .16 .26]);
set(Bar7,'title',...
    text(0,0,'Lboro-Oxford',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Fourth-Left & Up
Bar8 = axes('Position',[0.61 .66 .16 .26]);
set(Bar8,'title',...
    text(0,0,'Lboro-Oxford',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Fifth-Left & Down
Bar9 = axes('Position',[0.8 .24 .16 .26]);
set(Bar9,'title',...
    text(0,0,'Lboro-Luton',...
        'FontSize',[11],...
        'FontWeight','Bold'));

% Fifth-Left & Up
Bar10 = axes('Position',[0.8 .66 .16 .26]);
set(Bar10,'title',...
    text(0,0,'Lboro-Luton',...
        'FontSize',[11],...
        'FontWeight','Bold'));

```



```

% -----
% Creating the console with respective buttons
% -----

uicontrol('Style','Text','Position',...
[5,5,1111,83],'BackgroundColor',[0.7 0.7 0.3]);

% ----- <CLOSE> BUTTON -----

But1 = uicontrol('Style','Pushbutton','Position',...
[990,15,110,60],'Callback','close(gcf)',...
'String','Close','BackgroundColor',[1 0.0 0.0]);

uicontrol('Style','Text','Position',...
[985,67,12,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[985,13,12,9],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[1085,67,9,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[1085,13,9,9],'BackgroundColor',[0.7 0.7 0.3]);

% ----- <HELP> BUTTON -----

But2 = uicontrol('Style','Pushbutton','Position',...
[22,15,110,60],'Callback','fd_help',...
'String','HELP','BackgroundColor',[1.0 1.0 0.0]);

uicontrol('Style','Text','Position',...
[29,67,9,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[29,13,9,9],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[124,67,12,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[124,13,12,9],'BackgroundColor',[0.7 0.7 0.3]);

% ----- <Start Evaluation of Control Group> BUTTON -----

But3 = uicontrol('Style','Pushbutton','Position',...
[134,15,853,60],'Callback','startnow',...
'String',...
'S T A R T I N G   E V A L U A T I O N
O F   C O N T R O L - G R O U P',...
'BackgroundColor',[0 .9 0]);

uicontrol('Style','Text','Position',...
[130,67,10,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[130,13,10,9],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[980,67,12,12],'BackgroundColor',[0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[980,13,12,9],'BackgroundColor',[0.7 0.7 0.3]);

% ----- Pre-declaring variables for startnow.m file -----

```

```

repeat_timing = 1; Intermediate_Results_Matrix = zeros(4,10);
Control_Group_Results_Matrix = zeros(4,2);
N1 = [5500 19000 11800 15000;
      4500 18000 11000 14000;
      4300 17000 11000 14000;
      4300 18200 10900 13900];

N2 = [1000 12000 6880 10900;
      3000 12000 6880 10900;
      1000 12000 7000 11000;
      1200 11000 6000 10000];

N3 = [5500 19500 10800 18000;
      4000 19500 11000 18000;
      3000 16000 10000 15000;
      5500 19500 10000 17000];

N4 = [2200 11000 8000 10000;
      2200 11000 8000 10000;
      2200 11000 8000 10000;
      2100 11100 8300 9000];

N5 = [5000 17000 8800 14000;
      4500 17800 8800 14000;
      4000 16000 8000 13000;
      5000 17000 8800 14000];

N6 = [2000 13000 9800 12000;
      2000 13000 9800 12000;
      3000 13000 9400 11500;
      2000 13000 9800 12000];

N7 = [5900 18000 16800 17000;
      6000 17000 15800 16000;
      5900 18000 16900 17100;
      6500 18000 16900 17000];

N8 = [1670 12500 8900 12000;
      1500 12500 8900 12000;
      1600 12500 8900 12000;
      1400 12000 8000 11000];

N9 = [3010 17500 9900 10435;
      3000 16000 8900 12000;
      2800 12000 9000 12000;
      3000 17500 10000 10400];

N10= [700 14000 6800 10000;
      700 14000 6000 10000;
      500 13000 6500 11000;
      1000 14000 7000 12000];

Correct_Answers_Matrix = [ 1 0 1 0 0 1 1 1 0 1;
                           0 0 1 0 0 1 1 1 0 1;
                           0 0 0 0 0 0 1 1 0 0;
                           0 0 1 0 0 1 1 0 0 1];

```

AxeSize;

B.2.2 startnow.m

The file startnow.m uses the previously created axes to start the presentation of network performance data.

```
% *****
% *   FD-INTERFACE EVALUATION           - PART 2-   *
% *****

% -- Written by Antonis Pagonis --
% ** M-File-Name: startnow.m **

LWid = (8 * XLim)/0.16;
AxW = (3 * XLim)/0.16;
grid_flag = 0;
echo off
big_cover = figure('Position',[15,100,1120,860]);

axis('off');
set(big_cover, 'color', 'black');
set(big_cover, 'Name', 'Preparing routes...', 'Resize', 'off', ...
    'Pointer', 'watch');
uicontrol('Style', 'text', 'Position', ...
    [350,432,380,21], ...
    'string', ' -- P L E A S E   W A I T... --',
    'backgroundcolor', ...
    [0.002 0.60 .9]);

figure(evaluation);
set(evaluation, 'visible', 'off');

% ----- Preparing Routes -----
subplot(Bar1);

set (Bar1, 'YLim', [0,20000], 'LineWidth', [AxW], 'XLim', [0,5], ...
    'XTick', [1,2,3,4], 'XTickLabels', ['Min'; ...
                                         'Max'; ...
                                         'Av. '; ...
                                         '5% '], ...
    'NextPlot', 'add', 'YLabel', ...
    text(0,0, 'Delay in Microseconds', 'FontSize', [10]), ...
    'YTick', [0, 5000, 14000, 17000, 19000]);

M = [1,2,3,4];
[Mb,Nb] = bar(M,N1(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.1 -----
Temp_Bar1 = Bar1;
Temp_Bar1 = plot(Mb,Nb);
set(Temp_Bar1, 'Color', 'green', 'LineWidth', [LWid]);

subplot(Bar2);
```

```

set (Bar2,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',{'Min';...
                                     'Max';...
                                     'Av.';...
                                     '5% '},...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 3000, 11500, 13500 ]});

[Mb,Nb] = bar(M,N2(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.2 -----
Temp_Bar2 = Bar2;
Temp_Bar2 = plot(Mb,Nb);
set(Temp_Bar2,'Color','green','LineWidth',[Lwid]);

subplot(Bar3);

set (Bar3,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',{'Min';...
                                     'Max';...
                                     'Av.';...
                                     '5% '},...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 5000, 14000, 17000, 19000]);

[Mb,Nb] = bar(M,N3(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.3 -----
Temp_Bar3 = Bar3;
Temp_Bar3 = plot(Mb,Nb);
set(Temp_Bar3,'Color','green','LineWidth',[Lwid]);

subplot(Bar4);

set (Bar4,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',{'Min';...
                                     'Max';...
                                     'Av.';...
                                     '5% '},...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 3000, 11500, 13500, 20000]);

[Mb,Nb] = bar(M,N4(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.4 -----
Temp_Bar4 = Bar4;
Temp_Bar4 = plot(Mb,Nb);
set(Temp_Bar4,'Color','green','LineWidth',[Lwid]);

subplot(Bar5);

```

```

set (Bar5,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',['Min';...
                                     'Max';...
                                     'Av. ';...
                                     '5% '],...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 5000, 14000, 17000, 19000]);

[Mb,Nb] = bar(M,N5(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.5 -----
Temp_Bar5 = Bar5;
Temp_Bar5 = plot(Mb,Nb);
set(Temp_Bar5,'Color','green','LineWidth',[Lwid]);

subplot(Bar6);

set (Bar6,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',['Min';...
                                     'Max';...
                                     'Av. ';...
                                     '5% '],...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 3000, 11500, 13500, 20000]);

[Mb,Nb] = bar(M,N6(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.6 -----
Temp_Bar6 = Bar6;
Temp_Bar6 = plot(Mb,Nb);
set(Temp_Bar6,'Color','green','LineWidth',[Lwid]);

subplot(Bar7);

set (Bar7,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',['Min';...
                                     'Max';...
                                     'Av. ';...
                                     '5% '],...
    'NextPlot','add','YLabel',...
    text(0,0,'Delay in Microseconds','FontSize',[10]),...
    'YTick',[0, 5000, 14000, 17000, 19000]);

[Mb,Nb] = bar(M,N7(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.7 -----
Temp_Bar7 = Bar7;
Temp_Bar7 = plot(Mb,Nb);
set(Temp_Bar7,'Color','green','LineWidth',[Lwid]);

subplot(Bar8);

set (Bar8,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
    'XTick',[1,2,3,4],'XTickLabels',['Min';...

```

```

                                'Max';...
                                'Av.';...
                                '5% '],...
'NextPlot','add','YLabel',...
text(0,0,'Delay in Microseconds','FontSize',[10]),...
'YTick',[0, 3000, 11500, 13500, 20000]);

[Mb,Nb] = bar(M,N8(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.8 -----
Temp_Bar8 = Bar8;
Temp_Bar8 = plot(Mb,Nb);
set(Temp_Bar8,'Color','green','LineWidth',[Lwid]);

subplot(Bar9);

set (Bar9,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
'XTick',[1,2,3,4],'XTickLabels',['Min';...
                                'Max';...
                                'Av.';...
                                '5% '],...
'NextPlot','add','YLabel',...
text(0,0,'Delay in Microseconds','FontSize',[10]),...
'YTick',[0, 5000, 14000, 17000, 19000]);

[Mb,Nb] = bar(M,N9(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.9 -----
Temp_Bar9 = Bar9;
Temp_Bar9 = plot(Mb,Nb);
set(Temp_Bar9,'Color','green','LineWidth',[Lwid]);

subplot(Bar10);

set (Bar10,'YLim',[0,20000],'LineWidth',[AxW],'XLim',[0,5],...
'XTick',[1,2,3,4],'XTickLabels',['Min';...
                                'Max';...
                                'Av.';...
                                '5% '],...
'NextPlot','add','YLabel',...
text(0,0,'Delay in Microseconds','FontSize',[10]),...
'YTick',[0, 3000, 11500, 13500, 20000]);

[Mb,Nb] = bar(M,N10(repeat_timing,:));

% --- Storing Handle for Bar-Graph No.10 -----
Temp_Bar10 = Bar10;
Temp_Bar10 = plot(Mb,Nb);
set(Temp_Bar10,'Color','green','LineWidth',[Lwid]);

% ----- DONE WITH ROUTES -----

% ----- COVERING PREVIOUS BUTTONS -----

Cover_button = uicontrol('Style','text','Position',[10,10,1100,70]);

```

```

ready_but =
uicontrol('Style','checkbox','Position',[900,15,200,55],...
    'callback',' ','string','Ready!');

explain_1 = uicontrol('Style','text','Position',[17,45,400,20],...
    'String',...
    'Min = Minimum Packet-Delay [<3,000 & <5,000 ]');
explain_2 = uicontrol('Style','text','Position',[16,20,410,20],...
    'String',...
    ' Max = Maximum Packet-Delay [<13,500 & <19,000]');

explain_3 = uicontrol('Style','text','Position',[388,45,410,20],...
    'String',...
    ' Av = Mean Packet-Delay [<11,500 & <14,000]');

explain_4 = uicontrol('Style','text','Position',[430,20,410,20],...
    'String',...
    ' 5% = Av. Slowest 5% of Packet-Delays [<11,500 & <17,000]');

explain_title =
uicontrol('Style','text','Position',[100,90,710,20],...
    'String',...
    '          T H R E S H O L D -- V A L U E S   F O R   S M A L L / L
A R G E   P A C K E T S',...
    'backgroundcolor',[0.7 0.7 0.3]);

    uicontrol('Style','text','Position',[5,90,98,10],...
        'backgroundcolor',[0.7 0.7 0.3]);

    uicontrol('Style','text','Position',[805,90,218,20],...
        'backgroundcolor',[0.7 0.7 0.3]);

    uicontrol('Style','text','Position',[1000,90,116,10],...
        'backgroundcolor',[0.7 0.7 0.3]);

grid_but = uicontrol
('Style','pushbutton','Position',[900,92,119,16],...
    'String','GRID','callback',...
    [ 'if (grid_flag == 0)',...
      'grid_flag = 1;',...
      'elseif (grid_flag == 1)',...
      'grid_flag = 0;',...
      'end;',...
      'Put_grids')],...
    'backgroundcolor',[0.002 0.7020 0.8020]);
dummy_grid = uicontrol ('Style','text','Position',[900,92,119,16],...
    'String','GRID','backgroundcolor',[0.002 0.7020 0.8020]);
set(ready_but,'Value',0);
pause(2);
delete(big_cover);
set(evaluation,'visible','on');

% ----- Preparing Timing-Loop -----

pause(0.001);
clicked = 0;
tic;
    while (clicked ~= 1)

```

```

pause(0.00001);
clicked = get(ready_but, 'Value');
end;
delete(dummy_grid);

continue_but =
uicontrol('Style','pushbutton','Position',[900,15,200,55],...
    'callback',...
        [ 'if (grid_flag == 1) ',...
          'grid_flag = 0 ;',...
          ' Put_grids;pause(0.0001);',...
          'end ;',...
          'if (repeat_timing < 4) ',...
          'repeat_timing = repeat_timing + 1;',...
          'delete(Temp_Bar1);',...
          'delete(Temp_Bar2);',...
          'delete(Temp_Bar3);',...
          'delete(Temp_Bar4);',...
          'delete(Temp_Bar5);',...
          'delete(Temp_Bar6);',...
          'delete(Temp_Bar7);',...
          'delete(Temp_Bar8);',...
          'delete(Temp_Bar9);',...
          'delete(Temp_Bar10);',...
          'startnow;',...
          'else ',...
          'Done_with_CtrGroup; ',...
          ' end; '],...
    'string','Continue to next screen...');

% ----- CREATING RADIO-BUTTONS FOR IDENTIFYING PROBLEMATIC ROUTES --
---

Finish_with_Clicking = uicontrol('Style','pushbutton','Position',...
    [900,15,200,55],...
    'callback',...
    [ ' Control_Group_Results_Matrix(repeat_timing,1) =
toc;',...
    'Calc_Suc_Per; '],...
    'string',...
    'Finished with current screen');

Route_but(1) =
uicontrol('Style','Checkbox','Position',[72,160,120,20],...
    'string','Not Healthy');

Route_but(2) =
uicontrol('Style','Checkbox','Position',[72,520,120,20],...
    'string','Not Healthy');

Route_but(3) =
uicontrol('Style','Checkbox','Position',[284,160,120,20],...
    'string','Not Healthy');

Route_but(4) =
uicontrol('Style','Checkbox','Position',[284,520,120,20],...
    'string','Not Healthy');

Route_but(5) =
uicontrol('Style','Checkbox','Position',[496,160,120,20],...

```



```

        'string','Not Healthy');

Route_but(6) =
uicontrol('Style','Checkbox','Position',[496,520,120,20],...
        'string','Not Healthy');

Route_but(7) =
uicontrol('Style','Checkbox','Position',[708,160,120,20],...
        'string','Not Healthy');

Route_but(8) =
uicontrol('Style','Checkbox','Position',[708,520,120,20],...
        'string','Not Healthy');

Route_but(9) =
uicontrol('Style','Checkbox','Position',[920,160,120,20],...
        'string','Not Healthy');

Route_but(10) =
uicontrol('Style','Checkbox','Position',[920,520,120,20],...
        'string','Not Healthy');

```

B.2.3. fd1.m

The file fd1.m is used to initiate the assessment of the experimental group. It prepares the relevant labels, object mapping information and calibration controls.

```

% *****
% *   Assessment of EXPERIMENTAL GROUP   -Part I-   *
% *****

%   -- Written by Antonis Pagonis --
%   ** M-File-Name: fd1.m **

echo off
evaluation2 = figure('Position',[15,100,1120,860]);

axis('off');
set(evaluation2,'color','b');
set(evaluation2,'Name','Evaluation Screen:  -PART II-
','Resize','off');

%-----
%  P R E P A R A T I O N   O F   R O U T E   - L A B E L S
%-----

uicontrol('Style','text','Position',...
        [250,822,380,21],...
        'string',' -- S M A L L   P A C K E T S --
','backgroundcolor',...
        [0.002 0.80 .9]);

uicontrol('Style','text','Position',...
        [250,440 ,380,21],...
        'string',' -- L A R G E   P A C K E T S --
','backgroundcolor',...
        [0.002 0.80 .9]);

```

```

text('color','cyan','string',...
    ['Lboro-Lancaster',...
      'Lboro-London',...
      'Lboro-Leicester',...
      'Lboro-Oxford',...
      'Lboro-Luton'],...
'FontSize',[+16],'position',[-.135,1]);

text('color','cyan','string',...
    ['Lboro-Lancaster',...
      'Lboro-London',...
      'Lboro-Leicester',...
      'Lboro-Oxford',...
      'Lboro-Luton'],...
'FontSize',[+16],'position',[-.135,.455]);

% ----- Providing Mapping Information -----

text('color','red','string',...
    'Parameter Mapping','FontSize',[+18],'position',[.889,.457]);

text('color','red','string',...
    'Parameter Mapping','FontSize',[+18],'position',[.890,.4571]);

text('color','m','string',...
    'Average','FontSize',[+20],'position',[.903,.132],'Rotation',[+90]);
text('color','m','string',...
    'Minimum','FontSize',[+19],'position',[.96,.34]);
text('color','m','string',...
    'Maximum','FontSize',[+19],'position',[.96,.224]);
text('color','m','string',...
    'Slowest 5%','FontSize',[+19],'position',[.96,.108]);

fdDraw(950,145,200,350,1,1,1,1,1,1,1,1,0,1,0,13);

% ----- Calibration Section -----

text('color','red','string',...
    'Object Calibration','FontSize',[+18],'position',[.891, 1.046]);

text('color','red','string',...
    'Object Calibration','FontSize',[+18],'position',[.8911, 1.0461]);
blue = 0;
green = 1;
red = 0;
lengthVar = 30;
heightVar = 60;
thickVar = 4;
sensitivity = 1.5;
Model =
fdDraw(933,590,lengthVar,heightVar,1,1,1,1,1,1,1,1,red,green,blue,thi
ckVar);
senseSlider = uicontrol
('Style','slider','Position',[910,90,80,15],...
    'Min', 1, 'Max', 2.1,'Value', 1.5 , 'callback',...

```

```

        ['sensitivity = get(senseSlider,','...
        ''Value'',',...
        ');' ]]);
senselabel = uicontrol
('Style','text','Position',[1000,90,106,31],...
    'string','SENSITIVITY');
senselabel2 = uicontrol
('Style','text','Position',[910,106,80,15],...
    'string','Min      Max', 'HorizontalAlignment','left');
lengthSlider = uicontrol
('Style','slider','Position',[910,555,100,20],...
    'Min', 29, 'Max', 160, 'Value', 30, 'callback',
    'lengthF');
lengthInfo = uicontrol
('Style','text','Position',[1020,555,86,20],...
    'string','LENGHT');
heightSlider = uicontrol
('Style','slider','Position',[910,531,100,20],...
    'Min', 29, 'Max', 290, 'Value', 60, 'callback',
    'heightF');
heightInfo = uicontrol
('Style','text','Position',[1020,531,86,20],...
    'string','HEIGHT');
thickSlider = uicontrol
('Style','slider','Position',[910,507,100,20],...
    'Min', 3, 'Max', 30, 'Value', 4, 'callback',
    'thickF');
thickInfo = uicontrol ('Style','text','Position',[1020,507,86,20],...
    'string','THICKNESS');
colouringRed = uicontrol
('Style','slider','Position',[908,463,62,10],...
    'Min', 0 , 'Max', 1 , 'Value', 0,'callback','fixred');
colouringGreen =
uicontrol('Style','slider','Position',[972,463,68,10],...
    'Min', 0 , 'Max', 1 , 'Value', 1,'callback','fixgreen');
colouringBlue = uicontrol
('style','slider','Position',[1042,463,64,10],...
    'Min', 0 , 'Max', 1 , 'Value', 0,'callback','fixblue');
Redlabel = uicontrol('Style','text','string','0.00','position',...
    [908,474,62,14]);
uicontrol('style','text','string','RED','position',[908,489,62,14]);
Greenlabel = uicontrol('Style','text','string',' 0.00','position',...
    [972,474,68,14]);
uicontrol('style','text','string','GREEN','position',[972,489,68,14])
;
Bluelabel = uicontrol('Style','text','string','0.00','position',...
    [1042,474,64,14]);
uicontrol('style','text','string','BLUE','position',[1042,489,64,14])
;

% -----
%   Creating the console with respective buttons
%

uicontrol('Style','Text','Position',...
    [5,83,7,770],'BackgroundColor',[0.7 0.7 0.3]);

uicontrol('Style','Text','Position',...
    [5,850,1109,7],'BackgroundColor',[0.7 0.7 0.3]);

uicontrol('Style','Text','Position',...

```

```

[900,440,210,21], 'BackgroundColor', [0.7 0.7 0.3]);

uicontrol('Style','Text','Position',...
[5,5,1111,83], 'BackgroundColor', [0.7 0.7 0.3]);

uicontrol('Style','Text','Position',...
[900,83,7,770], 'BackgroundColor', [0.7 0.7 0.3]);

uicontrol('Style','Text','Position',...
[1109,87,7,770], 'BackgroundColor', [0.7 0.7 0.3]);

% ----- <CLOSE> BUTTON -----

But1 = uicontrol('Style','Pushbutton','Position',...
[990,15,110,60], 'Callback','close(gcf)',...
'String','Close','BackgroundColor',[1 0.0 0.0]);

uicontrol('Style','Text','Position',...
[985,67,12,12], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[985,13,12,9], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[1085,67,9,12], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[1085,13,9,9], 'BackgroundColor', [0.7 0.7 0.3]);

% ----- <HELP> BUTTON -----

But2 = uicontrol('Style','Pushbutton','Position',...
[22,15,110,60], 'Callback','fd_help2',...
'String','HELP','BackgroundColor',[1.0 1.0 0.0]);

uicontrol('Style','Text','Position',...
[29,67,9,12], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[29,13,9,9], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[124,67,12,12], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...
[124,13,12,9], 'BackgroundColor', [0.7 0.7 0.3]);

% ----- <Start Evaluation of Experimental Group> BUTTON -----
-

But3 = uicontrol('Style','Pushbutton','Position',...
[134,15,853,60], 'Callback','startnow2',...
'String',...
'S T A R T I N G   E V A L U A T I O N   O F   E X P E R I M
E N T A L - G R O U P',...
'BackgroundColor',[0 .9 0]);

uicontrol('Style','Text','Position',...
[130,67,10,12], 'BackgroundColor', [0.7 0.7 0.3]);
uicontrol('Style','Text','Position',...

```

```

    [130,13,10,9], 'BackgroundColor', [0.7 0.7 0.3]);
    uicontrol('Style','Text','Position',...
    [980,67,12,12], 'BackgroundColor', [0.7 0.7 0.3]);
    uicontrol('Style','Text','Position',...
    [980,13,12,9], 'BackgroundColor', [0.7 0.7 0.3]);

```

```

% ----- Pre-declaring variables for startnow.m file -----

```

```

repeat_timing = 1;
Intermediate_Results_Matrix = zeros(4,10);
Control_Group_Results_Matrix = zeros(4,2);

```

```

N1 = [5500 19000 11800 15000;
      4500 18000 11000 14000;
      4300 17000 11000 14000;
      4300 18200 10900 13900];

```

```

N2 = [1000 12000 6880 10900;
      3000 12000 6880 10900;
      1000 12000 7000 11000;
      1200 11000 6000 10000];

```

```

N3 = [5500 19500 10800 18000;
      4000 19500 11000 18000;
      3000 16000 10000 15000;
      5500 19500 10000 17000];

```

```

N4 = [2200 11000 8000 10000;
      2200 11000 8000 10000;
      2200 11000 8000 10000;
      2100 11100 8300 9000];

```

```

N5 = [5000 17000 8800 14000;
      4500 17800 8800 14000;
      4000 16000 8000 13000;
      5000 17000 8800 14000];

```

```

N6 = [2000 13000 9800 12000;
      2000 13000 9800 12000;
      3000 13000 9400 11500;
      2000 13000 9800 12000];

```

```

N7 = [5900 18000 16800 17000;
      6000 17000 15800 16000;
      5900 18000 16900 17100;
      6500 18000 16900 17000];

```

```

N8 = [1670 12500 8900 12000;
      1500 12500 8900 12000;
      1600 12500 8900 12000;
      1400 12000 8000 11000];

```

```

N9 = [3010 17500 9900 10435;
      3000 16000 8900 12000;
      2800 12000 9000 12000;
      3000 17500 10000 10400];

```

```

N10= [700 14000 6800 10000;
      700 14000 6000 10000;
      500 13000 6500 11000;
      1000 14000 7000 12000];

```

```
Correct_Answers_Matrix = [ 1 0 1 0 0 1 1 1 0 1;
                           0 0 1 0 0 1 1 1 0 1;
                           0 0 0 0 0 0 1 1 0 0;
                           0 0 1 0 0 1 1 0 0 1];
```

B.2.4 fdDraw.m

The function fdDraw.m is used to create the generic “E”-shape objects. As with any other Matlab function, the initial comments can be viewed as a form of help information by typing ‘help fdDraw’ in the command window.

```
function v = fdDraw(a,b,c,d,A,B,C,D,T1,T2,T3,T4,E,F,G,x,sense)
```

```
% *****
% * Function to Visualise Network parameters *
% * by using the Notion of Figural Deformity *
% * (selected visual object: the letter "E") *
% *****
%
% -- Written by Antonis Pagonis --
% ** M-File-Name: fdDraw.m **
%
% USAGE: { HANDLE = fdDraw(a,b,c,d,A,B,C,D,E,F,G,x) }
%
% (a,b,c,d are declaring the local window area)
%
% a = distance from left-side
% b = distance from bottom
% c = window length
% d = window height
%
% (A-D: Network Data parameters)
% (E-F: Colour Inputs [input range: 0-1 inclusive])
%
% A = Min. Packet Delay
% B = Max. Packet Delay
% C = Average Packet Delay
% D = average of the 5% slowest Packet Delays
% T1 = Threshold Value for Min. Delay
% T2 = Threshold Value for Max. Delay
% T3 = Threshold value for Mean. Delay
% T4 = Threshold Value for average slowest 5%
% E = Colour Input 1 (red)
% F = Colour Input 2 (Green)
% G = Colour Input 3 (Blue)
% x = width of visual object
% sense = sensitivity (Zooming Factor)
%
% Checking for any Sensitivity Requirements
%
if (nargin < 17)
    sense = 1;
```

```

end

% --- Checking for Huge-Deviations from Input-Thresholds ---
if (A > T1)
    A = A*sense;
end;

if (B > T2)
    B = B*sense;
end;

if (C>T3)
    C = C*sense;
end;

if (D>T4)
    D = D*sense;
end;

if (C > (2*T3))
    C = 2*T3;
end;

if (B > (2*T2))
    B = 2*T2;
end;

if (D > (2*T4))
    D = 2*T4;
end;

if (A > (2*T1))
    A = 2*T1;
end;

% --- Calibrated Size at 50% of window-area for Threshold values ---
% ---

% -----
% -- Designing the "E" object ---
% -----

%           -- Designing the left side of "E" object --

v(1) = uicontrol ('Style','text','Position',...
    [a,...
    b,...
    x,...
    ((C * d/2)/T3)],...
    'backgroundcolor',[E,F,G]);

%           -- Designing the bottom bar of the "E" object --

v(2) = uicontrol ('Style','text','Position',...
    [(a+x),...
    b,...
    ((B* c/2)/T2),...
    x],...
    'backgroundcolor',[E,F,G]);

%           -- Designing the middle bar of the "E" object --

```

```

v(3) = uicontrol ('Style','text','Position',...
    [(a+x),...
    (b+(((C*d/2)/T3)/2) ) - (x/2) ,...
    ((D* c/2)/T4),...
    x],...
    'backgroundcolor',[E,F,G]);

%      -- Designing the upper bar of the "E" object --

v(4) = uicontrol ('Style','text','Position',...
    [(a+x),...
    (b+ ( (C * d/2)/T3 )-x),...
    ((A* c/2)/T1),...
    x],...
    'backgroundcolor',[E,F,G]);

```

Section B.3. Source code for the telescopic antenna visualisation prototype

B.3.1. dokimi.m

The file dokimi.m is used to create the initial interface platform with the relevant user controls and FD-objects.

```

% *****
% *   File To Create FD-Interfaces with an   *
% *       alternative Visual Object         *
% *****

% File Name: - Dokimi.m -
% Written by ## Antonis Pagonis ###

param(7,13)=zeros;

param(1,:) = 'Max. Delay   ';
param(2,:) = 'Min. Delay   ';
param(3,:) = 'Median Delay ';
param(4,:) = 'Average Delay';
param(5,:) = 'St. Deviation';
param(6,:) = 'Slowest 5%   ';
param(7,:) = 'Fastest 95%  ';

Main = figure ('Position',[100,100,900,860]);
set(gcf,'color','w','Name','Alternative Visual
Object','Resize','off');
axis('off');

LabTxt1 = uicontrol ('style','text','position',[1,276,900,17],...
    'backgroundcolor',[.9 .7 .6],...

```



```

        'HorizontalAlignment','left','string',...
    [ 'Lancaster      ',...
      'Oxford         ',...
      '  London       ',...
      '    Luton       ',...
      'Leicester      ',...
      '  Cardiff      ',...
      '    Walsall     ',...
      '  Nottingham   ',...
      'Birmingham     ',...
      'Aberdeen        ']);

LabTxt2 = uicontrol ('style','text','position', [1,533,900,17],...
                    'backgroundcolor',[ .8 .7 .6],...
                    'HorizontalAlignment','left',...
                    'string',...
                    ['Ref. Point: Loughborough
                    ',...
                    '
                    L A R G E   P A C K E T S']]);

LabTxt3 = uicontrol ('style','text','position', [1,845,900,17],...
                    'backgroundcolor',[ .8 .85 .6],...
                    'HorizontalAlignment','left',...
                    'string',...
                    ['Ref. Point: Loughborough
                    ',...
                    '
                    S M A L L   P A C K E T S']]);

% Visual Ruler

where2 = 841;
where1 = 529;
Ruler2 = uicontrol ('style','text','position', [1,where2,900,4],...
                    'backgroundcolor',[ .8, .85, .6]);
Ruler1 = uicontrol ('style','text','position', [1,where1,900,4],...
                    'backgroundcolor',[ .8, .7, .6]);

x= 0.06;
y = 0.03;
p = [ 10 10 10 10 10 10 10];
c = [ 1 0.3 1];
l = [ 15 15 15 15 15 15 15];
m = .05;
n = .06;
r = 10;
on = 0;

% DEMO OF COLLECTION

p2 = [ 18 10 16 10 10 17 13];

g1 = Antenna (.01,0.34,p,c,l,m,n,r,0);
g2 = Antenna (.11,0.34,p2,c,l,m,n,r,0);
g3 = Antenna (.21,0.34,p,c,l,m,n,r,0);
g4 = Antenna (.31,0.34,p,c,l,m,n,r,0);
g5 = Antenna (.41,0.34,p,c,l,m,n,r,0);
g6 = Antenna (.51,0.34,p2,c,l,m,n,r,0);
g7 = Antenna (.61,0.34,p,c,l,m,n,r,0);
g8 = Antenna (.71,0.34,p,c,l,m,n,r,0);

```

```

g9 = Antenna (.81,0.34,p2,c,l,m,n,r,0);
g10 = Antenna (.91,0.34,p,c,l,m,n,r,0);
g12 = Antenna (.01,0.64,p,c,l,m,n,r,0);
g13 = Antenna (.11,0.64,p2,c,l,m,n,r,0);
g14 = Antenna (.21,0.64,p,c,l,m,n,r,0);
g15 = Antenna (.31,0.64,p,c,l,m,n,r,0);
g16 = Antenna (.41,0.64,p,c,l,m,n,r,0);
g17 = Antenna (.51,0.64,p2,c,l,m,n,r,0);
g18 = Antenna (.61,0.64,p,c,l,m,n,r,0);
g19 = Antenna (.71,0.64,p,c,l,m,n,r,0);
g20 = Antenna (.81,0.64,p2,c,l,m,n,r,0);
g21 = Antenna (.91,0.64,p,c,l,m,n,r,0);

% Mapping-Object
g11 = Antenna (.88,0.03,[10 10 10 10 10 10 10],c,...
    [25 25 25 25 25 25 25],0.08, 0.15,r,0);

UPLAT = uicontrol ('Style','text','Position',[200,1,545,260],...
    'backgroundcolor',[.5 .5 .3]);

BackCol1 =uicontrol('Style','pushbutton','Position',...
    [225,30,65 ,20],'String','White','callback',...
    [ ' set(Main,',...
      '''color''','',',',...
      '''white''',...
      ');']);

BackCol2 =uicontrol('Style','pushbutton','Position',...
    [225,57,65 ,20],'String','Yellow','callback',...
    [ ' set(Main,',...
      '''color''','',',',...
      '''yellow''',...
      ');']);

BackCol3 =uicontrol('Style','pushbutton','Position',...
    [225,84,65 ,20],'String','Blue','callback',...
    [ ' set(Main,',...
      '''color''','',',',...
      '''blue''',...
      ');']);

BackCol4 =uicontrol('Style','pushbutton','Position',...
    [225,111,65 ,20],'String','Magenta','callback',...
    [ ' set(Main,',...
      '''color''','',',',...
      '''Magenta''',...
      ');']);

BackCol5 =uicontrol('Style','pushbutton','Position',...
    [225,138,65 ,20],'String','Black','callback',...
    [ ' set(Main,',...
      '''color''','',',',...
      '''black''',...
      ');']);

BackCol6 =uicontrol('Style','pushbutton','Position',...

```

```

[225,165,65,20], 'String', 'Cyan', 'callback', ...
[ ' set(Main, ', ...
  '''color''', ', ', ...
  '''cyan''', ...
  '); ']);

MoveRuler2 = uicontrol
('style', 'slider', 'position', [225,224,65,13], ...
  'Min', 550, 'Max', 841, 'value', 841,
'callback', ...
  [
    'where2 = get(MoveRuler2, ', ...
    '''value''', ...
    '); ', ...
    'set(Ruler2, ', ...
    '''position''', ...
    ', [1,where2,900,4] ', ...
    '); ']);

MoveRuler1 = uicontrol
('style', 'slider', 'position', [225,210,65,13], ...
  'Min', 290, 'Max', 529, 'value', 529,
'callback', ...
  [
    'where1 = get(MoveRuler1, ', ...
    '''value''', ...
    '); ', ...
    'set(Ruler1, ', ...
    '''position''', ...
    ', [1,where1,900,4] ', ...
    '); ']);

RulerLabel = uicontrol
('style', 'text', 'string', 'Rulers', 'position', ...
  [225,237,65,14]);
RulerLabel2 = uicontrol ('style', 'text', 'string', 'Down
Up', 'position', ...
  [225,196,65,14]);

g = Antenna (x,y,p,c,l,m,n,r,0);

Howmany = size(p);

AdjustLabel = uicontrol ('Style',
'text', 'Position', [205,1,105,25], ...
  'string', 'Window Color');

AdjustLabel2 = uicontrol ('Style',
'text', 'Position', [310,1,200,20], ...
  'string', 'DEMO');

AdjustLabel3 = uicontrol ('Style',
'text', 'Position', [510,1,230,25], ...
  'string', 'Set Object Color & Shape');

```

```

for i =1:Howmany(1,2)

Slider(i) = uicontrol
('Style','slider','Position',[410,(25*i),100,20],...
'Min', 4, 'Max', 16, 'Value', 10, 'callback',...
[ ' for i=1:Howmany(1,2)',...
' HowTal = floor(get(Slider(i),'...
''Value'','',...
'))','...',...
'p(1,i) = HowTal;','...',
'end; ','...',
'delete(g);','...',
'g = Antenna(x,y,p,c,l,m,n,r,on);']);

SlidLabel(i) = uicontrol
('Style','text','Position',[310,(25*i),100,20],...
'string',[param(i,:)],'backgroundcolor',[.8 .8 .3]);

Map(i) = uicontrol ('Style','text','Position',...
[820,(31*(i*1.1)),12,17],...
'string',[num2str(i)],'backgroundcolor',[.8 .8 .3]);

end;


L1 = uicontrol ('Style','text','Position',...
[ 10,4,170,20],...
'string','OBJECT CALIBRATION','backgroundcolor',[.8
.8 .3] );

L2 = uicontrol ('Style','text','Position',...
[ 780,4,100,20],...
'string','MAPPING','backgroundcolor',[.8 .8 .3]);

L3 = uicontrol ('Style','text','Position',...
[ 630, 130, 100,20],...
'string','HEIGHT');

SlidHeight = uicontrol
('style','slider','Position',[530,130,100,20],...
'Min', 0.03 , 'Max', 0.1, 'Value' , 0.06,
'callback',...
[' HeightVal = get(SlidHeight','...
''Value'','',...
)];',...
' n = HeightVal; ','...',
'delete(g, g1,g2,g3,g4,g5,g6,g7,g8,g9,g10','...',
' g12,g13,g14,g15,g16,g17,g18,g19,g20,g21
)';',...
'g = Antenna(x,y,p,c,l,m,n,r,on);','...',
' g1 = Antenna
(.01,0.34,p,c,l,m,n,r,on);','...',
' g2 = Antenna
(.11,0.34,p2,c,l,m,n,r,on);','...',

```

```

        ' g3 = Antenna
(.21,0.34,p,c,l,m,n,r,on);',...
        ' g4 = Antenna
(.31,0.34,p,c,l,m,n,r,on);',...
        ' g5 = Antenna
(.41,0.34,p,c,l,m,n,r,on);',...
        ' g6 = Antenna
(.51,0.34,p2,c,l,m,n,r,on);',...
        ' g7 = Antenna
(.61,0.34,p,c,l,m,n,r,on);',...
        ' g8 = Antenna
(.71,0.34,p,c,l,m,n,r,on);',...
        ' g9 = Antenna
(.81,0.34,p2,c,l,m,n,r,on);',...
        ' g10 = Antenna
(.91,0.34,p,c,l,m,n,r,on);',...
        ' g12 = Antenna
(.01,0.64,p,c,l,m,n,r,on);',...
        ' g13 = Antenna
(.11,0.64,p2,c,l,m,n,r,on);',...
        ' g14 = Antenna
(.21,0.64,p,c,l,m,n,r,on);',...
        ' g15 = Antenna
(.31,0.64,p,c,l,m,n,r,on);',...
        ' g16 = Antenna
(.41,0.64,p,c,l,m,n,r,on);',...
        ' g17 = Antenna
(.51,0.64,p2,c,l,m,n,r,on);',...
        ' g18 = Antenna
(.61,0.64,p,c,l,m,n,r,on);',...
        ' g19 = Antenna
(.71,0.64,p,c,l,m,n,r,on);',...
        ' g20 = Antenna
(.81,0.64,p2,c,l,m,n,r,on);',...
        ' g21 = Antenna (.91,0.64,p,c,l,m,n,r,on);'
]);

ManualSliderCover = uicontrol ('style','text','position',
[412,24,98,176],...
    'backgroundcolor',[ .8 .8 .3]);

L4 = uicontrol ('style','checkbox','position', [310,200,200,20],...
    'string','Manual Control','callback', 'cover',...
    'backgroundcolor',[ .8 .8 .3]);

% -- ADJUSTING HEIGHT & LENGTH OF VISUAL OBJECT --

ThickSl = uicontrol ('Style','slider','Position',
[530,105,100,20],...
    'Min', .02,'Max', .10,'Value',0.05,'callback',...
    [' HowThick = get(ThickSl,','...
    ''Value'','...
    ');',...
    ' m = HowThick;',...
    'delete(g, g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,','...
    '
g12,g13,g14,g15,g16,g17,g18,g19,g20,g21);',...

```

```

        'g = Antenna(x,y,p,c,l,m,n,r,on);',...
        ' g1 = Antenna
(.01,0.34,p,c,l,m,n,r,on);',...
        ' g2 = Antenna
(.11,0.34,p2,c,l,m,n,r,on);',...
        ' g3 = Antenna
(.21,0.34,p,c,l,m,n,r,on);',...
        ' g4 = Antenna
(.31,0.34,p,c,l,m,n,r,on);',...
        ' g5 = Antenna
(.41,0.34,p,c,l,m,n,r,on);',...
        ' g6 = Antenna
(.51,0.34,p2,c,l,m,n,r,on);',...
        ' g7 = Antenna
(.61,0.34,p,c,l,m,n,r,on);',...
        ' g8 = Antenna
(.71,0.34,p,c,l,m,n,r,on);',...
        ' g9 = Antenna
(.81,0.34,p2,c,l,m,n,r,on);',...
        'g10 = Antenna
(.91,0.34,p,c,l,m,n,r,on);',...
        ' g12 = Antenna
(.01,0.64,p,c,l,m,n,r,on);',...
        ' g13 = Antenna
(.11,0.64,p2,c,l,m,n,r,on);',...
        ' g14 = Antenna
(.21,0.64,p,c,l,m,n,r,on);',...
        ' g15 = Antenna
(.31,0.64,p,c,l,m,n,r,on);',...
        ' g16 = Antenna
(.41,0.64,p,c,l,m,n,r,on);',...
        ' g17 = Antenna
(.51,0.64,p2,c,l,m,n,r,on);',...
        ' g18 = Antenna
(.61,0.64,p,c,l,m,n,r,on);',...
        ' g19 = Antenna
(.71,0.64,p,c,l,m,n,r,on);',...
        ' g20 = Antenna
(.81,0.64,p2,c,l,m,n,r,on);',...
        ' g21 = Antenna (.91,0.64,p,c,l,m,n,r,on);'
1);

ThickLabel = uicontrol ('Style',
'text','Position',[630,105,100,20],...
        'string','Thickness');

Slred = uicontrol ('style','slider','Position',[530,80,100,20],...
        'Min', 0 , 'Max', 1 , 'Value' , .5, 'callback',...
        [' HowRed = get(Slred,','...
        ''Value'','',...
        ');',...
        ' c(1) = HowRed; ',...
        'colormap(c)']);

RedLabel = uicontrol ('Style', 'text','Position',[630,80,100,20],...
        'string','RED');

```

```

Slblue = uicontrol ('style','slider','Position',[530,55,100,20],...
    'Min', 0 , 'Max', 1 , 'Value' , .5, 'callback',...
    [' HowBlue = get(Slblue,','...
        ''Value'',...
    ');','...
    ' c(3) = HowBlue; ','...
    'colormap(c)']);

BlueLabel = uicontrol ('Style', 'text','Position',[630,55,100,20],...
    'string','BLUE');

Slgreen = uicontrol ('style','slider','Position',[530,30,100,20],...
    'Min', 0 , 'Max', 1 , 'Value' , .5, 'callback',...
    [' HowGreen = get(Slgreen,','...
        ''Value'',...
    ');','...
    ' c(2) = HowGreen; ','...
    'colormap(c)']);

GreenLabel = uicontrol ('Style',
    'text','Position',[630,30,100,20],...
    'string','GREEN');

ShadThick = uicontrol
    ('style','slider','Position',[530,155,100,20],...
    'Min', 0.003 , 'Value' ,
    0.003,'visible','off',...
    'callback','Shad2','backgroundcolor',[ 1 0 0 ]);

% Notice that the following checkbox is responsible
% for setting the Max. value of the ShadThick slider
% by calling Shad.m

Onslider = uicontrol ('Style',
    'checkbox','Position',[630,155,100,20],...
    'string','VIEW','callback',...
    [' if (on == 1)',...
    ' on = 0;','...
    'set(ShadThick,','...
    ''visible'',...
    ',','...
    ''off'',...
    ');','...
    'else ',...
    ' on = 1;','...
    'set(ShadThick,','...
    ''visible'',...
    ',','...
    ''on'',...
    ');','...
    'end;','...
    'delete(g,
g1,g2,g3,g4,g5,g6,g7,g8,g9,g10,','...
g12,g13,g14,g15,g16,g17,g18,g19,g20,g21);','...
    'g = Antenna(x,y,p,c,l,m,n,r,on);','...
    ' g1 = Antenna
(.01,0.34,p,c,l,m,n,r,on);','...

```

```

        ' g2 = Antenna
(.11,0.34,p2,c,l,m,n,r,on);',...
        ' g3 = Antenna
(.21,0.34,p,c,l,m,n,r,on);',...
        ' g4 = Antenna
(.31,0.34,p,c,l,m,n,r,on);',...
        ' g5 = Antenna
(.41,0.34,p,c,l,m,n,r,on);',...
        ' g6 = Antenna
(.51,0.34,p2,c,l,m,n,r,on);',...
        ' g7 = Antenna
(.61,0.34,p,c,l,m,n,r,on);',...
        ' g8 = Antenna
(.71,0.34,p,c,l,m,n,r,on);',...
        ' g9 = Antenna
(.81,0.34,p2,c,l,m,n,r,on);',...
        ' g10 = Antenna
(.91,0.34,p,c,l,m,n,r,on);',...
        ' g12 = Antenna
(.01,0.64,p,c,l,m,n,r,on);',...
        ' g13 = Antenna
(.11,0.64,p2,c,l,m,n,r,on);',...
        ' g14 = Antenna
(.21,0.64,p,c,l,m,n,r,on);',...
        ' g15 = Antenna
(.31,0.64,p,c,l,m,n,r,on);',...
        ' g16 = Antenna
(.41,0.64,p,c,l,m,n,r,on);',...
        ' g17 = Antenna
(.51,0.64,p2,c,l,m,n,r,on);',...
        ' g18 = Antenna
(.61,0.64,p,c,l,m,n,r,on);',...
        ' g19 = Antenna
(.71,0.64,p,c,l,m,n,r,on);',...
        ' g20 = Antenna
(.81,0.64,p2,c,l,m,n,r,on);',...
        ' g21 = Antenna (.91,0.64,p,c,l,m,n,r,on);',
'Shad'] );

ButEnd = uicontrol ('style','pushbutton','Position',
[530,190,200,50],...
    'string','CLOSE CURRENT SCREEN','callback',...
    'close(gcf)','...
    'backgroundcolor',[ .9 .7 .3]);

StartDraw = uicontrol ('style','pushbutton','Position',
[310,222,200,28],...
    'string','Visual Truncation','callback',...
    'VisTrunc','backgroundcolor',[ .5 .7 .3]);

```

B.3.2 Antenna.m

The function Antenna.m is responsible for drawing the relevant antennas on the previous interface in order to visualise seven parameters per network route.


```

function Ant = Antenna(x,y,p,c,l,m,n,r,switch2,shadow)

% *****
% *      Function to Visualise Network Parameters      *
% *      by using the Notion of Figural Deformity      *
% *      (selected Visual Object: Telescopic Antenna)  *
% *****
%
% - Written by Antony Pagonis -
% ** M-File-Name: Antenna.m **
%
% Usage: { Handle = Antenna(x,y,p,c,l,m,n,r,switch2,shadow) }
%
% x = X Coordinate for draw object
% y = Y Coordinate for draw object
% p = input vector for network parameter values
% c = input vector for colormap
% l = input vector for network parameter thresholds
% m = width of base-cylinder
% n = height of base-cylinder
% r = resolution
% switch2 = View of Cylinders
% shadow = Adjusting the Width of the shadow when switch2 in ON
% Shadow's Unit-Type: {Normalised}
echo off

% Presetting Minimum Width of red-shadow for inverted cylinders

if (nargin < 10)
    shadow = .003;
end

pieces = size(p);

% Checking and fixing Huge deviations

for i=1:pieces(1,2)
    if (p(i) > (2*l(i))) p(i) = 2*l(i)
    end;
    if (p(i) < 0.001) p(i) = 0.001
    end;
end;

right = 0;
up = 0;
left = 0;

% (The higher the value...the closer the objects)

elev = 0.4;
switch1 = 0;
Prevheight = 0;

% --- Designing the Object ---

for i = 1:pieces(1,2)
    height = ((n*p(i))/l(i));

% Avoiding Axes Dimension to be zero...for too many pieces

```

```

if (m-left) == 0
    left = m-0.001;
end

% ## Switch2 is used for change the view of the object ##

if (switch2 == 1)
    right = 0;
    left = 0;
    height = n/4;
    m = ((n*p(i))/l(i));

    Ant(2,i) = uicontrol('style','text','BackgroundColor',[1 0 0.0]);
    set(Ant(2,i),'Units','normalized','Position',...
        [x,(y+up)+0.006 , shadow, n/5] );
end

Ant(1,i) = axes('Position',[(x+right), (y+up) (m-left) height]);
[X,Y,Z] = cylinder(r);
if (switch2 == 1)
    T = Z;
    Z = Y;
    Y = T;
end
    if (switch1 == 0)
        surf(X,Y,Z);axis off;
        switch1 = 1;
    else
        mesh(X,Y,Z);axis off;
        switch1 = 0;
    end

right = right + (0.07 * m);

LastPieceEnd = up + (height-(elev*height)/2);
up = up + (height - (elev * height));
left = left + (0.16*m);
Prevheight = height;

end;

% Positioning the top piece at the very end of the Antenna

if (switch2 == 0)
    pieces(1,2) = pieces(1,2)+1;
    right = right - 3*(0.07 * m);
    left = left - 3*(0.16*m);

    Ant(1,pieces(1,2)) = axes('Position',[(x+right),...
        (y+LastPieceEnd) (m-left) (n/5) ]);

    [X,Y,Z] = sphere(r);
    surf(X,Y,Z); axis off;

colormap (c );
else
axis off;
end

```

APPENDIX –C

Section C.1. t-Distribution critical values [Wal93]

ν	α						
	0.40	0.30	0.20	0.15	0.10	0.05	0.025
1	0.325	0.727	1.376	1.963	3.078	6.314	12.706
2	0.289	0.617	1.061	1.386	1.886	2.920	4.303
3	0.277	0.584	0.978	1.250	1.638	2.353	3.182
4	0.271	0.569	0.941	1.190	1.533	2.132	2.776
5	0.267	0.559	0.920	1.156	1.476	2.015	2.571
6	0.265	0.553	0.906	1.134	1.440	1.943	2.447
7	0.263	0.549	0.896	1.119	1.415	1.895	2.365
8	0.262	0.546	0.889	1.108	1.397	1.860	2.306
9	0.261	0.543	0.883	1.100	1.383	1.833	2.262
10	0.260	0.542	0.879	1.093	1.372	1.812	2.228
11	0.260	0.540	0.876	1.088	1.363	1.796	2.201
12	0.259	0.539	0.873	1.083	1.356	1.782	2.179
13	0.259	0.537	0.870	1.079	1.350	1.771	2.160
14	0.258	0.537	0.868	1.076	1.345	1.761	2.145
15	0.258	0.536	0.866	1.074	1.341	1.753	2.131
16	0.258	0.535	0.865	1.071	1.337	1.746	2.120
17	0.257	0.534	0.863	1.069	1.333	1.740	2.110
18	0.257	0.534	0.862	1.067	1.330	1.734	2.101
19	0.257	0.533	0.861	1.066	1.328	1.729	2.093
20	0.257	0.533	0.860	1.064	1.325	1.725	2.086
21	0.257	0.532	0.859	1.063	1.323	1.721	2.080
22	0.256	0.532	0.858	1.061	1.321	1.717	2.074
23	0.256	0.532	0.858	1.060	1.319	1.714	2.069
24	0.256	0.531	0.857	1.059	1.318	1.711	2.064
25	0.256	0.531	0.856	1.058	1.316	1.708	2.060
26	0.256	0.531	0.856	1.058	1.315	1.706	2.056
27	0.256	0.531	0.855	1.057	1.314	1.703	2.052
28	0.256	0.530	0.855	1.056	1.313	1.701	2.048
29	0.256	0.530	0.854	1.055	1.311	1.699	2.045
30	0.256	0.530	0.854	1.055	1.310	1.697	2.042
40	0.255	0.529	0.851	1.050	1.303	1.684	2.021
60	0.254	0.527	0.848	1.045	1.296	1.671	2.000
120	0.254	0.526	0.845	1.041	1.289	1.658	1.980
∞	0.253	0.524	0.842	1.036	1.282	1.645	1.960